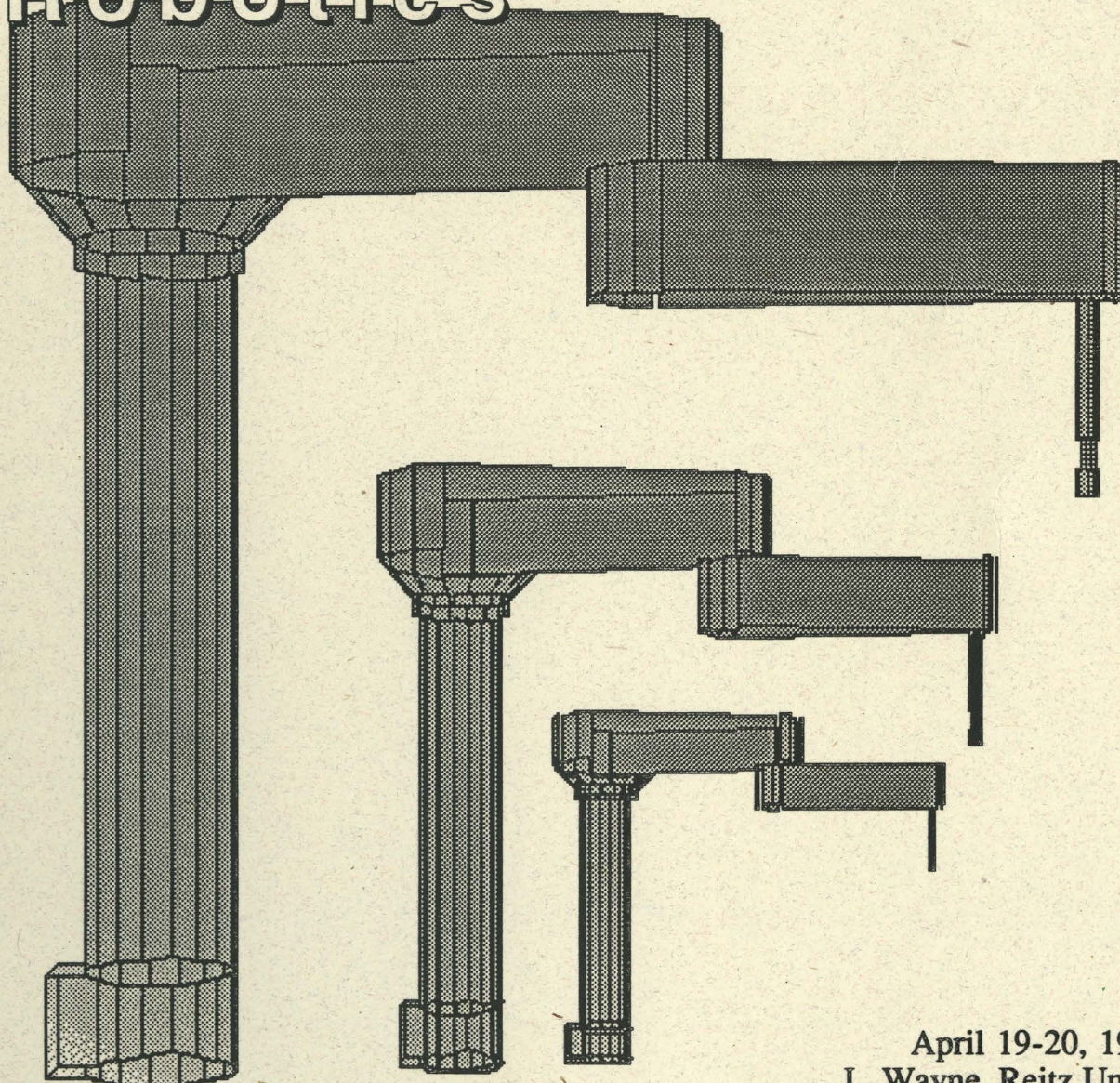


Proceedings

Sixth Annual Conference on Recent Advances in Robotics



April 19-20, 1993
J. Wayne Reitz Union
University of Florida
Gainesville, Florida

The papers appearing in this book comprise the *Proceedings of the Sixth Annual Conference on Recent Advances in Robotics*. They reflect the authors' opinions and are published as presented and without change, in the interests of timely dissemination. Their inclusion in this publication does not necessarily constitute endorsement by the editors.

Cover designed by K. Cephus, K. Doty, P. Fericola and E. Schwartz

Reprint Permission: This document may be reprinted in part or in its entirety with full credit to the source and authors.



UNIVERSITY OF FLORIDA

JOHN V. LOMBARDI
PRESIDENT

April 19, 1993

Dear Friends:

I am delighted to welcome the Sixth Annual Conference on Recent Advances in Robotics to the University of Florida campus.

It is a privilege for the University of Florida to serve as a hub at such an exciting time of technological advances. This is the first year we have hosted the conference, and I look forward to participating in the activities that will help spur our country towards innovative leadership in the robotics industry.

This conference is a chance for the University of Florida community to be an important part of the accelerating changes that will soon encompass the whole world. This is a time to reflect on the developments that scientists and engineers in both academia and industry have made in robotics, manufacturing, automation, machine intelligence and space applications within the past few decades and to anticipate the discoveries yet to be made.

I hope you also will use this time to explore our campus and the resources we have here for technological research and development. We are happy to serve as a meeting ground for leaders of the robotic community in the state of Florida.

On behalf of the faculty, students and staff of the University of Florida, I extend a hearty welcome and wish you all a successful and productive conference.

Sincerely yours,

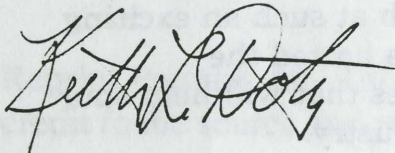
John V. Lombardi, President

CONFERENCE CHAIR'S MESSAGE

The University of Florida welcomes participants and guests to the *Sixth Annual Conference on Recent Advances in Robotics*. This yearly conference, established by Zvi Roth and his colleagues at Florida Atlantic University five years ago, provides an opportunity for researchers, industrialists and State government employees to meet and discuss the state of robotics, manufacturing, automation and machine intelligence, in the State of Florida. This year I propose we establish and charter a *Florida Robotics Society* whose objectives would be 1) to help sponsor these yearly meetings, 2) to provide a consultant service to industry and the state, 3) to act as an advisory body to State government concerning technical matters in robotics and manufacturing, 4) to encourage the development of robotics and high-tech manufacturing in the State, 5) to expand graduate programs in robotics to include possible internships, and, in general, 6) to keep Florida in the vanguard of robotics, high-tech manufacturing and machine intelligence. Such an endeavor can only be successful, of course, with your support and enthusiasm. I look forward to meeting all of you and wish all a successful conference.

Acknowledgement:

Many thanks to Kimberly Cephus and Eric Schwartz for their hard work in planning and executing the administrative functions so necessary for making this conference a reality. Thanks to Professor Joe Duffy and CIMAR and the Department of Electrical Engineering, University of Florida for their support and, of course, to you the presenters for making it all possible.



Keith L. Doty
Conference Chairman,
*Sixth Annual Conference on
Recent Advances in Robotics*,
April 19, 1993
Gainesville, Florida

CONFERENCE COMMITTEE

Chair: Keith L. Doty
Coordinator: Kimberly G. Cephus
Coordinator: Eric M. Schwartz

*Machine Intelligence Laboratory
University of Florida*

PROGRAM COMMITTEE

Kimberly G. Cephus, *MIL-UF*
Carl D. Crane III, *CIMAR-UF*
Keith L. Doty, *MIL-UF*
Joseph Duffy, *CIMAR-UF*
Ming Z. Huang, *FAU*
Zvi Roth, *FAU*
Eric M. Schwartz, *MIL-UF*

SPONSORS

*Center for Intelligent Machines and Robotics (CIMAR)
University of Florida*

*Machine Intelligence Laboratory (MIL)
University of Florida*

*Department of Electrical Engineering
University of Florida*

AUTHOR INDEX

Author	Session/Paper
Adjouadi, Malek	2D/2
Banks, Kent	1A/1
Beck, Hal	1C/2
Brzakovic, Dragana	1C/2
Candocia, Frank	2D/2
Crane, Carl D. III	2A/1
Doty, Keith L.	1B/1,2B/1
Duffy, Joseph	1A/2,2C/3
Fernandez, E.B.	1C/3
Griffis, Michael W.	1A/2
Han, C. P.	1C/3
Harrell, Roy	2A/3
Huang, Ming Z.	2B/3
Jiahua, Yan	2C/1
Joarder, Kunal	1D/2
Kumar, Arun	1A/3,1B/2
Liljenstam, M.	2A/2
Ling, Shou-Hung W.	2B/3
Marquis, Lawrance	1A/3
Masory, Oren	1A/3,1B/2,1C/1,2C/1,2C/2
Nease, Allen	2A/1
Negahdaripour, Shahriar	2D/1,2D/3
Parton, W.	1D/1
Pigoski, Tom	2C/3
Qu, Zhihua	2B/2
Raviv, Daniel	1D/2,1D/3,2D/4
Rosinski, D.	1D/1
Roth, Zvi S.	1B/3,2B/2
Sandstrom, M.	2A/2
Sari-Sarraf, Hamed	1C/2
Schwartz, Eric M.	1B/1
Shi, Sulan	2D/1
Singh, Ajit	2D/1
Sklar, Michael E.	1A/1
Subramanian, Chenthilvel	1A/3,1C/1
Sudhakar, R.	1D/4
Sumargo, Habibie	2D/2
Thomas, Mark	1A/1
Venkatesh, Kurapati	1C/1
Wallquist, O.	2A/2
Wang, Jian	2C/2
Wang, Luke	1B/3
Wegerif, Daniel G.	1D/1
Xie, Xiang Dong	1D/4
Yakali, Huseyin Hakan	1D/3
Yeralan, Sencer	2A/2
Yu, Chih-Ho	2D/3
Zhuang, Hanqi	1B/3,1D/4,2B/2,2C/2

TABLE OF CONTENTS

Letter from the Chairman	iii
Letter from the President of the University of Florida.....	iv
Conference Committee	v
Program Committee	v
Sponsors.....	v
Author Information	vi
Author Index	ix

1A: ROBOTIC APPLICATIONS (SPACE OPERATIONS)

(Chair: Carl D. Crane III)

Remote Operations Technology	1A/1:1
<i>Michael Sklar & Kent Banks & Mark Thomas</i>	
A Smart Kinestatic Interactive Platform.....	1A/2:1
<i>Michael Griffis & Joseph Duffy</i>	
Design and Construction of a 'Space Emulator'	1A/3:1
<i>Oren Masory & Lawrence Marquis & Chenthilvel Subramanian & Arun Kumar</i>	

1B: ROBOT THEORY 1

(Chair: Keith L. Doty)

The Weighted Generalized-Inverse Applied to Mechanism Controllability	1B/1:1
<i>Eric M. Schwartz & Keith L. Doty</i>	
Optimal Selection of Manufacturing Tolerances for Serial Mechanisms.....	1B/2:1
<i>Oren Masory & Arun Kumar</i>	
Implementation Issues on Simultaneous Calibration of a Robot and a Hand-Mounted Camera	1B/3:1
<i>Hanqi Zhuang & Luke Wang & Zvi S. Roth</i>	

1C: AUTOMATION AND MANUFACTURING 1

(Chair: Zvi S. Roth)

A Sequence Controller Based on Augmented Timed Petri Nets.....	1C/1:1
<i>Kurapati Venkatesh & Chenthilvel Subramanian & Oren Masory</i>	
Issues in Designing a Web Inspection System: Case Study.....	1C/2:1
<i>Dragana Brzakovic & Hal Beck & Hamed Sari-Sarraf</i>	
Object-Oriented Design of Flexible Manufacturing Systems	1C/3:1
<i>E. B. Fernandez & C. P. Han</i>	

TABLE OF CONTENTS (continued)

1D: ROBOT SENSING 1

(Chair: Malek Adjouadi)

Results of Proximity Sensing Research for Real-Time Collision Avoidance of Articulated Robots Working1D/1:1
Near the Space Shuttle

Daniel G. Wegerif & D. Rosinski & W. Parton

Autonomous Obstacle Avoidance using Visual Fixation and Looming1D/2:1

Kunal Joarder & Daniel Raviv

Vision-Based Methods for Autonomous Road Following1D/3:1

Huseyin Hakan Yakali & Daniel Raviv

An Improved Method of Extracting Eye Features using Deformable Template1D/4:1

X. Xie & R. Sudhakar & H. Zhuang

2A: AUTOMATION AND MANUFACTURING 2

(Chair: Michael W. Griffis)

Construction Automation: Navigation of an Autonomous Vehicle2A/1:1

Carl D. Crane & Allen Nease

A Visual Programming Language for Manufacturing Automation.....2A/2:1

S. Yeralan & M. Liljenstam & M. Sandstrom & O. Wallquist & Roy Harrell

Automation For Plant Tissue Culture2A/3:1

R. C. Harrell

2B: ROBOT THEORY 2

(Chair: Eric M. Schwartz)

Calculation of Singularities in Non-Redundant Serial Kinematic Chains.....2B/1:1

Keith L. Doty

Study of the Jacobian of Single-Beam Laser Tracking Systems.....2B/2:1

Hanqi Zhuang & Zvi S. Roth & Zhihua Qu

A Symbolic Program to Formulate and Simplify Jacobians of Robot Manipulators2B/3:1

Shou-Hung W. Ling & Ming Z. Huang

2C: ROBOT THEORY 3

(Chair: Sencer Yeralan)

Spatial Coordinate Measurement Using One Theodolite2C/1:1

Yan Jiahua & Oren Masory

Kinematic Calibration and Compensation of a Stewart Platform2C/2:1

Oren Masory & Jian Wang & Hanqi Zhuang

A Reverse Force Analysis of a Planar Two-Spring System.....2C/3:1

Tom Pigoski & Joseph Duffy

TABLE OF CONTENTS (continued)

Remote Operations Technology

2D: ROBOT SENSING 2

(Chair: Roy Harrell)

Segmenting moving objects and estimating their motion in the presence of camera pan and zoom action2D/1:1
Shahriar Negahdaripour & Sulan Shi & Ajit Singh

A Stereo Matching Technique using Orthogonal Transformations.....2D/2:1
Malek Adjouadi & Frank Candocia & Habibie Sumargo

Computing Optical Flow for Scenes with Time-Varying Brightness.....2D/3:1
Shahriar Negahdaripour & Chih-Ho Yu

Flat Surface: A Visual Invariant.....2D/4:1
Daniel Raviv

Remote Operations Technology

Michael Sklar, Kent Banks, Mark Thomas
McDonnell Douglas Space Systems - Kennedy Space Division

Abstract

Space programs currently in operation and those planned for the near future have an extensive need to operate robotic systems remotely. This capability would allow ground controllers to perform critically needed tasks without the use of on-board astronauts. It also makes advanced exploration possible for various planets that are unlikely to be explored by humans in our lifetime. However, today's state-of-the-art robot control technology and space communications capability is inadequate to perform difficult tasks remotely. A new method of user interaction based on high-level commands and users acting at an intuitive level is required. This paper describes what high-level control is and the technologies required to make it work. Any system employing this technology requires the use of standardized, supervisory-control computer architectures. An overview of control architectures being used or developed by various government, industrial and university laboratories is also presented here. The paper provides an introduction to remote operations technology, high-level user interfaces and some of the component technologies required for this enabling robot control capability.

The Need For Remote Operations

Remote operations involve an interactive user or teleoperator controlling a robotic system while located away from the worksite. The operator may be quite far from the robot, or fairly close to it, but has an obstructed view of the worksite. Many robotic applications, especially those involving military, space and other hazardous operations, require remote operation. Environments which would expose humans to dangerous hazards such as toxic fumes and radioactive material must use robotic devices operated remotely by humans. In many cases it is not feasible to get a human operator where a robot can reach. Planetary operations and deep underwater environments require the use of remotely operated robots in most cases.

The majority of robotic systems in field or production use today are programmed or controlled by teach pendants or

joystick controllers. Teach pendants allow the operator to control either the individual joints of an arm or the motion of the end-effector in each of six Cartesian coordinate directions. This is a very cumbersome method of control, and is typically used only for highly repetitive manufacturing operations. Joystick controllers are used to command the end-effector of a robot to duplicate the motions of an operator's arm.

There are numerous difficulties and problems with these control methods, especially when they're used for remote operations. Tasks that involve fine, precise motions or force interactions are difficult to perform with joysticks. Also, since a joystick device commands a robot arm in real-time, extensive high-speed data communication is required. Furthermore, because the operator's physical motion is directly coupled to the arm motion, a complete view of the worksite and all of its obstacles is required. This is not possible for remote operations unless extensive sensor and video feedback is provided, which places even greater demands on the available data communications.

Advances in remote operations technology are required to perform more difficult, realistic robotic tasks. Examples of these tasks include ground controlled Space Station Freedom Orbital Replacement Unit installation and spacecraft ground operations performed by robotic systems. There are two basic concepts that are now being considered to improve control capability. One of these is telepresence. This method of control is an extension of joystick control since it is based on physical motion of the human operator. This provides the operator with the feeling of actually performing the task at the robot site. Complete visual feedback, which is dependent on the position of the operator's head, is provided. Force and tactile information is applied to the operator's hands. Furthermore, computer graphics are used to place an image of the operator in the visual scene showing the effect of the operator's motions. This allows much more difficult tasks to be performed. However, the large volume of data that must be transferred to and from the operator exceeds present communications capabilities for remote operations. The other alternative method of control is the ability to provide natural language commands to a robot system. This relieves the operator

from the burden of direct physical coupling. This relies on the ability for an intelligent robot system to plan and control its own motions. Creating an operator interface that provides this capability is the first step in developing improved remote operations technology.

What is a High-Level User Interface

Interactive human control of robotic devices can be performed at a number of different levels of abstraction, and can be grouped into three general operating regimes, Reference 1:

1. Teleoperation - The human actively controls the individual maneuvers and actions of the robots;
2. Telerobotics - The robotic system can carry out maneuvers on its own under full-time operator supervision; and
3. Supervised Autonomy - Shared or traded control, in which the operator control and monitoring occurs on a less frequent basis.

Figure 1 shows an example of the various levels of command abstraction for a planetary exploration vehicle and their corresponding operating regimes. There is a corresponding hierarchy of status modeling levels defining the operating state of the robotic system. At the higher levels, commands and status change less frequently and involve less detail, but typically require more abstract reasoning. Providing a user interface at these higher levels reduces the operator burden and fatigue and can improve system safety. For these reasons, and the fact that machines are not yet intelligent enough for fully autonomous operation, a high-level, or telerobotic, interface provides the best regime for control of robotic systems performing unstructured tasks such as space operations, nuclear maintenance, etc. This section discusses some of the features and capabilities of such an interface.

The telerobotic interface interacts with a user at a high level of abstraction, which means that the software must accept high-level commands, decompose these into low-level device control commands, abstract the low-level sensor feedback into composite status measures, and present the status information to the user. In order to

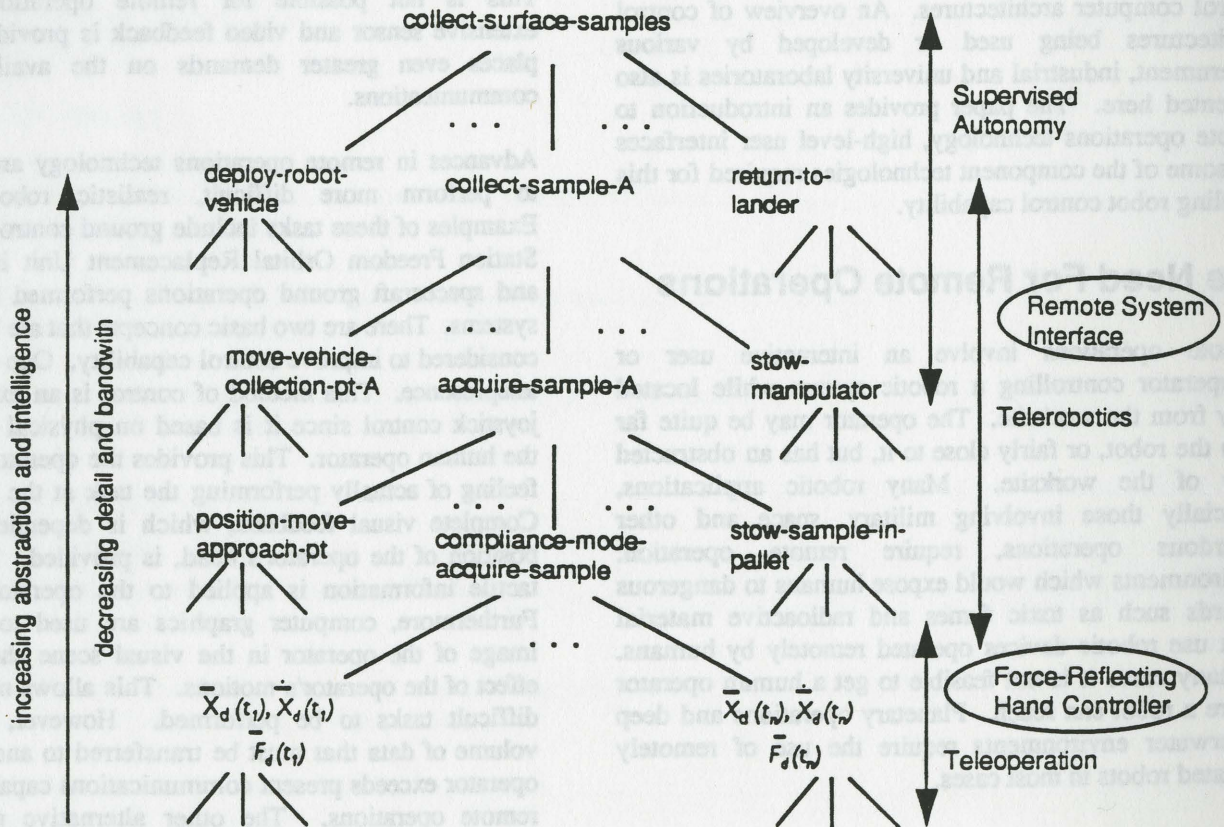


Figure 1: Example Command Decomposition Hierarchy

Table 1. Telerobotic Controller Functions

Command Script Editing
Task Decomposition
Motion Planning
World and State Modeling
Graphic Simulation of Worksite
Task Execution Control
User Site/Robot Site
Communications
Device Interfaces

accomplish this, the controller must have the functionality listed in Table 1. These processing capabilities can be distributed between the User Interface, located at the "user site," and the Robot Controller, at the "robot site." The higher the level of interaction between these sites, the lower the communication bandwidth requirements but the greater the processing required at the robot site. The inter-site communication level is bounded by the level at which the user interacts.

Since a user interacts with a high-level user interface mainly through text and simplified graphics, standard computer platforms with rich graphic interfaces such as engineering workstations and high-end PCs, are the preferred implementation platforms for such an interface. The user enters data through the keyboard and simple position inputs such as a mouse, tracking ball or space ball, and receives feedback mainly through screen displays and possibly simple audio. The high-level interface is less likely to involve complex, special-purpose components, such as exoskeletal systems and force-feedback hand controllers, typically associated with teleoperation. The high-level interface hardware platforms have the benefits of low cost, flexibility, user familiarity, and rapidly increasing computational power.

provides a number of windows or "panes," each of which displays status data from or accepts command inputs for a different activity. The interface is reconfigurable - windows are sized and made visible or active based on the activity being performed, the devices being controlled, the level of control, the state of the system, and user preferences. Figure 2 shows an example of a user interface configured for script editing.

Because of the abstract level of user interaction, the telerobotic interface is a universal controller capable of handling a variety of devices of different types and geometries. For example, the user interface for a robotic system performing space processing tasks would need to provide control for the base platform motion, the manipulator arm or arms, a suite of video cameras, and the various end effectors. The user interface would need to maintain configuration files for the different devices, with information such as the device geometries, number of controllable freedoms, joint motion ranges, etc. These data would be used to bound the control actions, support

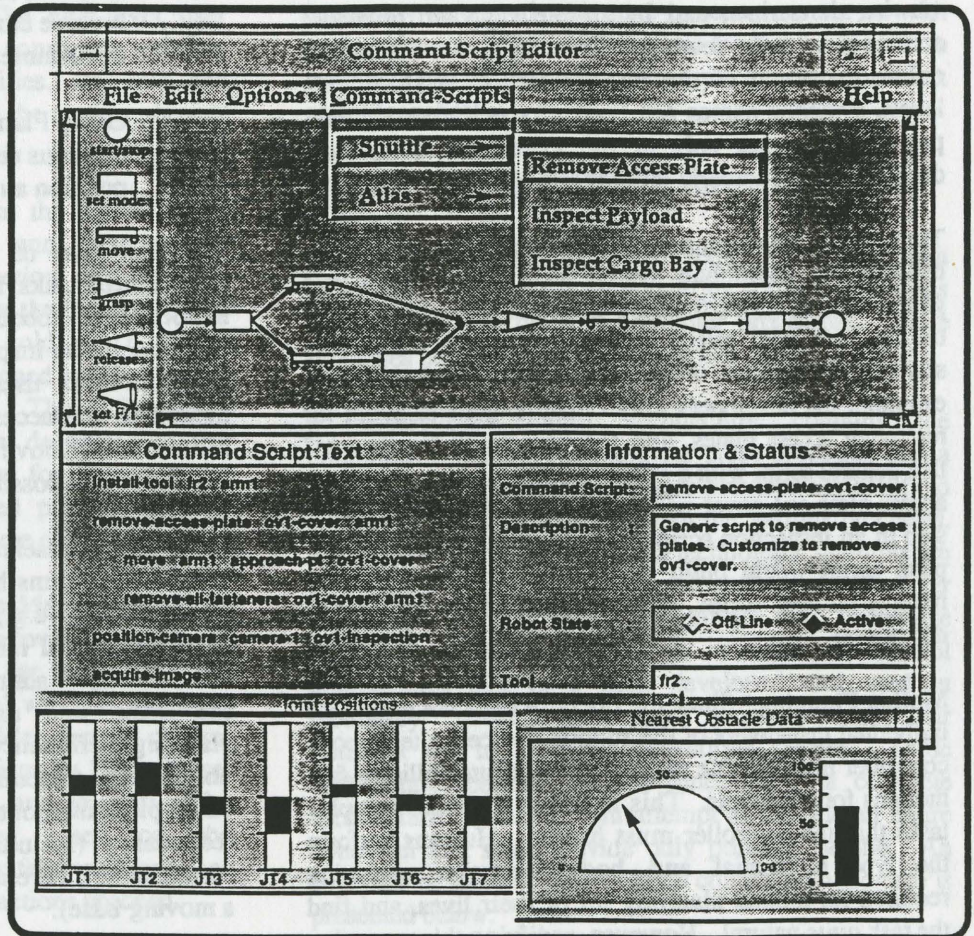


Figure 2: Typical Configuration of a High-Level Interface

The computer display screen for a high-level user interface

motion planning and graphical simulation, and configure the controls and displays of the user interface. The parameters for a device could be downloaded from that device's local controller at startup or connection time.

The remainder of this section discusses some aspects of the telerobotic interface functions listed in Table 1.

Command Script Editing and Task Decomposition - A command script consists of a sequence of actions for the robotic system components. It defines, either explicitly or implicitly, modes in which to perform the steps, terminating conditions for each step, and branch conditions for selecting successive actions. The user interface provides means for viewing and editing scripts, and loading and saving them to file storage. The command scripts are usually represented on the display screen as tabular data or flow diagrams with simple graphic icons. The scripts are organized hierarchically - a task at one level in the hierarchy decomposes into several lower-level steps of more detail (e.g., see Figure 1). The user can select the level of abstraction for viewing the scripts, and by simple keyboard or mouse actions move up or down the hierarchy to see the context or the expansion of the current task. At the higher levels, symbolic names are used to specify destinations, poses and actions; only at the lowest levels do these decompose into numeric joint angles, force levels, etc.

The task of developing command scripts is distributed between the telerobotic system developers who create generic scripts for performing common activities, and the on-line user who selects specific parameters for these scripts and combines them into a higher-level plan. For example, the developers may create a general script for removing access plates, and the user could "instantiate" that general script with a specific instance (e.g., "remove-access-plate ov1-cover with arm1") and include it as one step in an inspection script. The generic remove-access-plate script would build upon other scripts such as a remove-fastener script, also constructed by the system developers.

Motion Planning - Since the operator does not define the individual maneuvers of the robotic devices, a telerobotic controller must be able to plan and execute collision-free motions for the robots. This is one of the more complex tasks that the controller must handle. Humans perform the type of spatial and body-interaction reasoning required for motion planning all of their lives, and find the task quite natural. However, codifying this reasoning into efficient computational models is quite challenging. Motion planning is like object recognition in the sense

that the more natural and intuitive a task is for humans, the tougher it can be to implement it on computers. A high-level user interface could take advantage of human strengths by providing computer-assisted motion planning. For example, the operator might prescribe intermediate path points or select avoidance options, and the computer could test the resulting paths or produce motion alternatives. The interface would graphically display a representation of the robots and work environment, motion paths, and potential collisions. It would provide means for the user to specify robot positions and interactively select motion options.

Motion planning consists of three related planning activities:

1. Path Planning - finding a collision-free path for a robotic device (mobile robot or manipulator arm) through a cluttered space;
2. Fine-Motion Planning - planning robot motions, such as docking or assembly, that involve contact with bodies in the environment; and
3. Grasp Planning - planning motions for a dextrous end-effector to acquire and fine-position an object.

The first two of these are required for almost all unstructured tasks, and grasp planning will become more important as robotic hand technology improves. For instance, in the inspection task example discussed above, the high-level user interface must plan a free-space motion to the access cover, plan compliant motions to contact and remove its fasteners, and plan other motions to remove and possibly stow the access cover.

Much recent research deals with motion planning, and several algorithms have been demonstrated in laboratory environments (e.g., see References 2, 3). However, several practical motion planning issues have yet to be satisfactorily addressed including: world model conversion and use (see the following subsection), planning efficiency, handling model and control uncertainty, tradeoffs between model-based and sensor-based planning/control, and effective use of manipulator redundancy (including multi-arm coordination and the redundancy inherent in having manipulators attached to a moving base).

World Modeling - In order to use symbolic component specifications, model-based motion programming, and

automatic task decomposition, the telerobotic controller must maintain a "world model" that represents the configurations, locations and states of the robots and the objects in their environment. For path planning uses, the world model represents the external shapes of objects and robot links and their expected positions and orientations in some global reference. This geometric model also supports graphic simulation of robotic activities, allowing the user to visualize task performance on a simplified system model. The simulation can be used to support off-line programming of tasks and command scripts, preview execution of a task scenario, or monitor performance during actual system operation. The simulation allows selection of perspective parameters to show the view as it would be seen from robot site cameras. An advanced feature would be the ability to superimpose actual camera images on the simulation display, thereby supporting model registration and calibration.

The geometric model should be hierarchically organized to promote efficiency and allow planning and viewing at appropriate levels of resolution. The high-level user interface should include a means for converting existing CAD models of components, facilities, etc. into the special-purpose solids models used by the motion planner and graphic simulation.

In addition to geometric information, the world model must also maintain connectivity and functionality information. Considering the inspection task example again, the world model must represent the location of the access cover, know what fasteners hold this cover in place, where the fasteners are positioned relative to the cover, and how they are removed. This information would be accessed and used when decomposing the general part removal script into steps for removing the individual fasteners. Object-oriented programming is ideally suited for implementing this type of functionality.

Task Execution Control and Status Monitoring - The high-level user interface provides windows for control of the real-time system operation and for monitoring the system status. The execution control window is always available during operation, and provides general control actions such as Execute Script, Emergency Halt, Pause, Continue, Execute Immediate Move, and so forth. The immediate commands are limited based on the experience level of the user, and safety tests such as clearance checking are performed as actions proceed.

Feedback from the robot sensors is used to determine system status during operation. A hierarchy of status

representation is maintained by the telerobotic controller. For example, joint resolver or encoder signals at the lowest levels are abstracted into joint position information, Cartesian end-effector locations, and symbolic configuration information. Camera images could be processed to extract visual cues, such as points and lines, identify objects from these cues, and register the objects. Strain gage measurements could be converted into a set of forces and torques at a wrist sensor, and then these could be processed by subtracting gravity effects and adjusting the coordinate reference to determine contact forces on a part being assembled.

Windows are available in the user interface for displaying the status information, usually tabular data or simple graphics such as bar charts or gauges. For example, one set of gauges may display joint angles relative to their bounds, while another displays the force-torque state of the end-effector during a compliance operation.

During system operation, the status information is constantly compared with state expectations set up with the command script. The state feedback is used to terminate processing steps, enter error handlers, or select script branches.

Supervisory Control Architectures

Any advanced telerobotic system requiring direct human interaction is controlled by numerous software processes executing on one or more computer processors. These processes perform the functions shown above in Table 1.

A supervisory control architecture contains the components necessary to integrate these processes into a supervisory control system. It also provides the ability to control and communicate between the various processes. In developing remote operations technology, attempts are being made to use standard architectures and computer platforms to maximize portability and minimize development time. Thus, a number of control architectures that have been developed and used in laboratory test-beds have been examined for their potential use in a remote operations system. These architectures perform various functions of a complete architecture. Some of them attempt to provide an entire controller and some provide only portions of one. An overview of some of the available control architectures is presented below.

RPI - Intelligent Machine Architecture

Recent research at the RPI Center for Intelligent Robotic Systems for Space Exploration (CIRSSE) has concentrated on the *Theory of Intelligent Machines*. As described in Reference 4, the intelligent machine (IM) encapsulates autonomous robot functionality in a tri-level hierarchy comprised of the Execution, Coordination, and Organization levels.

The Execution Level contains physical devices and provides their associated basic control services. The Coordination Level is composed of a dispatcher and multiple coordinators. The dispatcher receives and, as needed, decomposes commands from the Organization Level and communicates them to the appropriate coordinators. The coordinators select the most efficient methods of accomplishing tasks in real time and communicates them to the Execution Level for implementation. The Coordination Level also provides communications for the IM, attempts to resolve error messages received from the Execution Level, and schedules the use of system resources. The Organization Level is responsible for performing intelligent planning and reasoning at an abstract level and for responding to errors that the Coordination Level was unable to resolve.

Together, the components of the IM comprise an architecture that represents the principle of *Increasing Precision With Decreasing Intelligence*, Reference 4. CIRSSE researchers have implemented the Execution and Coordination levels of the IM. The result of their efforts are the CIRSSE Test-bed Operating System (CTOS) and the Motion Control System (MCS) shown in Figure 3.

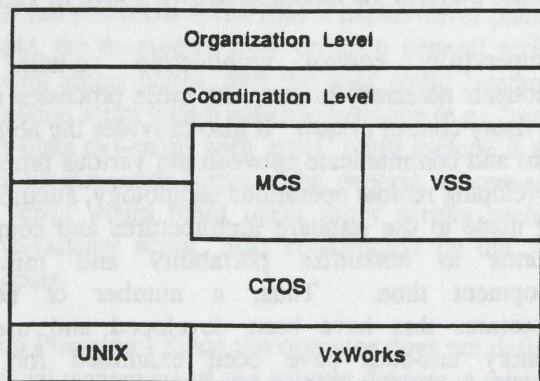


Figure 3: Components of the IM. Adapted from Reference 4.

CIRSSE Test-bed Operating System (CTOS) - CTOS was developed to provide a consistent programming interface to the hardware in the CIRSSE test-bed. More generally, CTOS provides a set of system services that facilitate the development of distributed telerobotic

applications. System-wide communication, task distribution, and task synchronization are available through the services that constitute CTOS.

The Bootstrap Service reads configuration files that specify desired task distributions, loads and initializes programs on the requested processors, and synchronizes the processes for system startup. The Task Identification Service associates a unique identifier with the symbolic name of every process run under CTOS. These identifiers serve as the "address" of processes in the distributed environment. The Message Passing Service uses these addresses to provide system-wide communication for CTOS. It dispatches messages to processes regardless of their location by obtaining their address through associated symbolic names. Low-level communications are supported by the final two services. The Synchronization and Inter-processor Blocking Services facilitate real-time execution performance through high-speed communication.

CTOS is an event-driven application environment, i.e., messages are passed and event-handlers are fired in response to events that occur in the system. As such, application developers create CTOS control applications by defining collections of event-handler functions. CIRSSE periodically offers training in the use of CTOS and MCS.

Motion Control System (MCS) - The CIRSSE Motion Control System is a modular real-time control system that provides control of multiple manipulators. MCS is implemented as an open architecture in a distributed environment and supports a variety of computing platforms. MCS makes up the lowest level of the IM; it maintains detailed state information on the physical

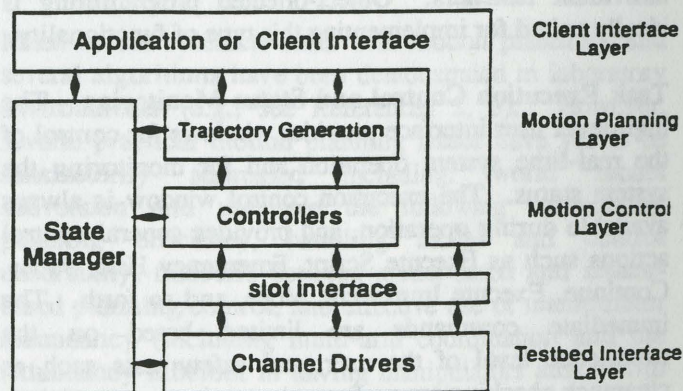


Figure 4: MCS Architecture. Adapted from Reference 4.

devices in the test-bed and passes only command status to higher levels of the IM. As seen in Figures 3 and 4, MCS is layered on top of CTOS in four modular layers with well defined interfaces.

The most abstract layer of MCS is the Client Interface Layer. As its name implies, the client interface serves as a consistent entry point for MCS. It is responsible for receiving commands from higher levels of the IM and passing them on to the Motion Planning Layer. The motion planner generates trajectory information, which the Motion Control Layer uses with feedback from the device sensor to determine joint torques values. These values are sent to the foundation of MCS, the Test-bed Interface Layer. The test-bed interface contains device drivers that convert the digital commands into analog signals and control the system actuators.

One additional component spans the four layers of MCS to oversee its operation. This component, the State Manager, is responsible for monitoring system configuration and for coordinating the functionality of the other layers. It also performs resource allocation and conflict resolution and maintains the state of the system.

MCS and CTOS were developed in the C programming language. They run in a VME-based multi-processor environment and on UNIX workstations, respectively. These components of the IM have been used at CIRSSSE to control an 18 DOF robotic system.

JPL - Modular Telerobotic Task Execution System

A telerobotic system architecture designed and implemented at the NASA Jet Propulsion Laboratory Supervisory Telerobotics (STELER) laboratory focuses on telerobotic control capabilities required at the robot site. The Modular Telerobot Task Execution System (MOTES), described fully in Reference 5, attempts to maximize robot control capabilities while operating within limited computational constraints that may be characteristic of remote robot sites. Beyond baseline teleoperation, MOTES accommodates shared control and supervised autonomous control of a redundant manipulator. The MOTES architecture, seen in Figure 5, is composed of a number of modules, which in many ways mirror the Primitive level of the NASREM telerobot control architecture. NASREM, described in Reference 6, is a model for hierarchical control of intelligent machines which was an attempt to standardize the control of these machines. It was developed jointly in the mid 80's by NASA and NBS (now called NIST).

The Shared Memory Module serves as a coordinator of global memory. It provides a communication capability between other modules, maintains systems status information, and contains queues of commands to be executed by specific modules. A task queue contains commands received from the user site. A reflex queue contains commands that should be executed in response to the occurrence of events.

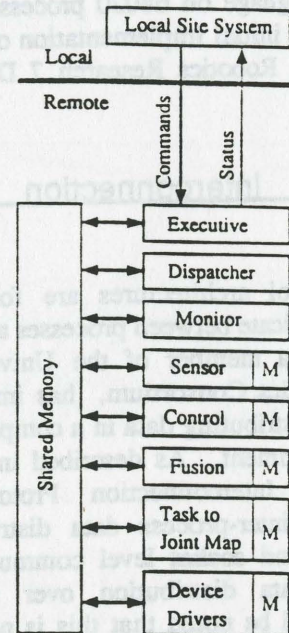


Figure 5: Motes Architecture.
Adapted from Reference 5.

The Executive Module is the liaison between the robot site and the user site. It receives commands from the user site and places them in the task queue in shared memory. The Executive also returns status information and requested data to the user site.

The Dispatcher Module acts as an event manager, scheduling reflex commands for execution in response to occurring events. It also schedules task commands according to the state of the system and the issue time of the commands. The Monitor Modules monitor the system status for both normal and abnormal conditions, and the Sensor Modules collect and provide both real and virtual sensor data.

The remaining modules combine to effect robot motion and therefore may be thought of collectively as an "execution module." The Control Modules generate motion control commands for the real and virtual sensors. These control commands are mapped to task

space and are placed in shared memory where the Fusion Module reads and aggregates them into task space commands. The task space commands are translated into actuator space commands for the physical devices by the Task to Joint Map Module. Finally, the Device Drivers Module communicates the actuator commands to the target robotic devices and collects status data.

JPL has implemented this architecture in the Ada programming language on 68020 processors in a VME environment. The initial implementation of MOTES was used to control a Robotics Research 7 DOF redundant manipulator.

Telerobotic Interconnection Protocol (TelRIP)

Telerobotic control architectures are founded on an ability to communicate between processes and processors. Rice University, a member of the Universities Space Automation/Robotics Consortium, has implemented an architecture for distributing data in a complex distributed computing environment. As described in Reference 7, the Telerobotic Interconnection Protocol provides mechanisms for inter-process data distribution using shared memory and socket level communications and inter-processor data distribution over networks via TCP/IP. It should be noted that this is not a complete robot control architecture, but simply a software tool that provides the basis for creating a control architecture. TelRIP includes a communication protocol that describes the format of data packets, an event-driven router process that sends and receives data packets, and a programming interface that allows users to create TelRIP applications.

The TelRIP communication protocol specifies a consistent format for data packets. It allows for the identification of numerous characteristics of each packet, such as data source, data size, and message type. This data packet format is enforced by the router processes and the programming interface.

TelRIP uses a router process to distribute data. Each processor in a distributed environment runs one router. The routers are configured in a fully connected network, i.e., each router is connected to every other router to allow data to be distributed among all processes. Each process that needs to receive data "registers an interest" with its local router. This "interest" is actually a callback function to be executed when data of a certain type is available, i.e., when an event occurs. A process sends data by telling its local router to distribute it. The router

sends the data to interested processes on the same processor and broadcasts the data to the other routers in the network so they may do the same. This data driven approach allows distributed applications to be easily reconfigured because processes may be spread among processors with little or no impact to other system components.

The final component of TelRIP, the application programming interface (API), provides a consistent library of C functions (TelRIP is implemented in the C programming language) for creating distributed applications. The library contains functions that allow users to register processes with a router, to manipulate data packets, and to distribute data. There are also functions that allow a process to send and receive short messages, to spawn processes on other processors, to register timer events (functions that execute after a specified amount of time passes), and to handle file I/O.

TelRIP runs on a number of platforms including Sun and Silicon Graphics workstations, MS-DOS PCs in a client-server relationship, and under the Lynx operating system. The available platform options, high-level data distribution mechanisms, consistent API, and straightforward user's guide offered by TelRIP provide users with the core capabilities required for supervisory control system development.

Conclusions

A number of current and planned robotics applications, ranging from nuclear maintenance, ground-based and on-orbit space operations, to interplanetary exploration, require human control from remote sites. A telerobotic controller with a high-level user interface is the best way to perform many of these operations. Such an interface allows the user to work at a more natural, intuitive level, specifying abstract commands using symbolic identifiers and viewing composite status measures. In order to support this mode of operation, the telerobotic controller must provide advanced features such as command script editing, autonomous task decomposition, motion planning, world modeling and status monitoring. Researchers at several laboratories have developed architectures based on standardized computer platforms to support implementation of such controllers. These architectures provide common services, such as processing task allocation and scheduling, and inter-process communication and control. McDonnell Douglas Space Systems researchers are integrating and advancing these technologies in the development of a universal

remote system interface that will be capable of controlling a variety of robotic devices performing a range of space-related processing operations.

References

- [1] Erickson, Jon D., Paschal J. Aucoin, Jr. and Peter G. Ossorio, "Person-like intelligent systems architectures for robotic shared control and automated operations," *Proceedings Cooperative Intelligent Robotics in Space III*, Boston, MA, November, 1992, pp. 149-160.
- [2] Latombe, Jean-Claude, *Robot Motion Planning*, Kluwer Publishing, Norwell, MA, 1991.
- [3] Montana, David J., "The Kinematics of Contact and Grasp," *International Journal of Robotics Research*, Vol. 7, No. 3, (June, 1988), pp. 17-32.
- [4] Lefebvre, D. R. and G. N. Saridis, "A Computer Architecture for Intelligent Machines," CIRSSE Report 108, Rensselaer Polytechnic Institute, Troy, NY, December 1991.
- [5] Backes, Paul, Mark K. Long, and Robert D. Steele, "Designing Minimal Space Telerobotics Systems For Maximum Performance," *Proceedings of the AIAA Aerospace Design Conference*, Irvine, CA, February 3-6 1992.
- [6] Albus, J., H. McCain and R. Lumia, "NASA/NBS Standard Reference Model For Telerobot Control System Architecture (NASREM)," National Bureau of Standards Robot Systems Division report, December 4, 1986.
- [7] Wise, J. D. and Larry Ciscon, "TelRIP Distributed Applications Environment Operating Manual," Technical Report 9103, Rice University, Houston, TX, 1992.

A SMART KINESTATIC INTERACTIVE PLATFORM

M. Griffis and J. Duffy

Center for Intelligent Machines and Robotics
University of Florida
Gainesville, Florida 32611

ABSTRACT

This paper presents a newly patented in-parallel platform which is geometrically simple. It has six distinct connecting points in both the base and top platforms. The forward solution can be determined by solving an eighth degree polynomial.

INTRODUCTION

Over the past few years, there has been an ever increasing interest in direct applications of parallel mechanisms to real-world industrial problems. In situations where the needs for accuracy and sturdiness dominate the requirement or a large workspace, parallel mechanisms present themselves as viable alternatives to their serial counterparts. This article is confined to the forward displacement analysis of Stewart Platform-type parallel mechanisms. In the general sense, each of these mechanisms consists of two platforms that are connected by six prismatic joints acting in-parallel. One of the platforms is defined as the "top platform." It has six degrees of freedom relative to the other platform, which is the "base." It is then required to compute all possible locations (positions and orientations) of the top platform measured relative to the base for arbitrary sets of six connecting prismatic leg lengths. Each prismatic pair is connected at each end to the base and top platforms by ball and socket joints.

Stewart¹ introduced his platform in 1965 as an aircraft simulator. Since then, many parallel mechanisms containing prismatic joints have been called Stewart Platforms, although Stewart originally suggested only two different arrangements. Hunt^{2,3}, Mohamed and Duffy⁴, Fichter⁵, Sugimoto^{6,7}, Rees-

Jones⁸, and Kerr⁹ all suggest use of Stewart Platforms, with various applications ranging from manipulators to force/torque sensors. Reinholz and Gokhale¹⁰ (as well as Miura, Furuya, and Suzuki¹¹, and Miura and Furuya¹²) investigated an interesting application in the form of "Variable Geometry Truss Robots (VGTs)." It is apparent that NASA's octahedral VGT is in fact founded on the simplest of the Stewart Platforms.

Much of the research in the literature has devoted extensive effort to the reverse displacement analysis that is inherently simple for parallel mechanisms (viz. it is required to compute a set of leg lengths given a desired location of the top platform relative to the base). Few, however, have investigated closed-form forward displacement analyses for parallel mechanisms. Instead, they depend on purely numerical solutions. Behi¹³ investigated a forward displacement analysis of a parallel mechanism that closely resembles a Stewart Platform. Numerically, he was able to find eight solutions. Reinholz and Gokhale¹⁰ used the Newton-Raphson technique to obtain an iterative solution for the forward displacement analysis of a Stewart Platform.

A closed-form forward displacement analysis (as opposed to an iterative one) will yield much important information on the geometry and kinematics of a parallel mechanism. For instance, a closed-form solution for a Stewart Platform will not only yield the exact number of real configurations of the top platform relative to the base for a specified set of leg lengths, but also quantify the effects of errors in leg lengths on the position and orientation of the top platform. Furthermore, a forward displacement analysis of a Stewart Platform manipulator will provide a cartesian controller with necessary feedback information, namely the position

and orientation of the top platform relative to the base. This is especially important when the actual position and orientation cannot be directly sensed, and when the manipulator's configuration is determined solely from lengths of the connecting prismatic legs. Near singularities, purely numerical solutions may experience difficulties, since they provide no way to determine changes in closure.

FORWARD ANALYSIS

The forward analysis of in-parallel platforms has attracted much attention in academia in recent years. The most general in-parallel platform is called a 6-6 platform because there are six distinct connecting points in the base and six distinct connecting points in the top platform (see Figure 1). In this paper all connecting points are considered to be coplanar in both the base and top platforms. The closed-form analysis for the 6-6 platform has proven to be a most challenging task.

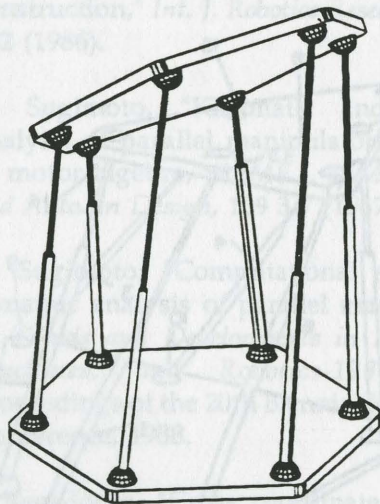


Figure 1: General 6-6 Platform

The most simplified form of the mechanism contains six legs which meet in a pair-wise fashion at three points in both the top platform and base (see Figure 2). This form of mechanism which was called the "3-3 Stewart Platform" was solved by Griffis and Duffy¹⁴. Their solution was easily extended to a "6-3 Platform" whose legs meet at six distinct points in a planar base (see Figure 3). Nanua, Waldron, and Murthy¹⁵ also obtained a closed-form solution for the 6-3 Platform. A 6-3 Platform with a non-planar base

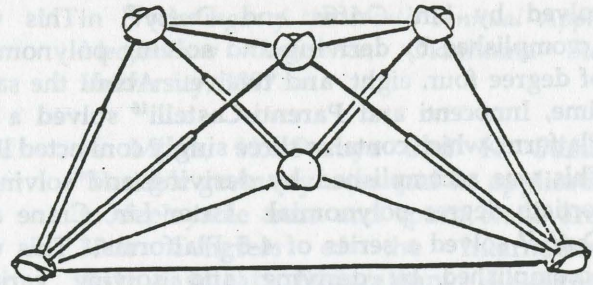


Figure 2: 3-3 Platform

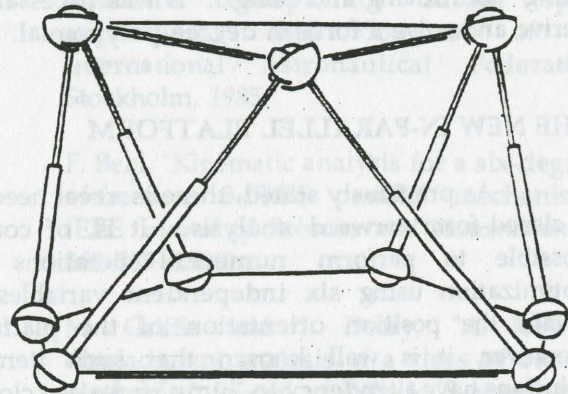


Figure 3: 6-3 Platform

was analyzed by Innocenti and Parenti-Castelli¹⁶. The maximum number of assembly configurations for the 3-3 and 6-3 Platforms are sixteen. For the 3-3 and for the 6-3 Platform with six distinct connecting points lying in a plane it is necessary to solve an eighth degree in a variable x^2 . This yields a maximum of 8 real assembly configurations above the base and 8 reflected assembly configurations through the base. For the more general 6-3 Platform with non-planar connecting points it is necessary to solve a general 16th degree polynomial.

It is clear that the 3-3 Platform, an octahedron, has the simplest forward solution and it further became clear that apart from the 6-3 Platform, as more distinct connecting points were introduced the degree of complexity of the forward analysis increased. The most complex forward analysis is that for the 6-6 Platform. In order to gain insight into the problem formulation for the general 6-6 Platform, the forward analysis for a sequence of platforms was performed.

A number of 4-4 Platforms, for which the six legs connected either singly or pair-wise at four points in both the top and base platforms were

solved by Lin, Griffis and Duffy¹⁷. This was accomplished by deriving and solving polynomials of degree four, eight, and twelve. About the same time, Innocenti and Parenti-Castelli¹⁸ solved a 5-5 Platform, which contains three singly connected legs. This was accomplished by deriving and solving a fortieth degree polynomial. Later Lin, Crane and Duffy¹⁹ solved a series of 4-5 Platforms. This was accomplished by deriving and solving various polynomials ranging from sixteenth degree to thirty second degree. Most recently, the solution to the general 6-6 Platform was obtained by Wen and Liang²⁰ and Zhang and Song²¹. It was necessary to derive and solve a fortieth degree polynomial.

THE NEW IN-PARALLEL PLATFORM

As previously stated, there is a real need for a closed-form forward analysis. It is, of course, possible to perform numerical iterations (an optimization using six independent variables) to obtain the position orientation of the platform. However, it is well known that such iterative solutions have a tendency to "jump" from one closure to another. From a practical viewpoint, this is undesirable.

The necessity for a closed-form analysis really manifests itself whenever a platform is to be used to control force and motion simultaneously. Any in-parallel structure lends itself well to a static force analysis particularly when utilizing the theory of screws^{8,9}. The wrench applied to the top platform can be statically equated to the summation of the forces measured along the lines of the six prismatic legs. Thus far this particular application of an in-parallel platform has depended on relatively "small" leg deflections, resulting in a "constant" configuration. However, employing a closed form forward analysis provides the analytics to monitor gross deflections of the platform and thereby permit one to consider the design of a compliant force/torque sensor.

There is, however, a dilemma. The geometrically simplest parallel mechanism is clearly the octrahedron which contains six legs that meet in a pair-wise fashion at three points in both the top platform and base. The closed form forward analysis involves the solution of an eighth degree polynomial. It is practical to solve such a polynomial rapidly and in real time. This is necessary when employing the platform to control force and motion simultaneously. However, such a platform has a serious mechanical

disadvantage. It is not possible to design the necessary concentric ball and socket joints at each of the double connection points without mechanical interference. Such a design is mechanically unacceptable. It is preferable to separate the double connection points in order to overcome the mechanical design problem. However, the closed-form forward solution for the general 6-6 Platform involves the solution of a fortieth degree polynomial. It is not practical to solve a fortieth degree polynomial in real time.

The newly patented platform, illustrated in Figure 4 incorporates the benefits of both the octrahedron Platform and the general 6-6 Platform. There are six distinct connecting points in both the top platform and the base which avoids the mechanical interference problem. At the same time it is possible to control the position of the top platform employing the eighth degree polynomial for the forward solution of the octrahedron. This combination makes possible the simultaneous control of force and motion using the newly patented device.

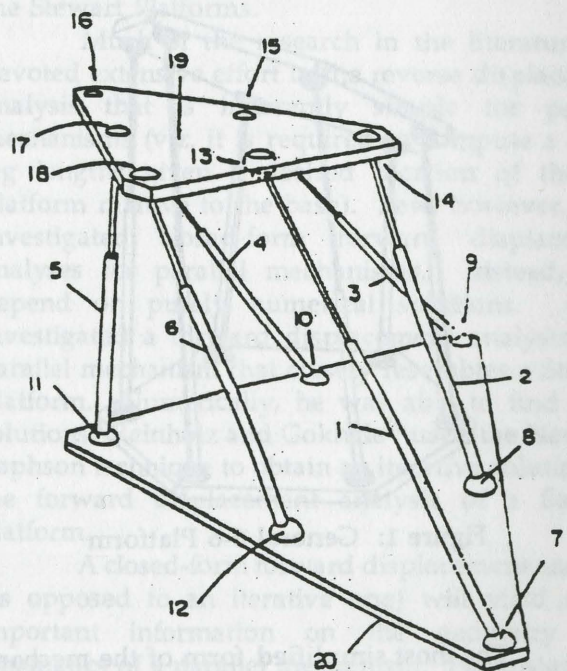


Figure 4: Special 6-6 Platform
(U.S. Patent # 5,179,525)

REFERENCES

1. D. Stewart, "A platform with six degrees of freedom," in *Proc. Inst. Eng., London*, Volume 180, 1965, pp. 371-386.
2. K.H. Hunt, *Kinematic Geometry of Mechanisms*, Oxford University Press, London, 1978.
3. K.H. Hunt, "Structural kinematics of in-parallel-actuated robot-arms," *Trans. ASME, J. of Mech., Trans., and Auto. in Design*, 105, 705-712 (1983).
4. M.G. Mohammed and J. Duffy, "A direct determination of the instantaneous kinematics of fully parallel robotic manipulators," *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 107, 226-229 (1985).
5. E.F. Fichter, "A Stewart platform-based manipulator: General theory and practical construction," *Int. J. Robotics Research*, 5, 157-182 (1986).
6. K. Sugimoto, "Kinematic and dynamic analysis of parallel manipulators by means of motor algebra," *ASME J. of Mech. Trans. and Auto. in Design*, 109 3-7 (1987).
7. K. Sugimoto, "Computational scheme for dynamic analysis of parallel manipulators," in *Trends and Developments in Mechanisms, Machines, and Robotics-1988*, ASME Proceedings of the 20th Biennial Mechanisms Conference, 1988.
8. J. Rees-Jones, "Cross coordinate control of robotic manipulators," in *Proceedings of the International Workshop on Nuclear Robotic Technologies and Applications, Present and Future*, University of Lancaster, UK, June 29-July 1, 1987.
9. D.R. Kerr, "Analysis, properties, and design of a Stewart-platform transducer," in *Trends and Developments in Mechanisms, Machines, and Robotics-1988*, ASME Proceedings of the 20th Biennial Mechanisms Conference, 1988.
10. C. Reinholz and D. Gokhale, "Design and analysis of variable geometry truss robots," in *Proceedings of the Ninth Annual Applied Mechanisms Conference*, Oklahoma State University, 1987.
11. K. Miura, H. Furuya and K. Suzuki, "Variable geometry truss and its application to deployable truss and space crane arm," 35th Congress of the International Astronautical Federation, Lausanne, Switzerland, Pergamon, New York, 1984.
12. K. Miura and H. Furuya, "An adaptive structure concept for future space applications," 36th Congress of the International Astronautical Federation, Stockholm, 1985.
13. F. Behi, "Kinematic analysis for a six-degree-of-freedom 3-PRPS parallel mechanism," *IEEE Journal of Robotics and Automation*, 4, 561-565 (1988).
14. M. Griffis and J. Duffy, "A forward displacement analysis of a class of Stewart platforms," *Journal of Robotic Systems*, John Wiley, 6(6), 703-720 (1989).
15. P. Nanua, K.J. Waldron and V. Murthy, "Direct kinematic solution of a Stewart platform," *IEEE Trans. on Robotics and Automation*, 6(4), 438-444 (1990).
16. C. Innocenti and V. Parenti-Castelli, "Direct position analysis of the Stewart platform mechanism," *Mechanism and Machine Theory*, Vol. 25, No. 6, 611-621 (1989).
17. W. Lin, M. Griffis and J. Duffy, "Forward displacement analyses of the 4-4 Stewart platforms," Proceedings of the 21st ASME Mechanisms Conference, Chicago, IL, Sept. 16-19 (1990).
18. C. Innocenti and V. Parenti-Castelli, "Closed-form direct position analysis of a 5-5 parallel mechanism," Dipartimento di Ingegneria delle costruzioni meccaniche, nucleair, aeronautiche e di Metallurgia - Universita de Bologna, Italy (1990). (Private Communication).
19. W. Lin, C.D. Crane and J. Duffy, "Closed-form forward displacement analyses of the 4-

5 in-parallel platforms," Proceedings of the 22nd ASME Mechanisms Conference, Phoenix, AR, Sept. (1992).

20. F.A. Wen and C.G. Liang, "Displacement Analysis for the General Stewart Platform-Type Mechanism," Beijing University of Posts and Telecommunications, private communication, June 1992.
21. Change-de Zhang and Shin-Min Song, "Forward Position Analysis of Nearly General Stewart Platforms," Proceedings of the 22nd Biannual ASME Mechanisms Conference, Scottsdale, Arizona, Sep 1992.

DESIGN AND CONSTRUCTION OF A "SPACE EMULATOR"

Masory O., Marquis L., Subramanian C. and Kumar A.

Robotics Center
Florida Atlantic University
Boca Raton, FL 33431

Abstract

A "Space Emulator" by which the motions of a manipulator carried by a small spacecraft can be emulated in lab environment is described. It consists of two Stewart platforms whose velocity and acceleration are controlled by Admittance Controller that acquires the interaction forces between the manipulator's base and the space vehicle and between the manipulator and the task from two six degree-of-freedom force/torque sensors, mounted on the platforms. In return, the controller determines the required velocity for each platform so that a free-float motion is emulated. The required platform velocity is fed to platform controller which resolves it to link velocities, using the inverse kinematic solution of the platform, and feed the results to control modules which regulate the velocity of each link in a closed-loop fashion. The descriptions of the mechanical structure, the force/torque sensors, the controller and the control/interface software are presented.

I. INTRODUCTION

As humanity branches out beyond the confines of our planet, robots will play an increasingly important role. The goal of robots in space will be to replace humans in a dangerous working environment with machines that can work harder longer and without the extreme safety precautions or expense [1]. Thus, robots will be an extremely important

element in mans establishing an extensive and long term presence in space.

Control of manipulators in space environment is difficult since the vehicle to which the manipulator is attached to is free to float. As a result, motions of the robot and task interaction forces would cause a disturbance in the velocity/position of the vehicle and consequently of the robot itself. Currently, robots carried by satellites are remotely controlled by ground or space based human operators which is difficult and expensive. Therefore, control techniques, which will make it easier for human operators to guide the manipulators through demanding tasks in space, need to be developed.

In order to develop applications and control schemes to overcome these problems, it is necessary to emulate the free-float behavior in a laboratory environment. Space emulators, as described in this paper, can be used to develop and test new control strategies, path planning algorithms and applications [2]. They could be also used to train operators, here on earth, with greater safety and less expense than space based training.

Such a space emulators is currently under construction at Florida Atlantic University Robotics Center and the purpose of this paper is to describe this particular system.

II. SYSTEM DESCRIPTION

The space emulator consists of two Stewart platforms [3,4], shown in Figure 1, where one is used to emulate the motions of the spacecraft carrying the robot and the other the task

dynamics. A robot is placed on top of one of the platforms and the application hardware upon which the task is performed is placed on top of the other. Six degree-of-freedom force-torque sensors are mounted between the manipulator and its platform and between the device and its platform. These sensors measure the forces generated by the manipulator motions and the interaction forces between the robot and the task. These forces in conjunction with the dynamic model of the spacecraft and the task are used by Admittance Control to maneuver the platforms such that space conditions (zero gravity) are emulated.

A Stewart Platform, illustrated in Figure 2, is composed of six variable length links (prismatic joints), a fixed base and a movable plate. Each link is attached to the plate by a ball joint and to the base by a U-joint. Since the link is realized by hydraulic piston whose rod is free to rotate about its axis and thus providing additional rotational axis, each link is actually connected at both end by U-joints. These particular pistons have 1.5" bore, 30" stroke and their minimum length is 54".

The pose of the platform is defined as the relative location of {P} with respect to {B} where {P} is located at the center of the moving plate, O_p , and {B} at the center of the base O_b . The pose and the velocity of {P} are determined by the length and velocity of each link respectively. Electro-hydraulic servo valve is used to regulate the flowrate of oil into the piston and thus determine its velocity. The flowrate through the electro-hydraulic servo valve is directly proportional to the electric current that flows through the valves inductors. Amplifiers are used to convert the voltage signal produced by a motion controller and a dither signal into the current required to drive the valves. A 50HP (1500psi 50gpm) hydraulic pump is used to power each platform and the maximum flowrate through the valves at this pressure extend the piston at a maximum velocity of 15in/sec. A rotary optical encoder, attached to the piston through a rack and pinion arrangement, provides the necessary

position and velocity feedback signals, that are used by a closed-loop controller as will be explained later.

The base and the plate are circular with radii $R_b=3ft$ and $R_p=1ft$ respectively. The U-joints attached to the plate and the base are at P_1 to P_6 and B_1 to B_6 respectively. The coordinates of B_i , $i=1...6$ with respect to {B} denoted as b_i , and those of P_i with respect to {P} as p_i . The X axis of {B} is selected along the line which bisects the angle $B_1O_bB_6$, and similarly for X axis of {P}. The location of joints on the base and the plate are at:

$$b_i = R_b (\cos \alpha_{b_i}, \sin \alpha_{b_i}, 0)$$

$$\alpha_{b_i} = 30^0, 90^0, 150^0, 210^0, 270^0, 330^0 \quad (1)$$

$$p_i = R_p (\cos \alpha_{p_i}, \sin \alpha_{p_i}, 0)$$

$$\alpha_{p_i} = 15^0, 105^0, 135^0, 225^0, 255^0, 345^0$$

The plate pose can be described a 3*3 orientation matrix R and a translation vector q which define {P} with respect to {B}. The corresponding links lengths are given by the norms of the vectors connecting B_i to P_i expressed in {B}:

$$L_i = |R p_i + q - b_i| \quad i=1, \dots, 6 \quad (2)$$

This relationship is used by the controller to determine the length and velocity of each link necessary to drive the platform to the required pose at a given trajectory velocity.

The force sensors use straingages as transducers and their signals are amplified by instrumentation amplifiers before being fed to the admittance controller through a data acquisition board. The admittance controller is implemented on a 33MHz 486DX IBM/AT computer which communicate the results (velocity references for both platforms) to the platform controllers through RS-232 serial ports.

The platform controller is implemented on a 25MHz 386/7 IBM/AT computer in which a motion control board was installed. Six modules, each implements a PID controller, are installed on this board and thus control all six links of the platform can be controlled simultaneously. At each sampling period each module is fed by the feedback signal produced by the encoder and with a velocity reference (communicated through the IBM/AT bus) and in return it produces a new reference for the servo valves.

The required velocities of the links are determined by the platform controller using the inverse kinematic solution of the platform. However, before applying these results the controller checks out whether any of the following kinematic constraints is being violated: 1) Link length limitations; 2) Joint angle limitations; and 3) Links interference [9]. If no constraint is being violated, the results are communicated to the control modules.

III. THE FORCE/TORQUE SENSOR

Most force sensors include structurally flexible members in which high stresses are developed due to the load. The strain on these members are measured by transducers which their readings are related back to the load [5-7]. Such sensors can be made of simple, easy to fabricate structure and instrumented with low cost transducers. With this concept in mind, a sensor which constructed as two rings connected by three flexible members, have been designed (figure 3). In order to select dimensions of the flexible members (b - width and h - length of the element) the sensor was modeled using a commercial FEM code and the following characteristics were determined as function of these dimensions (see Figure 4):

1. Maximum stresses under maximum anticipated load in order to avoid structure failure.
2. Maximum strains in order to avoid straining failure.

3. Sensor stiffness matrix condition number in order to estimate to what degree the six measured quantity are separately.
4. Sensor sensitivity given by the norm of the vector F calculated under the assumption that the minimum possible reading from each bridge is one microstrain.
5. Sensor natural frequency in order to avoid dynamic excitation of the sensor.

Once the sensor was fabricated its stiffness matrix was determined by a calibration procedure the sensor was loaded with a uniaxial force (F_x , F_y or F_z), which its magnitude was measured by uniaxial load cell, and the corresponding reading from the straingages were recorded. Since it was impossible to apply a pure moment, a combination of a uniaxial force and moment was used. However, since the sensor readings due to the force are already known, it is possible to determine its readings due to a pure moment.

An example of the calibration results are shown in figure 5 which illustrates the output from the six bridges during loading and unloading the sensor with a force in X direction. Linear relationships between the load and the outputs were established by fitting a straight line to the data. The slopes of these lines are the elements of the first column of the compliance matrix, C :

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \end{bmatrix} = \begin{bmatrix} C_{11} & 0 & 0 & 0 & 0 & 0 \\ C_{12} & 0 & 0 & 0 & 0 & 0 \\ C_{13} & 0 & 0 & 0 & 0 & 0 \\ C_{14} & 0 & 0 & 0 & 0 & 0 \\ C_{15} & 0 & 0 & 0 & 0 & 0 \\ C_{16} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} F_x \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3)$$

Repeating the process for all other loads produces the rest of the coefficients of the matrix C . If the bridge output is not zero at 'no load' condition a constant offset vector, G , has to be added to Eq. 1:

$$V = CF + G \quad (4)$$

Where: V - Sensor outputs (Bridges voltage)
 F - Load
 C - Compliance matrix
 G - Offset voltage vector

At the end of the calibration process a load applied to the sensor can be determined by:

$$F = C^{-1}[V - G] \quad (5)$$

The fabricated sensor has a sensitivity of about 5, natural frequency of about 700Hz and its stiffness matrix condition number is 70.

VI. LINK CONTROLLER DESIGN

Each control module, used to control the velocity and position of the piston, implements a digital PID controller with a sampling time of $T_s = 341$ microseconds. The required feedback is obtained by an optical encoder with a resolution of 21,600 counts per revolution which translates through the rack and pinion arrangement to 7832 counts per inch. The module output, which is fed to the servo valve amplifier, has the ranges $\pm 10V$. The PID algorithm implemented by these modules is given by the following difference equation:

$$u(n) = K_p e(n) + K_d [e(n) - e(n-1)] + K_i \sum_{n=0}^n e(n) \quad (6)$$

where: K_p - Proportional gain.
 K_d - Derivative Gain.
 K_i - Integration Gain
 $u(n)$ - Controller output at sampling period n .
 $e(n)$ - Velocity/Position error at sampling period n .

A dynamic model for the plant, (see Figure 6 for system block diagram) consisted of the amplifier, valve and piston connected in series,

was determined through open loop step and frequency response experiments (Figure 7):

$$G(s) = \frac{K e^{-0.011s} \left(\frac{s}{180} + 1 \right)}{\left(\frac{s}{56 + 38i} + 1 \right) \left(\frac{s}{56 - 38i} + 1 \right) \left(\frac{s}{105} + 1 \right)} \quad (7)$$

The static gains of the plant, K , was found experimentally to be approximately 6.73 inches/sec/volt and the time delay was measured directly from the step response. In order to design a sampled-data controller, $G(s)$ with additional ZOH was transformed to Z domain. The ZOH added to the plant a gain of 0.000305 volt/count that corresponds to the gain of the DAC on the controller module. The transformation of Eq. (7) to Z domain yields [8]:

$$G(z) = \frac{1.5912 \times 10^{-7} (z + 3.695)(z + 0.265)(z + \infty)}{z^{30} (z - 1)(z - 0.987)(z - 0.973)}$$

Similarly, the controller difference equation, Eq. (8), was transformed to Z domain:

$$\frac{U(z)}{E(z)} = \frac{(K_p + K_i + K_d)z^2 - (K_p + 2K_d)z + K_d}{(z - 1)z}$$

A PID controller was designed in Z frequency domain and then tuned on the actual system [8]. A PID controller with gains $K_p = 5$, $K_d = 500$, and $K_i = 10$ was found to produce adequate response as shown in Figure 8.

V. SOFTWARE DESCRIPTION

The software that controls each Stewart platform is divided into two major parts: 1) Menu driven user interface which allows the user to drive the platform in variety of modes; and 2) Real-time routines that perform the inverse kinematics, check for constraints

violation and communicate with the control modules. The software was implemented using Borland Turbo C++ programming language.

The user interface menu was design so that operator's errors will be minimized through "On-line Help" and Enabling/Disabling selections as the session progresses. In order to support variety of research activities, it provides several options by which velocity/position references can be fed to the controller. The following table illustrates the flow of menus and the options that are provided by the user interface:

Level 1	Level 2	Level 3	Level 4	Level 5
	Operation Selection	Input Source Selection	Action Selection	
Log in	Initilize System	-	Initialize Sequence	
	Platform Space	-Datafile -RS-232 -Joystick -Teach Pendent	-Move -Animate	
	Joint Space	-Datafile -RS-232 -Teach Pendent	-Move	
	Shut Down	-	Shut Down	Log off

The following provide a short description of the above options:

Login/Logoff: The user through a "Pass Word" gain access to the system and his activity during the session is recorded in a log file.

Initialization: This routine initializes the DCX motion controller's parameters, and moves the platform to the "Home" position defined as $X_{home} = [0, 0, 5.85, 0, 0, 0]$ where the element of X_{home} are x, y, z, roll, pitch and yaw. This pose correspondes to driving each link to $1/2(L_{min} + L_{max})$.

*Sixth Annual Conference on Recent Advances in Robotics
University of Florida, Gainesville FL, April 19-20, 1993*

Platform/Joint Space: This feature allows the user to select the space in which motion will take place. In case of "Platform Space" the inverse kinematics routine that calculate the requiried link lengths in invoked, while in "Joint Space" link lengths are directly given. In both cases references can be provided by:

Data File: Data file which contains precalculated poses (or lengths) along a required trajectory.

RS-232: References are computed by other computer and communicated to the controller through a serial port at 57000Baud

Joystick: A six degree-of-freedoms joystick is used to drive the platform (not available in "Joint Space").

Teach Pendent: This routine allows the user to move the platform along a straight line to a new pose, specified in absolute or relative coordinate systems, with a specific trapezodial velocity profile. by selecting options with a mouse, that are listed in a graphics window on the PC monitor.

Animation: This option allows the user to animate the platform motion and thus to determine whether the motion he had in mind will actually performed. Moreover, it make it possible to find out if any of the kinematic constraints would be violated during the planned motion. This option is extremly important since severe damage might occur in case of constraint violation.

Shut Down: This routine leads the user through the correct sequense of power shut down in order to eliminate hazard situations.

The Real-Time portion of the software performs the follwing tasks every 50 miliseconds:

1. Compute the required links' velocities using the inverse kinematic solution of the platform (1 milisecond).
2. Check whether a kinematic constraint is being violated, if so the motion is stopped (2miliseconds).

3. Communicate the new velocity references to the controller modules (22 milliseconds).
4. Check the status of the controller modules (25 milliseconds)

Most of the program execution time is devoted to communicating with the DCX motion control board, and together they consume about 90% of the program execution time T. In contrast, the inverse kinematic and the constraints checking subroutines use only about 6% of T.

VI. CONCLUSIONS

The design and construction of a "Space Emulator" that consists of two Stewart platforms whose velocity/acceleration is separately controlled by Admittance Controller is described. The design and calibration of the six degree-of-freedom force/torque sensor that is used as a feedback device for the admittance controller is discussed. A model for each link was derived and a PID velocity controller was designed. In addition, the structure of the software, which is composed of user interface and real time routines, is presented.

VII. ACKNOWLEDGEMENT

This work was supported by grants from the Technological Research and Development Authority of Florida.

VIII. REFERENCES

- 1 D. L. Akin, et. al, "Space Applications of Automation, Robotics and Machine Intelligence Systems", MIT Report, NASA Contract NAS-34381, Cambridge, MA, 1983.
2. Moris Fresco, "The Design and Implementation of a Computer Controlled Platform with Variable Admittance", M. Sc. thesis, MIT, January 1987.
3. Stewart, D., "A Platform with Six Degrees of Freedom", Proceedings of the Institution of Mechanical Engineering, Vol.180, Part 1, No. 15, 1965-66, pp.371-386.

4. Ficher, E. F., "A Stewart Platform Based Manipulator: General Theory and Practical Consideration", Journal of Robotic Research, Summer 1986, pp.157-182.
5. R. Bamford, C. J. Morrissey, "Improved Force-and-Torque Sensor Assembly", Nasa Tech Brief, Sep 1991, Vol 15.No 9, Item #4.
6. E. Bayo., J. R. Stubbe., "Analysis and Design of a Six Axis Truss Type of Force Sensor.", 4th International conference on Automation and Robotics, Ohio, June 1990.
7. P. C. Watson., S. H. Drake., "Pedestal and Wrist Force Sensors for Automatic Assembly", 5th Int Symposium on Industrial Robots, Sep 1975, pp501-511.
8. Franklin, G. F., Powell, D. J. and Workman, M. L., "Digital Control of Dynamic Systems", Addison-Wesley Publishing Company, Reading, Mass., 1990.
9. Masory O., Wang J., "Workspace Evaluation of A Stewart Platform", Proc. ASME Design Technical Conf., Scottsdale, Arizona, Sep. 13-16, 1992.

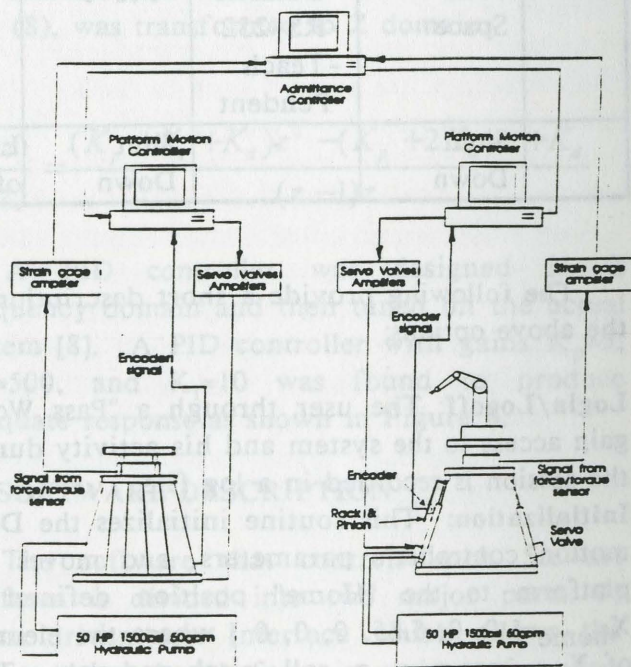


Figure 1: Emulator description.

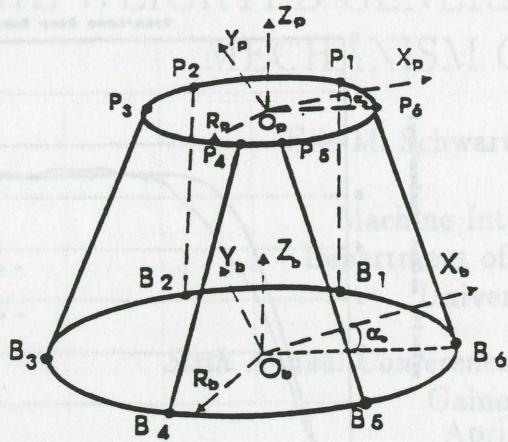


Figure 2: Stewart platform.

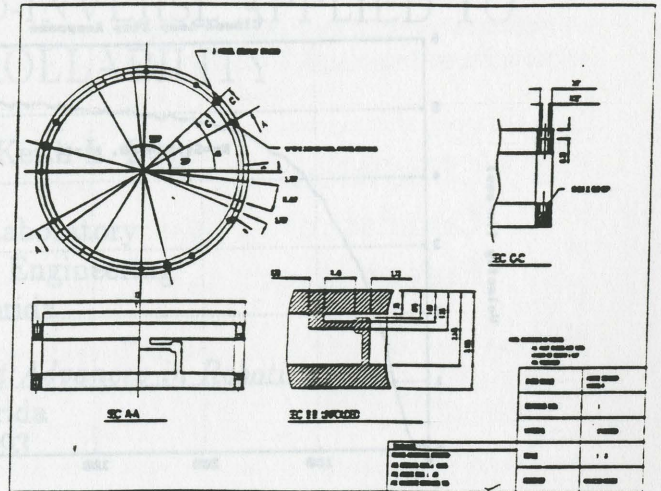


Figure 3: Force/Torque sensor.

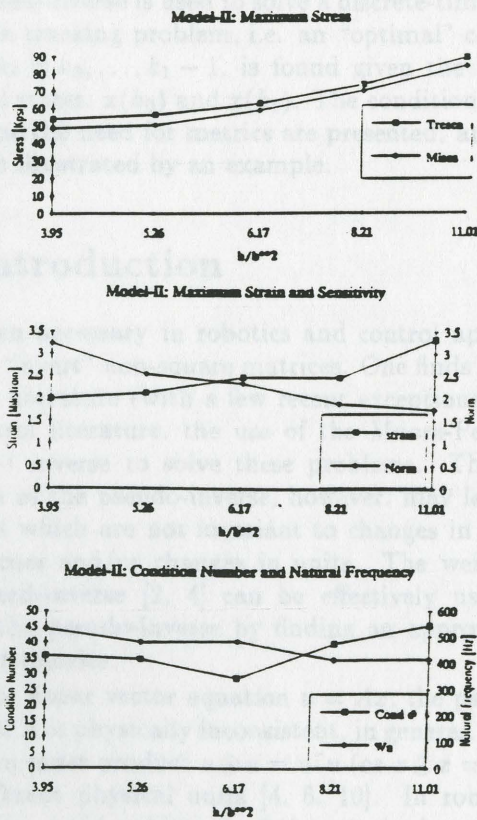


Figure 4: FEM Analysis results.

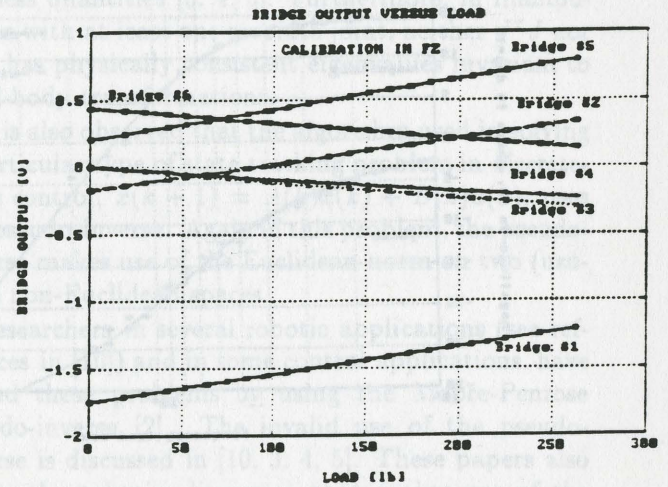


Figure 5: Sensor outputs versus force in X direction.

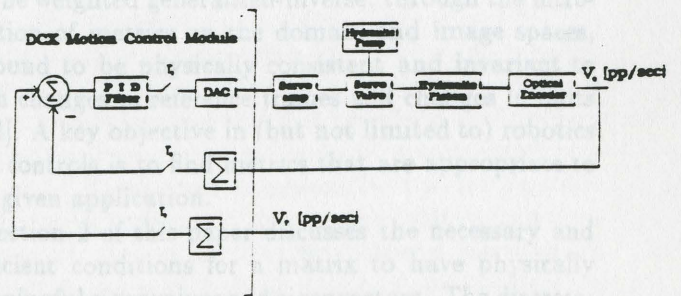


Figure 6: Controller block diagram.

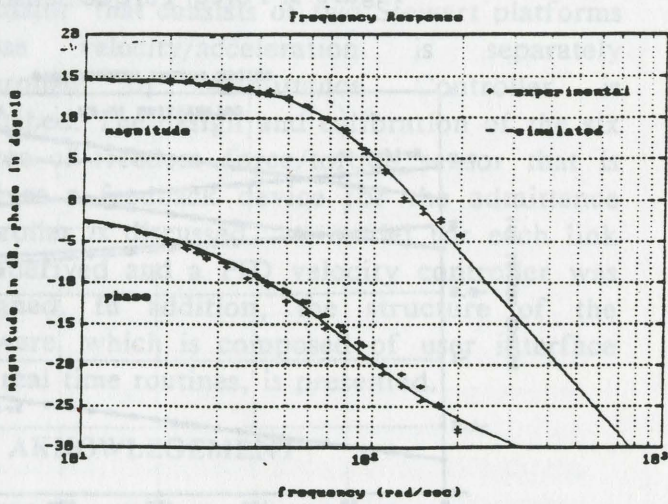
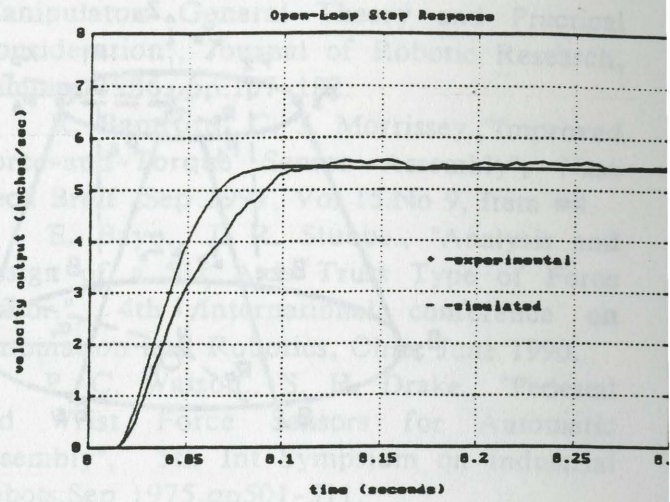
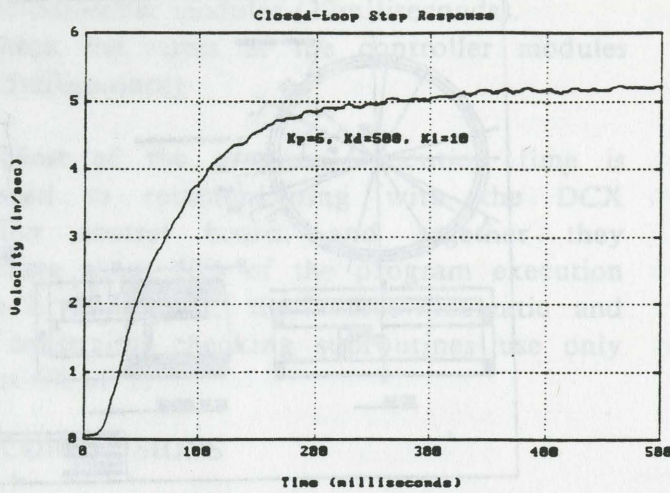


Figure 7: System open loop step and frequency responses.

Figure 8. Actual systems closed loop-step response with various controllers.

THE WEIGHTED GENERALIZED-INVERSE APPLIED TO MECHANISM CONTROLLABILITY

Eric M. Schwartz

Keith L. Doty

Machine Intelligence Laboratory
Department of Electrical Engineering
University of Florida

Sixth Annual Conference on Recent Advances in Robotics
Gainesville, Florida
April 19-20, 1993

Abstract

A units analysis of A , x , and u in the linear system $u = Ax$ leads to some generalizations about the eigenvalues of A and solutions x . The weighted generalized-inverse is used to solve a discrete-time control state tracking problem, i.e. an "optimal" control $u_i(k_i)$, $k_i = k_0, \dots, k_1 - 1$, is found given the initial and final states, $x(k_0)$ and $x(k_1)$. The conditions that determine the need for metrics are presented, and the theory is illustrated by an example.

1 Introduction

It is often necessary in robotics and control applications to "invert" non-square matrices. One finds in the robotics literature (with a few recent exceptions) and the control literature, the use of the Moore-Penrose (pseudo-) inverse to solve these problems. The application of the pseudo-inverse, however, may lead to solutions which are not invariant to changes in reference frames and/or changes in units. The weighted generalized-inverse [2, 4] can be effectively used to replace the pseudo-inverse by finding an appropriate metric or metrics.

For the linear vector equation $u = Ax$, the pseudo-inverse of A is physically inconsistent, in general, if the Euclidean inner product $u \odot u = u^T u$ (or $x \odot x = x^T x$) sum different physical units [4, 6, 10]. In robotics, A is often the Jacobian J of the manipulator, x is the joint velocity vector \dot{q} , and u is the 6-vector V composed of the three linear and three angular velocity components. The pseudo-inverse of J involves a term $J^T J$ (for J full column rank and manipulators with less than six joints) or $J J^T$ (for J full row rank and redundant manipulators, i.e. with more than six

joints). These matrix products often generate physically inconsistent terms that add length squared with unitless quantities [3, 4, 5]. Furthermore, in manipulators with at least one revolute joint, neither $J^T J$ nor $J J^T$ has physically consistent eigenvalues invariant to rigid-body transformations.

It is also observed that the algorithm used in solving a particular type of state tracking problem in discrete-time control, $x(k+1) = A(k)x(k) + B(k)u(k)$, uses the pseudo-inverse. Again in this problem, the pseudo-inverse makes use of the Euclidean norm on two (usually) non-Euclidean spaces.

Researchers in several robotic applications (see references in [10]) and in some control applications, have solved these problems by using the Moore-Penrose pseudo-inverse [2]. The invalid use of the pseudo-inverse is discussed in [10, 3, 4, 5]. These papers also refute the robotics literature that makes use of the "eigenvalues" and "eigenvectors" of matrices whose eigenvalues do not have physically consistent units and are not invariant to changes in scale or coordinate transformation.

The weighted generalized-inverse, through the introduction of metrics on the domain and image spaces, is found to be physically consistent and invariant to both changes in reference frames and changes in units [2, 4]. A key objective in (but not limited to) robotics and controls is to find metrics that are appropriate to the given application.

Section 2 of this paper discusses the necessary and sufficient conditions for a matrix to have physically meaningful eigenvalues and eigenvectors. The discrete-time control problem is formulated in Section 3. The application of weighted generalized-inverses in a state tracking control problem is given in Section 3.1. This section also discusses the conditions that determine the need for each metric. Finally, an example manipulator

system is analyzed in Section 3.2 to demonstrate conditions necessary for a system to have results invariant to scale and coordinate transformations.

2 Conditions for Physically Consistent Eigenvalues and Eigenvectors

When does a matrix A have physically consistent eigenvalues and eigenvectors? Let A be an $n \times n$ matrix,

$$A = \{a_{ij}\}, \quad (1)$$

and let the domain of A be \mathcal{X}^n , where \mathcal{X}^n is a space with physical units. The \mathcal{X}^n -space can be characterized as follows. Let β be an n -vector of possibly distinct physical units

$$\beta = [\beta_1 \ \beta_2 \ \dots \ \beta_n]^T. \quad (2)$$

Any $x \in \mathcal{X}^n$ is equivalent to an item-wise multiplication of β and y , $y \in \mathbb{R}^n$, i.e.

$$x = \beta \otimes y = [\beta_1 y_1 \ \beta_2 y_2 \ \dots \ \beta_n y_n]^T, \quad (3)$$

so that $\mathbb{R}^n \xrightarrow{\beta} \mathcal{X}^n$.

Theorem 1 *The equation $Ax = \lambda x$ is physically consistent if and only if $\text{units}[a_{kj}] \text{units}[x_j] = \text{units}[\lambda] \text{units}[x_k]$, for all j and k .*

Proof By hypothesis, $\sum_{j=1}^n a_{kj} x_j = \lambda x_k$ for all k . Recognizing that only identical physical units can be added together, we immediately conclude that $\text{units}[a_{kj}] \text{units}[x_j] = \text{units}[\lambda] \text{units}[x_k]$, for all j and k .

Now, assume $\text{units}[a_{kj}] \text{units}[x_j] = \text{units}[\lambda] \text{units}[x_k]$ for all j and k . Clearly, the equation $\sum_{j=1}^n a_{kj} x_j = \lambda x_k$ is physically consistent for all k , i.e. $Ax = \lambda x$ is physically consistent. ■

Observe that $\text{units}[a_{kj}] \text{units}[x_j] = \text{units}[\lambda] \text{units}[x_k]$ implies that $\text{units}[\lambda] = \text{units}[a_{ij}]$, for all i . Hence, any matrix with a physically consistent eigenvalue equation must have diagonal elements with the same physical units and all its eigenvalues must have those same units.

3 Weighted Generalized-Inverse in Discrete-Time Controls

A linear (or linearized) discrete-time control system can be described at time t_k by the state difference equation

$$x(k+1) = A(k)x(k) + B(k)u(k), \quad (4)$$

where $x(k)$ is the state vector, $x(k) \in \mathcal{X}^n$, and $u(k)$ is the input vector, $u(k) \in U^p$. If the matrices $A(k)$ (an $n \times n$ matrix) and $B(k)$ (an $n \times p$ matrix) are not functions of k (i.e. they are constant matrices), the system is time-invariant.

The state solution to (4) for $k > k_0$ — which can be found by iteration of (4) — is

$$x(k) = \phi(k, k_0)x(k_0) + \sum_{j=k_0}^{k-1} [\phi(k, j+1)B(j)u(j)], \quad (5)$$

where $\phi(k, j)$ is the $n \times n$ discrete-time state transition matrix,

$$\phi(k, j) = \begin{cases} A(k-1)A(k-2)\dots A(j), & \text{for } k > j \\ I_n, & \text{for } k = j \\ \text{not defined,} & \text{for } k < j \end{cases}, \quad (6)$$

and I_n is the $n \times n$ identity matrix. If the system of (4) is time-invariant, then

$$\phi(k, j) = \begin{cases} A^{k-j}, & \text{for time-invariant and } k \geq j \\ \text{not defined,} & \text{for } k < j \end{cases}, \quad (7)$$

Equation (5) at time $k_1 > k_0$ can be written as

$$x(k_1) = \phi(k_1, k_0)x(k_0) + P(k_0, k_1)\mu, \quad \text{for } k_1 > k_0, \quad (8)$$

where μ is the $(k_1 - k_0)p$ -vector of inputs,

$$\mu = \begin{bmatrix} u(k_1 - 1) \\ u(k_1 - 2) \\ \vdots \\ u(k_0 + 1) \\ u(k_0) \end{bmatrix}, \quad (9)$$

and

$$P(k_0, k_1) = [B(k_1 - 1), \dots, \phi(k_1, k_1 - 1)B(k_1 - 2), \phi(k_1, k_1 - 2)B(k_1 - 3), \dots, \phi(k_1, k_0 + 1)B(k_0)], \quad \text{for } k_1 > k_0. \quad (10)$$

where $P(k_0, k_1)$ is a $n \times (k_1 - k_0)p$ matrix. If the system of (4) is time-invariant, then

$$P(k_0, k_1) = \begin{bmatrix} B & AB & A^2B & \dots & A^{(k_1 - k_0 - 1)}B \end{bmatrix},$$

for time-invariant systems and $k_1 > k_0$. (11)

In both the time-varying and the time-invariant case, $P(k_0, k_1)$ is known as the controllability matrix of the system. A digital system is *completely state controllable* [8] if and only if the rank of $P(k_0, k_1)$ is n , i.e. the controllability matrix has full row rank. Observe that P typically does not satisfy Theorem 1. This has significant implications to the solution of the state tracking problem.

3.1 Solving for the Input μ

To solve for μ (with $k_1 > k_0$), (8) can be rewritten as

$$x(k_1) - \phi(k_1, k_0)x(k_0) = P(k_0, k_1)\mu, \quad (12)$$

or

$$y = P\mu, \quad (13)$$

where $y \stackrel{def}{=} x(k_1) - \phi(k_1, k_0)x(k_0)$ and $P \stackrel{def}{=} P(k_0, k_1)$.

If P is a square matrix with rank $n = (k_1 - k_0)p$, then P can be inverted and used to solve (12) for μ ,

$$\mu = P^{-1}(k_0, k_1) [x(k_1) - \phi(k_1, k_0)x(k_0)],$$

for $P(k_0, k_1)$ full rank. (14)

If $P(k_0, k_1)$ does not have full rank, then a pseudo-inverse of $P(k_0, k_1)$ might be considered in order to solve (12) for the inputs μ . As in the several instances in robotics where the pseudo-inverse is inappropriately employed, the use of the pseudo-inverse in this control problem may in certain circumstances not result in the minimum-norm least-squares solution expected.

If $P(k_0, k_1)$ has full row rank and $n < (k_1 - k_0)p$, then the solution to (12) with minimum-norm $\|\mu\|$ that minimizes the least-squares error $\|y - P\mu\|$ is often (inappropriately) taken as

$$\mu \stackrel{?}{=} P^\dagger(k_0, k_1) [x(k_1) - \phi(k_1, k_0)x(k_0)],$$

for $P(k_0, k_1)$ full row rank. (15)

where $P^\dagger = P^T(P P^T)^{-1}$. When $P(k_0, k_1)$ does not have full row rank (rank n), the system is not completely state controllable and no exact solution can be found, although the pseudo-inverse for full column rank P is $P^\dagger = (P^T P)^{-1} P^T$.

Doty *et. al.* [4] and Schwartz [10] show that in general, the pseudo-inverse does not apply unless μ and

y have physically consistent Euclidean norms. The weighted generalized-inverse, however, can be used to solve this problem with the appropriate use of metrics on the input and state spaces.

The weighted generalized-inverse that gives the minimum M_μ -norm, M_x -least-squares solution to $y = P\mu$ (see [4, 2]) is

$$P^\# = C^\# F^\# \quad (16)$$

$$F^\# = (F^T M_x F)^{-1} F^T M_x \quad (17)$$

$$C^\# = M_\mu^{-1} C^T (C M_\mu^{-1} C^T)^{-1} \quad (18)$$

where $P = FC$ is a full-rank factorization, F full column rank and C full row rank.

There are three special conditions of P that may occur to simplify the equation for $P^\#$:

- If P has full rank, then $P^\# = P^{-1}$ and no metrics on either the μ or x spaces are needed. Let F or C equal the identity matrix to easily verify this fact.
- If P has full column rank, then $P^\# = F^\#$ and no metric on the μ space is needed. Let C equal the identity matrix to easily verify this fact.
- If P has full row rank, then $P^\# = C^\#$ and no metric on the x space is needed. Let F equal the identity matrix to easily verify this fact.

3.2 Examples Using the Pseudo- and Weighted Generalized-Inverses

To clarify the problems with the pseudo-inverse in the above controls problem, the following examples will show that the pseudo-inverse is not invariant to either scaling or coordinate transformations (on the input space, for this example).

Consider the manipulator system of Figure 1. (In chapter 6 of Koivo [7], an alternate manipulator is given that would also demonstrate the concepts of this section.) Two masses M_1 and M_2 are separated by spring k_1 . A second spring k_2 and a damper c_2 connect M_2 to a revolute joint assumed fixed at some arbitrary angle for these examples. Mass M_1 has force f_1 acting on it. This force is created by torque τ_1 on the pulley with radius r . (Assume that the pulleys contribute nothing to the dynamics of the problem we are discussing other than the creation of the force acting on M_1 and that there is no slippage on the pulleys.) Force f_2 is acting on mass M_2 . The v , w , and z coordinate systems are all shown in the figure, where v is defined as the displacement of M_1 from the position of M_1 when the springs are uncompressed. w is defined

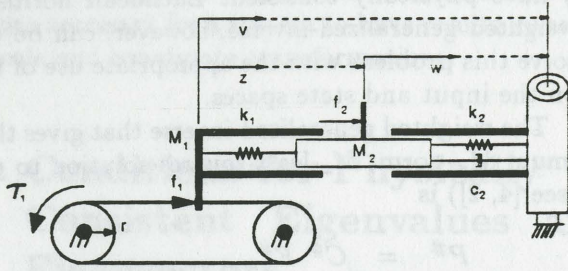


Figure 1: A manipulator system.

as the displacement of M_2 from the position of M_2 when spring k_2 is uncompressed, and z is the relative displacement of M_2 from M_1 , i.e. $z = v - w$.

The following continuous-time equation models this mechanical system for $x(t) = [v(t), w(t), \dot{v}(t), \dot{w}(t)]^T$ and $u(t) = [\tau_1(t), f_2(t)]^T$, where $\tau_1 = -r f_1$:

$$\dot{x}(t) = A x(t) + B u(t) \quad (19)$$

where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1}{M_1} & \frac{k_1}{M_1} & 0 & 0 \\ \frac{k_1}{M_2} & -\frac{(k_1+k_2)}{M_2} & 0 & -\frac{c_2}{M_2} \end{bmatrix}, \quad (20)$$

and

$$B^T = \begin{bmatrix} 0 & 0 & -\frac{1}{rM_1} & 0 \\ 0 & 0 & 0 & \frac{1}{M_2} \end{bmatrix}. \quad (21)$$

Since this A matrix satisfies the conditions of Theorem 1, the eigenvalues of A are physically meaningful. It is easily proven that all control state-space formulations have an A matrix with physically consistent eigenvalues.

Corollary 1 For the linear state differential system of (19), the matrix A has physically consistent eigenvalues and eigenvectors.

Proof From the physical consistency of the state differential equation, $\text{units}[x_k] = \text{units}[a_{kj}]\text{units}[x_j] = \text{units}[x_k]/\text{units}[\text{time}]$. This is identical to Theorem 1 if $\text{units}[\lambda] = 1/\text{units}[\text{time}]$, which is easily verified with a units analysis on (19) and $Ax = \lambda x$. ■

The continuous-time formulation can be recast as a discrete-time problem through use of the following equations [9]:

$$x(i+1) = A_d x(i) + B_d u(i) \quad (22)$$

where the discrete-time state $x(i) = [v(i), w(i), \dot{v}(i), \dot{w}(i)]^T$ and the input $u(i) = [\tau_1(i), f_2(i)]^T$ represent the same variables as in the original continuous-time system. and

$$A_d = e^{AT} \quad (23)$$

$$B_d = \left(\int_0^T e^{A\tau} d\tau \right) B \quad (24)$$

$$T = t_{i+1} - t_i, \text{ assumed constant for all } i. \quad (25)$$

The following numeric values were used for this example:

$$\begin{aligned} k_1 &= 5 \frac{\text{kg}}{\text{s}^2}, & M_1 &= 1\text{kg}, & c_2 &= 10 \frac{\text{kg}}{\text{s}}, & r &= 1\text{m} \\ k_2 &= 5 \frac{\text{kg}}{\text{s}^2}, & M_2 &= 2\text{kg}, & T &= 0.5\text{s} \end{aligned} \quad (26)$$

With the above values, the continuous-time system matrices (A, B) are transformed into the following discrete-time system matrices (A_d, B_d):

$$A_d = \begin{bmatrix} 0.4561 & 0.5243 & 0.4042 & 0.05447 \\ 0.1260 & 0.7284 & 0.02723 & 0.1523 \\ -1.885 & 1.749 & 0.4561 & 0.2520 \\ 0.2445 & -0.6251 & 0.1260 & -0.03281 \end{bmatrix} \quad (27)$$

$$B_d = \begin{bmatrix} -0.1127 & 0.003917 \\ -0.003917 & 0.02911 \\ -0.4042 & 0.02723 \\ -0.02723 & 0.07613 \end{bmatrix} \quad (28)$$

The initial and final states at discrete times i_0 and i_1 , respectively, are given as

$$\begin{aligned} x_0 = x(i_0) &= [0.1\text{m}, 0.1\text{m}, 1.0 \frac{\text{m}}{\text{s}}, 1.0 \frac{\text{m}}{\text{s}}]^T \\ x_f = x(i_1) &= [0.0\text{m}, 0.0\text{m}, 2.0 \frac{\text{m}}{\text{s}}, 2.0 \frac{\text{m}}{\text{s}}]^T \end{aligned}$$

To obtain a two step control — i.e. a control that takes two steps to move from the initial state to the final state — let $i_0 = 0$ and $i_1 = 2$. Since $P(i_0, i_1) = P(0, 2)$ is a full rank matrix, (14) is used to determine the inputs $\mu = [u(1), u(0)]$,

$$\begin{aligned} \tau_1(1) &= -8.420\text{N-m} & f_2(1) &= +21.64\text{N} \\ \tau_1(0) &= +4.301\text{N-m} & f_2(0) &= -23.64\text{N} \end{aligned} \quad (29)$$

The three step problem employs the same initial and final states as the two step problem. This time $i_0 = 0$ and $i_1 = 3$. After applying (15) with the pseudo-inverse $P^\dagger(i_0, i_1) = P^\dagger(0, 3)$ — since $P(0, 3)$ has full row rank — the three calculated inputs are

$$\begin{aligned} \tau_1(2) &= -2.157\text{N-m} & f_2(2) &= +21.43\text{N} \\ \tau_1(1) &= -4.884\text{N-m} & f_2(1) &= -11.09\text{N} \\ \tau_1(0) &= +7.492\text{N-m} & f_2(0) &= -8.659\text{N} \end{aligned} \quad (30)$$

The non-invariance of the pseudo-inverse to scaling will now be shown using the example problem formulated above. Let S be a diagonal scaling matrix for the input space of vectors $u(i)$ of the form

$$S = \begin{bmatrix} \frac{1}{\tau_s} & 0 \\ 0 & \frac{1}{f_s} \end{bmatrix} \quad (31)$$

Without changing the validity of the system equations (either continuous-time or discrete-time), the scaling matrix may be used as follows:

$$Bu = B(S^{-1}S)u = (BS^{-1})(Su) = B'u' \quad (32)$$

The continuous time matrices, B and $u(t)$, are transformed into the following:

$$B' = \begin{bmatrix} 0 & 0 & -\frac{\tau_s}{rM_1} & 0 \\ 0 & 0 & 0 & \frac{f_s}{M_2} \end{bmatrix}^T, \quad u'(t) = \begin{bmatrix} \tau_1 \\ \tau_s \\ f_2 \\ f_s \end{bmatrix} \quad (33)$$

If the scaling variables are given the values $\tau_s = 1\text{N-m}$ and $f_s = 1\text{N}$, the scaling matrix is the identity matrix, the units of the inputs torques and forces are dimensionless, and the results are identical to those of the previous section.

But now let us make a scaling matrix other than the identity matrix. Let

$$\begin{aligned} \tau_s &= \frac{1\text{N-m}}{10\text{N-dm}} = 0.1 \frac{\text{N-m}}{\text{N-dm}} \\ f_s &= 1 \end{aligned} \quad (34)$$

where dm is an abbreviation for decimeter (equal to 0.1 meters). These scaling elements change the units of the inputs. The new B and $u(t)$ matrices are

$$B' = \begin{bmatrix} 0 & 0 & -\frac{0.1}{rM_1} & 0 \\ 0 & 0 & 0 & \frac{1}{M_2} \end{bmatrix}^T, \quad u'(t) = \begin{bmatrix} \tau_1' \\ f_2' \end{bmatrix} \quad (35)$$

where $\tau_1' = 10\tau_1$.

These scaled equations are solved using the same initial state and final state used in the previous examples. When $P(i_0, i_1) = P(0, 2)$ is a full rank matrix, the resulting solution of inputs μ is

$$\begin{aligned} \tau_1(1) &= -84.20\text{N-dm} & f_2(1) &= +21.64\text{N} \\ \tau_1(0) &= +43.01\text{N-dm} & f_2(0) &= -23.64\text{N} \end{aligned} \quad (36)$$

which, as expected, equals the solution of (29).

For the case when $P(i_0, i_1) = P(0, 3)$ is not invertible, but has full row rank, the pseudo-inverse is applied. The resulting solution of inputs is

$$\begin{aligned} \tau_1(2) &= -55.02\text{N-dm} & f_2(2) &= +21.33\text{N} \\ \tau_1(1) &= -15.98\text{N-dm} & f_2(1) &= -15.81\text{N} \\ \tau_1(0) &= +42.82\text{N-dm} & f_2(0) &= -6.653\text{N} \end{aligned} \quad (37)$$

These inputs are not equal to the solution of (30) obtained before the problem was scaled. In fact, the solutions are significantly different.

When P has full rank or full row rank, the solution μ is invariant to both scaling and coordinate transformations on the states x . This can easily be shown by rewriting the system equations in terms of the coordinates z and w , for example, instead of v and w .

When \dot{P} is not invertible but does have full row rank, the pseudo-inverse solution is not invariant to coordinate transformations on the inputs u . This is easily seen by recycling the above example which demonstrated the non-invariance of the input space to scaling. Assume a coordinate transformation on u that will convert the torque τ_1 into a force $f_1 = \tau_1/r$, where r is the radius of the pulley. This coordinate transformation can be cast into (31) by letting $\tau_s = r$ and $f_s = 1$. Therefore, since scaling of u was found to cause inconsistent results, so too will coordinate transformations on u .

We next present an example which requires the use of a metric for the state vector. Eliminating either of the inputs τ_1 (or f_1) or f_2 will make B and B_d 4×1 matrices. Since P has full row rank when 4 or more steps are used in the control, no metric on x is required to give solutions which are invariant to coordinate transformations and scaling. When P does not have full row rank (3, 2, or 1 step control), a metric on x is required. If no metric is explicitly used, one implicitly assumes an identity scaling matrix (for M_x) which does not give solutions invariant to coordinate transformations or scaling [4].

The generalized-inverse for the 3 step control problem gives results invariant to coordinate transformations on the state vector x , whereas the pseudo-inverse solutions are not invariant to these transformations. The two coordinates systems defined by $x = [v, w, \dot{v}, \dot{w}]^T$ and $x' = [z, w, \dot{z}, \dot{w}]^T$ will be used to illustrate this point. The generalized-inverse solution $P^\#y$ with state variable x will use the metric M_x to minimize the instantaneous energy E of the system, where $E = x^T M_x x$ and

$$M_x = \begin{bmatrix} k_1 & -k_1 & 0 & 0 \\ -k_1 & k_1 + k_2 & 0 & 0 \\ 0 & 0 & M_1 & 0 \\ 0 & 0 & 0 & M_2 \end{bmatrix} \quad (38)$$

so that $|y - PP^\#y|$ is minimized. The metric for the system with state variable x' is

$$M'_x = T^{-T} M_x T^{-1} = \begin{bmatrix} k_1 & 0 & 0 & 0 \\ 0 & k_2 & 0 & 0 \\ 0 & 0 & M_1 & M_1 \\ 0 & 0 & M_1 & M_1 + M_2 \end{bmatrix} \quad (39)$$

where T is the coordinate transformation matrix defined by $x' = Tx$,

$$T = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (40)$$

Consider the system of Figure 1 with only the single input f_2 , i.e. $\tau_1 = f_1 = 0$. Since P has full column rank in the 3 step control, $P^\# = F^\#$ and (17) is the generalized-inverse of P . For the coordinates systems defined by state vectors x and x' , the 3 step control calculated with the generalized-inverse are equal: $\mu_g = \mu'_g = [17.78, 0.7317, 0.7846]^T N$, whereas for the pseudo-inverse the solutions are completely different: $\mu_p = [28.46, 33.09, -31.28]^T N$ and $\mu'_p = [16.67, 12.25, -2.672]^T N$!

4 Conclusions

It has been shown that the Euclidean norm is inadvertently applied to systems in both robotics and controls through the use of the pseudo-inverse. The necessary and sufficient conditions for a matrix to have physically consistent eigenvalues and eigenvectors is given. While the A matrix in the linear state difference (or differential) equation satisfies these conditions, the state transition matrix P does not, in general.

To illustrate the issues involved, a particular discrete-time control state tracking problem is formulated using the weighted generalized-inverse, instead of the pseudo-inverse, since the latter may lead to non-invariant results to both scaling and coordinate frame transformations while the former does not. The conditions that determine the need for each of the two metrics of the weighted generalized-inverse are given. A stylized manipulator system is analyzed and several examples are given that show the conditions in which the pseudo-inverse solutions are not invariant to scaling or coordinate transformations, whereas the generalized-inverse is invariant to both.

References

- [1] Åström, Karl J., and Wittenmark, Björn, 1990. *Computer Controlled Systems: Theory and De-*

sign. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

- [2] Ben-Israel, Adi, and Greville, Thomas N.E., 1974. *Generalized Inverses: Theory and Applications*. John Wiley and Sons, Inc., New York.
- [3] Doty, Keith L., 1992. "An Essay on the Application of Weighted Generalized-Inverses in Robotics." Fifth Conf. on Recent Advances in Robotics, Florida Atlantic University, Boca Raton, FL, June 1992, pp. 476-500.
- [4] Doty, Keith L., and Melchiorri, Claudio, and Bonivento, Claudio, 1993. "A Theory of Generalized Inverses Applied to Robotics." Intl. Journal of Robotics Research, 12(1):1-19, Feb. 1993.
- [5] Doty, Keith L., and Melchiorri, Claudio, and Bonivento, Claudio, 1991. "What is Wrong with Robot Kinematic- and Dynamic- Manipulability Theory?" MIL0991KLD. Machine Intelligence Laboratory, University of Florida, Gainesville, Florida.
- [6] Duffy, Joseph, 1990. "The Fallacy of Modern Hybrid Control Theory that is Based on 'Orthogonal Complements' of Twist and Wrench Spaces." Intl. Journal of Robotics Research, 7(2):139-144, 1987.
- [7] Koivo, Antti J., 1989. *Fundamentals for Control of Robotic Manipulators*. John Wiley and Sons, Inc., New York.
- [8] Kuo, Benjamin C., 1980. *Digital Control Systems*. Holt, Rinehart and Winston, Inc., New York.
- [9] Kwakernaak, Huibert, and Sivan, Raphael, 1972. *Linear Optimal Control Systems*. John Wiley and Sons, Inc., New York.
- [10] Schwartz, Eric M., and Doty, Keith L., 1992. "Application of the Weighted Generalized-Inverses in Robotics and Discrete-Time Controls." Fifth Conf. on Recent Advances in Robotics, Florida Atlantic University, Boca Raton, FL, June 1992, pp. 444-462.

OPTIMAL SELECTION OF MANUFACTURING TOLERANCES FOR SERIAL MECHANISMS

Oren Masory and Arun Kumar
Department of Mechanical Engineering
Florida Atlantic University
Boca Raton, FL 33431

ABSTRACT

The selection of manufacturing tolerances for serial manipulators, described by links and joints, is determined by cost constraint and required accuracy. These tolerances directly effect the accuracy of the mechanism and the manufacturing cost of the its elements. Once both the accuracy and the cost are expressed as function of these tolerances, a dual problem namely: 1) For a given cost find a set of tolerances which will optimize the mechanism accuracy; and 2) For a required accuracy find a set of tolerances which will minimize the cost, can be solved. This paper presents a methodology by which these two problems are formulated and solved.

I. INTRODUCTION

Serial manipulators can be considered as an open-chain mechanisms which are constructed by consecutive links connected by rotational or prismatic joints, each has one degree of freedom. The end point pose (pose is referred to as position and orientation) of such a mechanism can be specified either in task space or in joint space [1]. There is a direct conversion between the two descriptions given by the a kinematic model which is used to drive the manipulator in joint space to achieve a required pose in the task space. Pose error, defined as the difference between the required and the actual pose of the mechanism end point in task space, are caused by geometric and nongeometric errors. Geometric errors arise due

to dimensional imperfection of the mechanism caused by the assembly and the manufacturing process of its elements. These imperfections manifest themselves as deviations between the theoretical (nominal) model and the unknown model describing the actual mechanism. Since the nominal model is used to drive the mechanism, pose error is produced.

There are two ways to reduce the pose error due to geometric errors: 1) Calibration - which refers to a process in which the kinematic parameters of the model used to drive the mechanism is modified to better match the model of the actual mechanism [3,7] ; and 2) Construct/fabricate the mechanism to closely match its nominal kinematic model by specifying very tight manufacturing tolerances. From a design point of view, the calibration process has the drawback that it fails to indicate the actual sources of errors, and as a result, the designer is deprived of the necessary feedback to improve future designs.

The design of a mechanism is based on a set of specifications such as workspace, payload, velocity, acceleration, accuracy and cost. The workspace and the payload requirements determine the size and the nominal dimensions of the mechanism elements, its kinematic configuration and the size of its drives. All these factors effect the cost of the mechanism, however, if accuracy is a major design parameter the cost will be substantially influenced by the specified manufacturing tolerances. On one hand tight tolerances will increase production costs, and on the other hand loose tolerances will increase the

mechanism pose error.

The effect of manufacturing tolerances on the mechanism's pose error was investigated by few researchers. In [13] the pose error of planar mechanisms was considered and in [15] a particular distribution of the tolerances was assumed for the same purpose. In [12] an analytical model, based on dual screw transformation matrix method, was used for error analysis of three dimensional spatial mechanisms due to manufacturing tolerances. The model was used to describe the tolerances of links as well as the clearance at the joints and in turn provides estimation for the maximum pose error. However, it fails to provide estimation of the error due to a particular tolerance and indication for the cost associated with accuracy improvement.

This paper presents a procedure by which the effects of manufacturing tolerances on the mechanism pose error and its cost can be determined and evaluated. It can help the designer to specify the required tolerances by providing a solution for the following dual optimization problem: 1) For a given cost find a set of tolerances which will minimize the mechanism pose error; or 2) For a given pose error value find a set of tolerances which will minimize the cost.

The proposed procedure follows these steps:

1. The mechanism is described as an assembly sequence of 'Standard Elements' which include links and rotary/prismatic joints.
2. The nominal dimensions as well as the manufacturing tolerances of each individual element is specified by the designer.
3. Based on the nominal dimensions and the tolerances, a kinematic model is automatically determined for each element.
4. These models are automatically 'assembled' according to the specified mechanism configuration to obtain a kinematic model for the whole mechanism.
5. Using the mechanism model the norm of the pose error is calculated.
6. Base on the specified tolerances and the nominal dimensions of each element the

manufacturing cost of the mechanism is calculated.

7. At this point both the cost and the pose error are available and the dual optimizing mentioned above is solved.

II. KINEMATIC MODELING

Standard Elements and Basic Elements

Open-chain mechanisms are constructed by consecutive links connected to each other by rotational or prismatic one degree of freedom joints. These elements are defined as Standard Elements (SE). A Standard Element might be constructed by Basic Elements (BE) that are mechanical structure of simpler form.

Modeling of links:

The tolerances specified for a link are length and straightness as shown in figure 1. For a link of length L the nominal transformation between frame {1} and frame {2} is given by:

$$T_{12} = \begin{bmatrix} 1 & 0 & 0 & L \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Due to the length tolerance the value of the link length L changes to $L + \delta L$, where δL varies between upper and lower limits. δL causes a deviation in the position of frame {2} from the ideal position. This deviation is expressed as an error matrix as follows:

$$T_{e1} = \begin{bmatrix} 1 & 0 & 0 & \delta L \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

The straightness tolerance, specified in XY

and XZ planes, causes position and orientation deviations of frame {2} from the ideal one. When straightness tolerance is specified in XZ plane it results in a differential rotation about Y axis and translation along the Z axis. These changes are expressed as error matrix given by:

$$T_{e2} = \begin{bmatrix} 1 & 0 & \delta\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\delta\beta & 0 & 1 & \delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where $\delta\beta$ is given by:

$$\delta\beta = \tan^{-1} \left(\frac{UL - LL}{2L} \right) \quad (4)$$

where UL and LL are upper and lower limits for straightness in XZ plane.

Similarly, for straightness specified in XY plane, the rotation is about the Z axis and translation along Y axis. The error matrix due to this tolerance is given by:

$$T_{e3} = \begin{bmatrix} 1 & -\delta\gamma & 0 & 0 \\ \delta\gamma & 1 & 0 & \delta y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

where $\delta\gamma$ is defined the same way as $\delta\beta$

If the effect of other tolerances are neglected the total error matrix for the link due to manufacturing tolerances can be expressed as:

$$T_e = T_{e1} T_{e2} T_{e3} \quad (6)$$

and the new transformation from frame {1} to frame {2} is given by :

$$T_{12}^{new} = T_{12} T_e \quad (7)$$

Modeling of a Revolute Joint with Rotational

Axis Perpendicular to its Central Axis:

A revolute joint is constructed by two Basic Elements, shown in figure 2, each is analyzed independently. Three different types of manufacturing tolerances are considered for each Basic Element: 1) Flatness of the attaching faces; 2) Straightness of the central axis.; and 3) Perpendicularity of the rotational axis to the central axis. The flatness of attachment surface mainly changes the orientation of the YZ plane, as shown in figure 3, that might rotate about Y axis and/or about Z axis. Considering planar rotations and small angle approximations, two error matrices, T_{e1} and T_{e2} are resulted:

$$T_{e1} = \begin{bmatrix} 1 & 0 & \delta\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\delta\beta & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

$$T_{e2} = \begin{bmatrix} 1 & -\delta\gamma & 0 & 0 \\ \delta\gamma & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where the angle $\delta\beta$ and $\delta\gamma$ are as shown in figure 3.

The straightness specification is similar to the one explained in the above section and produces error matrices T_{e3} and T_{e4} . The tolerances specified for perpendicularity affect the orientation of the center frame as shown in figure 4. The differential rotations about Z axis and X axis result in error matrices T_{e5} and T_{e6} . The total transformation matrix for the first Basic Element from frame {1} to the center of the joint can be expressed as:

$$T_{p1} = T_{e1} T_{e2} T_{1c} T_{e3} T_{e4} T_{e5} T_{e6} \quad (9)$$

where the transformation matrix T_{1c} describes the ideal relation between frame {1} and the frame at the center of the joint.

A similar procedure is adopted in defining the transformation matrix for the second Basic Element of the revolute joint. However, in this case the transformation matrix T_{p2} is obtained starting from the center of the joint and moving towards frame {2} as follows:

$$T_{p2} = T_{e6} T_{e5} T_{e4} T_{e3} T_{e2} T_{e1} \quad (10)$$

where the transformation matrix T_{e2} describes the ideal relation between the center frame of the joint and frame {1}.

The total transformation matrix for the joint is defined by combining T_{p1} and T_{p2} as follows:

$$T_{12} = T_{p1} R(k, \theta) T_{p2} \quad (11)$$

where $R(k, \theta)$ is a rotation matrix about k axis (Y or Z).

Modeling of a Revolute Joint with Rotational Axis Aligned with its Central Axis:

The attachment surface is in YZ plane as shown in figure 5. There are two types of planar errors produced due to this manufacturing tolerance. The error matrix T_{e1} is due to the differential rotation about Y axis and T_{e2} is due to the small angle rotation about the Z axis and are given by Eqs. (3) and (5).

Any deviations from the perpendicularity to the YZ plane will result in differential rotations about Y and Z axes expressed by error matrices T_{e3} and T_{e4} . The straightness of the outer column in XZ plane will produce differential rotation about Y axis and translation along Z axis leading to an error matrix, T_{e5} . A similar analysis is performed for straightness in XY plane resulting in rotation about Z axis and translation along Y axis. The error matrix is represented by T_{e6} .

The total transformation matrix for the joint is defined by:

$$T_{12} = R(x, \alpha) T(L, 0, 0) T_{e1} T_{e2} T_{e3} T_{e4} T_{e5} T_{e6} \quad (12)$$

where: $R(x, \alpha)$ - Rotation matrix about the central axis of the joint.

$T(L, 0, 0)$ - Translation matrix along the central axis of the joint.

Modeling of a Prismatic Joint:

A prismatic joint, constructed by two Basic Elements, provides translatory motion along its principle axes, as shown in figure 6. The errors, produced due to the manufacturing tolerances are mainly functions of the geometry of the fixed member. The important manufacturing tolerances that are considered for the fixed member are: 1) Straightness of the central axis; 2) Flatness of the attachment face; and 3) Perpendicularity of the central axis to the attachment face.

The joint error analysis is conducted by assigning frame {1} to the fixed member and frame {2} to the moving member. Under ideal conditions the transformation between the frames is given by Eq. (1) where L is the translation of the frame {2} relative to frame {1}. The analysis for the fixed member of the prismatic joint is similar to the analysis for a link but the errors are functions of L . As a result the differential translations δy and δz are expressed by:

$$\begin{aligned} \delta y &= k \delta \Gamma \\ \delta z &= k \delta \beta \end{aligned} \quad (13)$$

The differential rotations $\delta \Gamma$ and $\delta \beta$ are calculated as discussed earlier in section 3.5.1 and the corresponding error matrices are expressed as:

$$T_{e1} = \begin{bmatrix} 1 & 0 & \delta \beta & 0 \\ 0 & 1 & 0 & 0 \\ -\delta \beta & 0 & 1 & \delta z(k) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{e2} = \begin{bmatrix} 1 & -\delta\Gamma & 0 & 0 \\ \delta\Gamma & 1 & 0 & \delta y(k) \\ \rho_{14} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The tolerances specified on the attachment face will affect the orientation of the frame {1} which in turn changes the position of frame {2}. These changes are expressed as error matrices T_{e3} and T_{e4} .

Any deviations from the perpendicularity of the central axis result in position errors at frame {2}. These deviations are with respect to the Z and/or Y axis and are expressed as error matrices T_{e5} and T_{e6} .

The total transformation from frame {1} to frame {2} is given by:

$$T_{12}^{new} = T_{e3} T_{e4} T_{e5} T_{e6} T_{12} T_{e1} T_{e2} \quad (15)$$

where the transformation matrix T_{12} describes the ideal relation between frames {1} and {2}.

Modeling of Assembly Errors:

The mechanism assembly is the process of joining the standard elements in a sequential order to achieve the required configuration. In order to assemble SE_i to SE_{i-1} its assembly location, location of frame {1}_i with respect to frame {2}_{i-1}, is used. This process is achieved by adopting the following steps:

1. Allocate a global frame {g} which defaults to frame {1} of the first SE.
2. Allocate frames to each SE.
3. Identify the sequence in which the SEs are to be assembled starting at the first SE.
4. This is the assembly process which involves three stages:
 - a. Defining the location of SE to be assembled with respect to previous frame.
 - b. Incorporating interface error between previous SE and SE to be assembled.
 - c. Defining new position and orientation with

respect to global frame by incorporating the new SE.

The assembly process includes both ideal transformation and the relevant errors.

As an example the assembly of the first two elements is performed as follows:

1. Add a revolute joint SE_1 . Since it is the first SE there is no interface error and frame {1} is referred as global frame. By adding appropriate errors we have:

$$T_{G1} = R(x,a) T(L_1,0,0) T_{e1} T_{e2} T_{e3} T_{e4} T_{e5} T_{e6}$$

where T_{G1} is the new position and orientation matrix.

2. Add SE_2 which is a link element of length L_2 . The position and orientation of link is referred with respect to T_{G1} , which is frame 2 of the previous SE. The new position and orientation matrix is given by

$$T_{G2} = T_{G1} T_{interface} T_{212} \quad (17)$$

where: T_{12}^2 - Transformation matrix from frame 1 to frame 2 for SE_2 .

T_{G2} - The new position and orientation matrix.

III. POSE ERROR, COST FUNCTIONS AND OPTIMIZATION

Pose Error Norm

The transformation matrix T defines the position and the orientation of the end point with respect to the global frame. Under ideal conditions the position of the end point is defined by the vector P, but under realistic conditions the position of the end point is defined by the vector Q. The pose error vector $R=P-Q$ is attributed to the errors in manufacturing process of the mechanism.

In general R is a 6x1 vector which contains the position and orientation errors of the pose. In this study only the position errors are considered and therefore the norm of R define the radius of a sphere within which the end effector is expected to be. In literature, the probability distribution of the shape of the error volume is arguable as spheroid. However, in the present work it is assumed as a sphere and its radius, R , is given by:

$$(R)^2 = (X - X_n)^2 + (Y - Y_n)^2 + (Z - Z_n)^2 \quad (18)$$

where X, Y, Z and X_n, Y_n, Z_n are expressions for the mechanism end point position including and excluding the manufacturing errors respectively.

The expression for R is a function of the elements' nominal dimensions, D_i , joint angles, Φ_i , and the manufacturing tolerances δ_i .

$$R = R(D_i, \Phi_i, \delta_i) \quad (19)$$

Once the expression for R is obtained, variety of analyses can be conducted: 1) Evaluation of R by substituting the relevant values for D_i , Φ_i and δ_i ; 2) Evaluation the tolerances specified for a certain SE on R by idealizing the other SEs of the mechanism. Thus, allows the designer to identify the critical SE in the mechanism; 3) Evaluating the effect of a single tolerance on R by varying its value a certain range; and 4) Evaluation of the effect of the nominal dimension on R .

Cost Function Formulation

Tolerances are functions of part size and limitations on the manufacturing process used to produce the part and have direct relationship with surface roughness produced by the process. Therefore to maintain a particular tolerance for the given part size it is necessary to select a particular manufacturing process which in turn effects its manufacturing costs as shown in figure 7. In general manufacturing cost increases exponentially for tighter tolerances.

The manufacturing cost of each element is proportional to its size and a penalty factor which depends on its tolerances. From figure 7 the penalty factor can be approximated by:

$$C_{ti} \cong 20t^{-0.35} \quad (20)$$

where: C_{ti} - Cost of the i^{th} tolerance.
 t - Tolerance variable.

As a result, the manufacturing cost, C_{SEi} , of the i^{th} SE can be expressed as:

$$C_{SEi} = KD_i(C_{t1} + C_{t2} + \dots + C_{tn}) \quad (21)$$

where: D_i - Nominal dimension of the i^{th} element.
 K - Cost constant

The total manufacturing cost of the assembly, C , will be the sum of the manufacturing cost of its elements.

Optimization criterions

Once the mechanism accuracy, R , and its cost, C , were defined, the dual optimization problems can be formulated. Both are solved using constrained optimization techniques in which an objective function is being minimized. Here, the quasi-Newton technique is adopted and IMSL subroutines are used.

Maximizing Accuracy under Cost Constraint

In this formulation the objective is to find a set of tolerances that will produce the best possible accuracy for a given cost. The objective function is formulated as:

$$I_R = R + |C - C_c|P \quad (22)$$

where: I_R - Objective function.
 C - Computed cost.
 C_c - Cost constraint
 R - Computed error volume radius.

P - Penalty factor.

Minimizing Manufacturing Cost under Accuracy Constraint

In the dual problem the objective is to find a set of tolerances which will produce a specified error volume radius at minimum cost. For this case, the objective function is defined as:

$$I_C = C + |R - R_c|P \quad (23)$$

where: I_C - New objective function.
 R - Computed error volume radius.
 R_c - Constrained error volume radius.
 C - Computed manufacturing cost.
 P - Penalty factor.

The penalty factor is selected based on the problem requirement. The main criterion in the selection of penalty constant is to keep the solution of the objective function within the feasible solution zone. It also maintains a minimum difference between computed value and specified value of the constraint.

VI. SOFTWARE DESCRIPTION

A user friendly program that assist the designer in using the above methodology was developed. The program is grouped into three modules: 1) Mechanism configuration setup; 2) Symbolic computation of R ; and 3) Optimization and Analysis. These modules are developed to perform a set of tasks and can be executed independently. However, to generate the optimization function the modules need to be executed in a sequence.

Module 1: Mechanism configuration setup.

This module, which interfaces with a library of SEs, allows the designer to select the required SEs and construct the required mechanism interactively. Once the designer selects a particular SE the corresponding

nominal and error transformation matrices are automatically incorporated in its model. An optional choice is provided during this process where ideal SE can be selected. The nominal dimensions and the manufacturing tolerances at this point are expressed in symbolic form.

Module 2: Symbolic manipulation

This module accepts the configuration determined in the module 1 and generates expressions for the actual and the nominal end point pose and for R . The symbolic language processor, "REDUCE", is used to generate these expressions. The final equation for R is expressed in symbolic form and thus:

1. Allows the user to perform sensitivity analysis since all dimensions and tolerances are expressed as variables.
2. Allows the user to evaluate cost by assigning a cost value to each tolerance.
3. Time consuming numerical computation is done at the final stage.

Module 3: Optimization and Analysis

This module is mainly devoted for simulation work. The structure of module 3 is problem oriented, and it is left to the user's discretion to modify the program according to his needs. The output from module 2 provides the designer with an expression for R which is a function of the nominal dimensions, joint angles and the manufacturing tolerances. As a first step the user has to fix the mechanism's nominal dimensions. Then, considering the its workspace or task space the designer can fix the and the joint angles. At this point R is function only of the manufacturing tolerances and accuracy sensitivity analysis can be performed. In addition, the user can assign cost to each tolerance and therefore cost sensitivity analysis can be performed. Since cost and accuracy are defined, the dual optimization problem can be solve by defining the objective functions. All these features are controlled by the user.

V. EXAMPLES

Few open chain manipulators were investigated using the above program in order to demonstrate the capability of the proposed methodology.

Case study 1: A four SE manipulator, as shown in figure 8, in which SE_3 and SE_4 assumed to be ideal was constructed. The expressions for R and C contains eleven tolerances variables. To optimize the objective functions it is necessary to provide the initial guess values and limits for all variable. Since the objective functions are nonlinear it is advisable to perform a preliminary study in which C and R are calculated for a wide range tolerance limits in order to determine initial guess that has to be within a feasible region of the solution.

In the preliminary analysis R and C were computed under the assumption the tolerances of all variables are equal and vary between 0.1 to 0.0005. The results, shown in figure 9, helps the user in selecting the initial guess values of the tolerances for the optimization. For example, for cost optimization under a error constraint of 0.5 the initial guess of the tolerances is approximately 0.005. For error optimization under cost constraint of 13500, achievable value of R is 0.5 which dictates tolerances in the order of 0.05. The results of the both optimization problems are shown in figure 10 where C_{optimum} is plotted versus R_{optimum} . As expected, the cost increases exponentially with the demand for higher accuracy.

Case study 2: In this case the same manipulator as in the previous case is considered but SE_1 and SE_2 are assumed ideal.

The results of this optimization is presented in figure 11. As expected, in comparison to the previous example, the results indicate that higher accuracy can be reached for less cost this is due elimination of error magnification caused

by 'boom effect' of SE_2 and SE_4 .

Case study 3: In this case a manipulator, shown in figure 12, in which SE_1 and SE_2 (ideal) are rotational and SE_3 is a prismatic joints, is considered. In the optimization process twelve tolerance variables, varied from 0.1 to 0.001, were evaluated and the results are given in Table 1. As shown, for low accuracy requirement only few tolerances were changed and most remained at their maximum value of 0.1. As higher accuracy is imposed, tighter tolerances are specified and the corresponding cost increases.

VI. CONCLUSIONS

In the present research work an evaluation tool has been developed by which the interrelation between the manufacturing tolerances, mechanism accuracy and manufacturing cost can be investigated. A new kinematic error model for open-chain mechanisms which incorporates manufacturing tolerances has been developed. The software package developed is user friendly and allows the user to configure the mechanism interactively and optimize its design for cost and accuracy.

VII. REFERENCES

1. B.W. Mooring, Zvi S. Roth and Morris R. Driels, *Fundamentals of Manipulator Calibration*, New York: J.Wiley, 1991.
2. Earlwood T. Fortini, *Dimensioning for interchangeable manufacturing*, New York, Industrial press, 1967.
3. Hanqi Zhuang and Zvi S. Roth, "A unified approach to kinematic modeling, Identification and Compensation for robot calibration," *Control and Dynamic systems*, Vol.39; 1991.
4. Anthony C. McDonald, *Robot Technology Theory, Design and Applications*, Englewood cliffs, N.J.: Prentice-Hall, 1986.
5. R.P. Paul, *Robot Manipulators:*

- Mathematics, Programming, and Controls, MIT Press, 1982.
6. Oren Masory, "Improving Robot Positioning Accuracy by local Interpolation," Proc. NAMRC XVI, Urbana, IL, May 1988.
 7. Zvi S. Roth, B.W. Mooring and B. Ravani, "An overview of Robot Calibration," IEEE Journal of Robotics and Automation, Vol. 5, Oct 1987.
 8. C.H. Wu, "A kinematic CAD tool for the design and control of robot manipulators," International Journal of Robotics Research, Vol. 3, 1984.
 9. W.K. Veitshegger and C.H. Wu, "Robot Accuracy Analysis Based on Kinematics," IEEE Journal of Robotics and Automation, Vol. RA-2, No. 3, Sep. 1986.
 10. B.W. Mooring, "The effect of joint axis misalignment on Robot Positioning Accuracy," Proc. of the ASME Computers in Engineering Conf., Vol. 2, 1983.
 11. B.W. Mooring and G.R. Tang, "An improved method for Identifying the Kinematic Parameters in a six axis Robot," Proc. ASME Computers in Engineering Conf., Vol. 1, 1984.
 12. Tyang Liu, "Error Analysis of robot manipulators with manufacturing tolerances," 1st National, Applied mechanisms and Robotics Conf., Nov. 5-8, Vol. 1, 1989.
 13. R.G. Fenton, W.L Cleghorn, Jing-Fan Fu, "Allocation of dimensional tolerances for multiple loop Planar Mechanism," Journal of Mechanisms, Transmissions and Automation in Design, Dec. 1989.
 14. E. Ramsli, "Probability Distribution of Repeatability of Industrial Robots," ASME ,WAM, DSC Vol.14, Dec. 1989.
 15. Woo-Jong Lee, T.C. Woo, "Tolerance Volume Due to joint Variable Errors in Robots," Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 111/597, Dec. 1989.
 16. Placid M. Ferreira, C. Richard Liu, "An Analytical Quadratic Model for the Geometric Error of a Machine Tool," Journal of Manufacturing Systems, Volume 5/No. 1, 1989.
 17. W.J. Lee, T.C. Woo, "Optimum Selection of Discrete Tolerances," Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 111/243, June 1989.
 18. S.G. Dhande, J.Chakraborty, "Analysis and Synthesis of Mechanical Error in Linkages - A Stochastic Approach," Trans. of the ASME Jour. of Engineering for Industry, Aug 1973.
 19. O.M.A. Sharfi and M.R. Smith, "A Simple method for the Allocation of Appropriate Tolerances and Clearances in Linkage Mechanisms," Mechanism and Machine Theory, Vol. 18, No. 2, 1983.

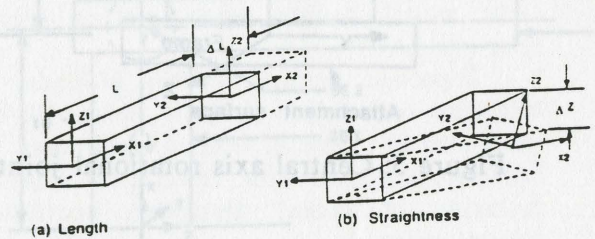


Figure 1: Tolerances for a link element.

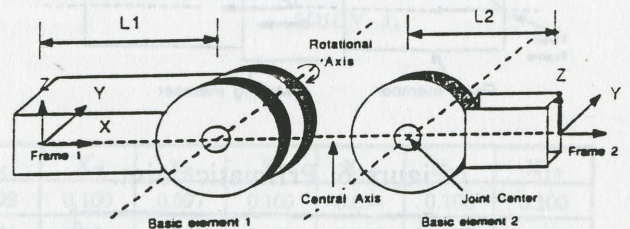


Figure 2: Revolute joint geometry.

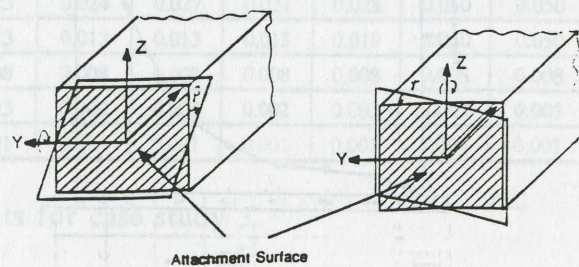


Figure 3: Rotation of attachment surfaces.

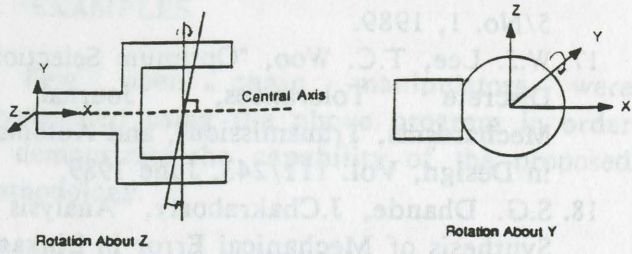


Figure 4: Perpendicularity of rotational axes.

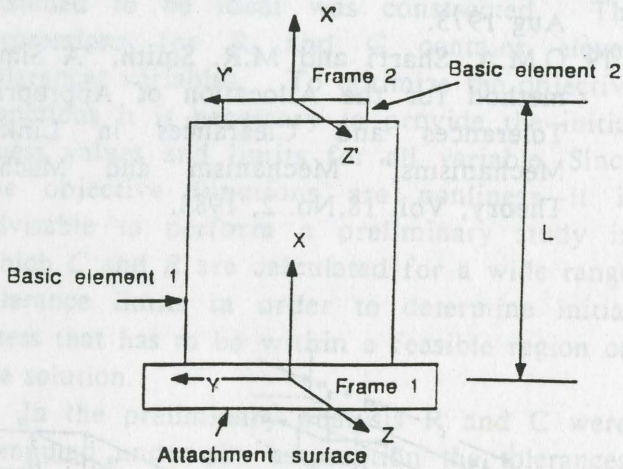


Figure 5: Central axis rotational joint.

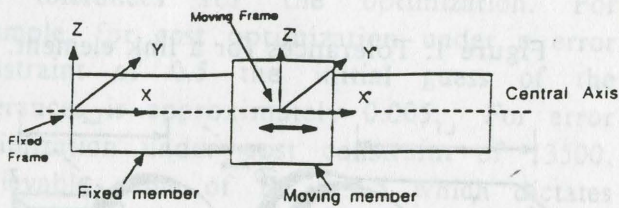


Figure 6: Prismatic joint.

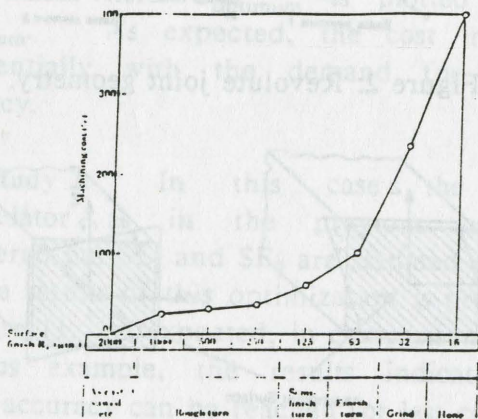


Figure 7: Tolerance and cost relationship.

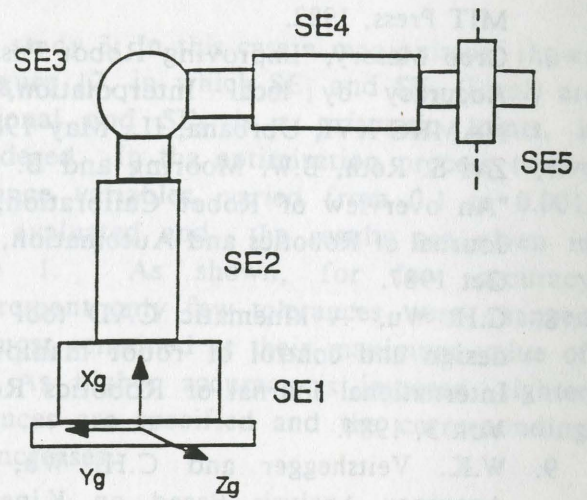


Figure 8: Manipulator configuration for case study 1 and case study 2

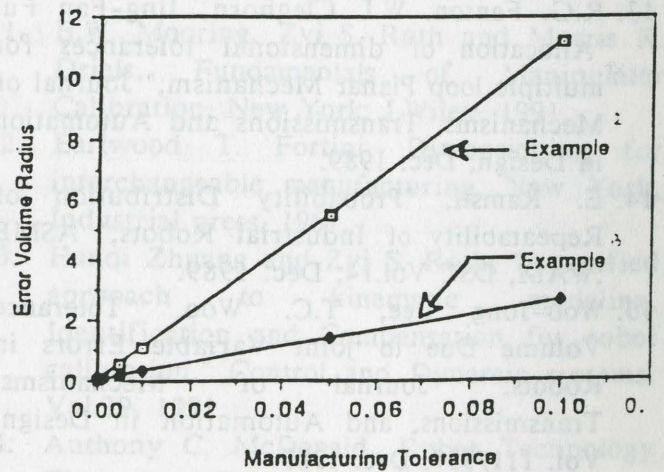
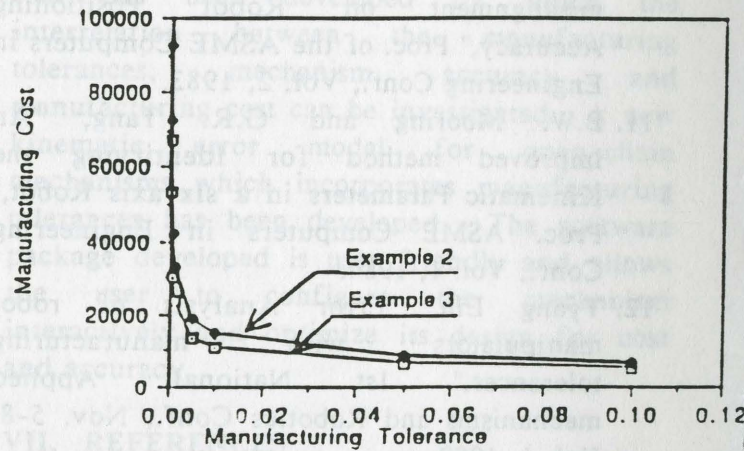


Figure 9: Preliminary calculation of C and R for case study 1.

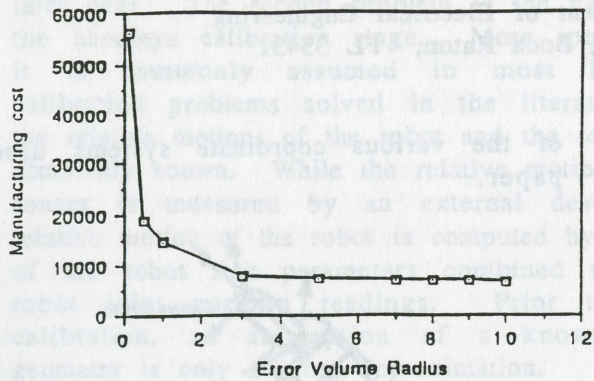


Figure 10: Optimization results for case study 1.

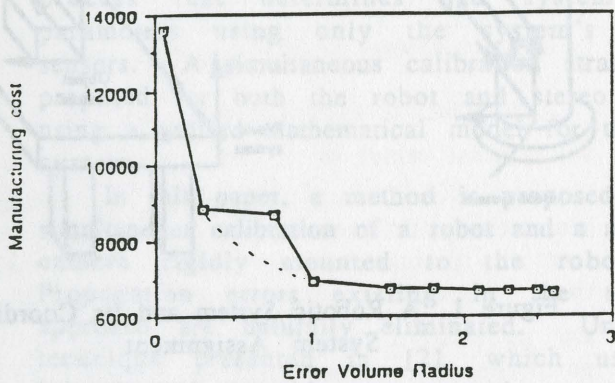


Figure 11: Optimization results for case study 2.

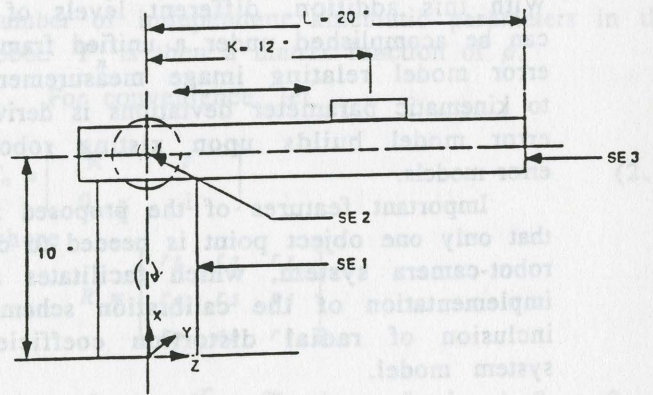


Figure 12: Manipulator configuration for case study 3.

R	C	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇	X ₈	X ₉	X ₁₀	X ₁₁	X ₁₂
8.0	11000	0.100	0.100	0.100	0.081	0.086	0.098	0.100	0.097	0.100	0.094	0.100	0.100
5.0	12690	0.045	0.051	0.044	0.053	0.052	0.054	0.051	0.057	0.100	0.100	0.100	0.100
4.0	13360	0.043	0.048	0.041	0.051	0.050	0.053	0.049	0.054	0.049	0.054	0.100	0.100
2.5	15919	0.028	0.031	0.028	0.031	0.034	0.030	0.032	0.031	0.031	0.031	0.050	0.050
2.0	16983	0.023	0.026	0.021	0.025	0.023	0.025	0.024	0.027	0.024	0.028	0.050	0.050
1.0	20881	0.010	0.013	0.010	0.013	0.013	0.013	0.013	0.013	0.013	0.019	0.050	0.050
0.65	26137	0.007	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008
0.2	19199	0.003	0.002	0.003	0.003	0.003	0.003	0.002	0.002	0.002	0.002	0.003	0.003
0.06	52890	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001

Table 1: Optimization results for case study 3.

Implementation Issues on Simultaneous Calibration of a Robot and a Hand-Mounted Camera

Hanqi Zhuang, Luke Wang and Zvi S. Roth
Robotics Center and Department of Electrical Engineering
Florida Atlantic University, Boca Raton, FL 33431

Abstract

In this paper, a method is presented for simultaneous calibration of a robot and a hand-mounted monocular camera. Unlike conventional approaches based on first calibrating the camera and then calibrating the robot, the algorithm solves for the kinematic parameters of the robot and camera in one stage, thus eliminating error propagation and improving noise sensitivities. Only two extra parameters are added to the robot calibration model to represent the camera geometry. With this addition, different levels of calibration can be accomplished under a unified framework. An error model relating image measurement residuals to kinematic parameter deviations is derived. This error model builds upon existing robot accuracy error models.

Important features of the proposed approach is that only one object point is needed to calibrate the robot-camera system, which facilitates autonomous implementation of the calibration scheme, and the inclusion of radial distortion coefficient in the system model.

1 Introduction

Visual sensing is an important aspect of an intelligent robotic system. A popular system configuration, which has been widely used in various robotic applications, is to mount a camera on the hand of a robot manipulator.

In order to accurately measure the position and orientation of an object in a reference ("world") coordinate system by the robot-camera system, various components in the system have to be calibrated. This includes the determination of the pose (that is, the relative position and orientation) of the robot base with respect to the robot world, the robot hand with respect to the robot base, the camera with respect to the robot hand, and the object with respect to the camera. These four tasks are respectively known as robot base calibration, robot manipulator calibration (in short robot calibration), hand/eye calibration and camera calibration. Robot base calibration can sometimes be considered as a sub-task of robot calibration. Readers are referred to Figure 1 for the illustration

of the various coordinate systems used in this paper.

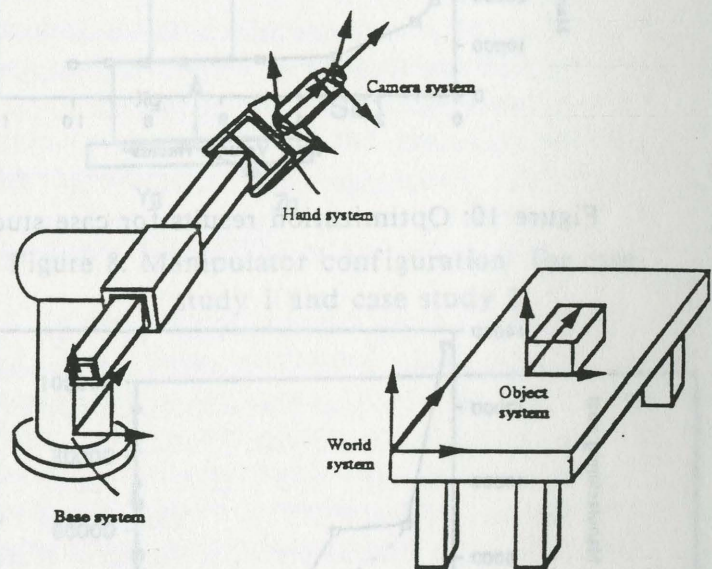


Figure 1 A Robotic System and Its Coordinate System Assignment

There has been an extensive research aimed at solving each of the above tasks. Conventionally, each of these tasks is handled individually. One may start by calibrating the camera to determine the relative pose between the object and the camera, follow by calibrating the hand/camera to determine the pose between the robot hand and the camera, and then use the camera and hand/eye models to calibrate the robot to determine the pose of the robot hand in the robot world coordinate system. After all system components are individually calibrated, the pose of an object in the robot world system can be determined.

Such a multistage approach has two main advantages. First, since system calibration is performed by calibrating its components or subsystems separately, each component calibration task is relatively simple. Second, if some of the system components have changed their location or parameters, calibration needs only to be repeated for these system components. For example, if the camera changed its focal length, only the camera

needs to be recalibrated.

The multi-stage approach, however, has some drawbacks. The first problem is that parameter estimation errors in early stages propagate to the later ones. The second problem is the validity of the hand/eye calibration stage. More specifically, it is commonly assumed in most hand/eye calibration problems solved in the literature that the relative motions of the robot and the sensor are accurately known. While the relative motion of the sensor is measured by an external device, the relative motion of the robot is computed by the use of the robot link parameters combined with the robot joint position readings. Prior to robot calibration, an assumption of a known robot geometry is only a gross approximation.

An early experimental study of robot calibration using stereo cameras rigidly mounted to the robot hand is [1]. In [2], the concept of *autonomous calibration* was defined as an automated process that determines the system model parameters using only the system's internal sensors. A simultaneous calibration strategy was proposed for both the robot and stereo cameras using a unified mathematical model for the entire system.

In this paper, a method is proposed for the simultaneous calibration of a robot and a monocular camera rigidly mounted to the robot hand. Propagation errors existing in the two-stage approach are naturally eliminated. Unlike the technique presented in [2], which used two instrumented movable cameras, the new method employs a single passive camera. By using a single camera, the field of view of the camera is larger than that of using stereo cameras, and consequently the measurable workspace of the robot manipulator is relatively larger. An additional feature of the new method is the inclusion of a radial camera lens distortion parameter, which is the dominant factor in lens distortion.

2 Kinematic Model and Cost Function

The basic geometry of the system is shown in Fig. 1. $\{x_w, y_w, z_w\}$ denotes the world coordinate system, which is normally at a convenient location outside the robot and camera. $\{x_b, y_b, z_b\}$ denotes the base coordinate system of the robot, physically located at the base of the robot. $\{x, y, z\}$ denotes the camera coordinate system, whose origin is at the optical center point O , and whose z axis coincides with the optical axis. $\{X, Y\}$ denotes the image coordinate system (not shown in Fig. 1) centered at

O_I (the intersection of the optical axis z and the image plane). $\{X, Y\}$ lies on a plane parallel to the x and y axes.

Let the 4×4 homogeneous transformation T_n relating the end-effector pose to the world coordinates be $T_n = A_0 A_1 \dots A_{n-1} A_n$, where A_0 is the 4×4 homogeneous transformation from the world coordinate system to the base coordinate system of the robot, A_i is the transformation from the $(i-1)$ th to the i th link coordinate systems of the robot, and A_n is the transformation from the n th link coordinate system of the robot to the camera coordinate system.

A_i can be represented in terms of any proper kinematic modeling convention. Let $\rho = [\rho_1 \ \rho_2 \ \dots \ \rho_p]^T$ be the kinematic parameter vector consisting of all link parameters of the robot, where p is the number of independent kinematic parameters in the robot. T_n is then a matrix function of ρ .

For convenience, let

$$T_n = \begin{bmatrix} R & t \\ 0_{1 \times 3} & 1 \end{bmatrix} \quad (2.1)$$

where

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}$$

and $t \equiv [t_x \ t_y \ t_z]^T$. Clearly, r_i for $i = 1, 2, \dots, 9$ and t_i for $i = x, y, z$ are all functions of the robot kinematic parameter vector ρ .

The camera-robot model, relating the world coordinate system $\{x_w, y_w, z_w\}$ to the image coordinate system $\{X, Y\}$, is [3]

$$X(1 + \alpha(sX^2 + Y^2)) = s_x f \frac{r_1 x_w + r_2 y_w + r_3 z_w + t_x}{r_7 x_w + r_8 y_w + r_9 z_w + t_z} \quad (2.2a)$$

$$Y(1 + \alpha(sX^2 + Y^2)) = s_y f \frac{r_4 y_w + r_5 y_w + r_6 z_w + t_y}{r_7 x_w + r_8 y_w + r_9 z_w + t_z} \quad (2.2b)$$

where f is the focal length of the camera, s_x and s_y are the camera scale factors, $s = s_y/s_x$, and α is the radial lens distortion coefficient. If s_x and s_y together with f are treated as unknowns, one should bear in mind that these parameters are not independent. Thus at most two parameters from the set $\{s, s_x, s_y, f\}$ need to be identified. Define

$$f_x \equiv f s_x \quad (2.3a)$$

$$f_y \equiv f s_y \quad (2.3b)$$

$$f_{xy} \equiv sX^2 + Y^2 \quad (2.3c)$$

(2.2) is then rewritten as

$$X(1 + \alpha f_{xy}) = f_x \frac{r_1 x_w + r_2 y_w + r_3 z_w + t_x}{r_7 x_w + r_8 y_w + r_9 z_w + t_z} \quad (2.4a)$$

$$Y(1 + \alpha f_{xy}) = f_y \frac{r_4 y_w + r_5 y_w + r_6 z_w + t_y}{r_7 x_w + r_8 y_w + r_9 z_w + t_z} \quad (2.4b)$$

As seen in (2.1) and (2.4), the extrinsic parameters of the camera are all absorbed into the pose of the robot.

The problem of simultaneous calibration of robot and camera can be stated as follows: Given a number of calibration points whose world coordinates are known and whose image coordinates are measured, estimate the robot kinematic parameter vector ρ and the camera parameters f_x , f_y , and α .

In order to apply optimization techniques to solve the calibration problem, a cost function needs to be constructed.

Define a 16x1 vector ϕ in the following manner:

$$\phi(\rho, f_x, f_y, \alpha) \equiv [f_x r_1 \quad f_y r_4 \quad r_7 \quad \alpha r_7 \quad f_x r_2 \quad f_y r_5 \quad r_8 \quad \alpha r_8 \quad f_x r_3 \quad f_y r_6 \quad r_9 \quad \alpha r_9 \quad f_x t_x \quad f_y t_y \quad t_z \quad \alpha t_z]^T \quad (2.5a)$$

and a 2x16 matrix C as follows:

$$C \equiv \begin{bmatrix} x_w & 0 & -x_w X & -f_x y x_w X & y_w & 0 & -y_w X & -f_x y y_w X \\ 0 & x_w & -x_w Y & -f_x y x_w Y & 0 & y_w & -y_w Y & -f_x y y_w Y \\ z_w & 0 & -z_w X & -f_x y z_w X & 1 & 0 & -X & -f_x y X \\ 0 & z_w & -z_w Y & -f_x y z_w Y & 0 & 1 & -Y & -f_x y Y \end{bmatrix} \quad (2.5b)$$

(2.4) can then be rewritten in the following form

$$C \phi(\rho, f_x, \alpha) = 0 \quad (2.6)$$

Note that the dependence of ϕ on f_y is omitted as f_y can be recovered from f_x using (2.3c).

In (2.6), C is a known coefficient matrix, assuming that the scale factor s is determined in advance. Readers are referred to [4] for methods of determining the scale factor. ϕ is a vector function of the unknown parameter vector. According to (2.4), one calibration point can only provide two scalar equations. To estimate all unknown parameters, a sufficient number of calibration

points have to be used. Let C_i , whose structure is given in (2.5b), denote the computed coefficient matrix using the i th calibration point. Let m be the number of points used for parameter estimation. The problem is then reduced to determining ρ, f_x, α that minimize the cost function E in a least squares sense, where

$$E = \sum_{i=1}^m [C_i \phi(\rho, f_x, \alpha)]^T [C_i \phi(\rho, f_x, \alpha)] \quad (2.7)$$

An iterative procedure is needed to obtain the optimal solution. The Gauss-Newton method for solving non-linear least square problems has the advantage of fast convergence in the neighborhood of a solution. A modification of this algorithm is the more robust Levenberg-Marquardt [5] algorithm. To properly apply the Gauss-Newton algorithm to the problem at hand, the following issues have to be addressed:

1. Choice of an initial condition
2. The structure of the Jacobian matrix.

A practical assumption is that ρ^0, f_x^0 and α^0 , the nominal values of ρ, f_x and α are known. In the case of robot-camera calibration, the nominal robot kinematic parameter vector ρ^0 is usually provided from the design drawings of the robot, except for those parameters associated with A_0 and A_n , which are application dependent and can be roughly determined by proper gauging devices. The initial values of the scale factors s_x and s_y can be set by the camera specifications. The nominal value of the distortion coefficient α is set to zero. The nominal value of the focal length may be read from the lens.

3 The Identification Jacobian

The Identification Jacobian is a Jacobian matrix relating measurement residuals to parameter errors $d\rho, df_x$ and $d\alpha$. It will be shown that the original robot identification Jacobian, derived by many robot researchers (for instance, refer to [6,7], will be used as one of the building blocks of the robot/camera Identification Jacobian.

Prior to the derivation of the Identification Jacobian, let us introduce a more compact notation. A function $vec(X)$ will denote a vector valued function of a matrix X, resulting from stacking the matrix columns one on top of the other, first column on top, second column right under and so on. The notation $(X)_{ij}$ is used to represent the first i rows and j columns of a matrix X. Thus, ϕ in (2.5a) can

be compactly rewritten as

$$\phi = \text{vec}(FT_n) \quad (3.1)$$

where

$$F \equiv \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \alpha & 0 \end{bmatrix} \quad (3.2)$$

Assume initially that the nominal vector ϕ^0 deviates from ϕ^* , the optimal solution by a small amount. Thus,

$$\phi^* \approx \phi^0 + d\phi \quad (3.3)$$

where $d\phi$ is a differential change of ϕ . To simplify the notation, from now on, the superscript 0 may be omitted if no confusion arises. By (3.2),

$$\begin{aligned} d\phi &\approx \text{vec}(dF T_n + F dT_n) \\ &= \text{vec}(dF T_n) + \text{vec}(F dT_n) \end{aligned} \quad (3.4)$$

Define the vectors

$$J_x \equiv [r_1 \ 0 \ 0 \ 0 \ r_2 \ 0 \ 0 \ 0 \ r_3 \ 0 \ 0 \ 0 \ t_x \ 0 \ 0 \ 0]^T \quad (3.5a)$$

$$J_y \equiv [0 \ r_4 \ 0 \ 0 \ 0 \ r_5 \ 0 \ 0 \ 0 \ r_6 \ 0 \ 0 \ 0 \ t_y \ 0 \ 0]^T \quad (3.5b)$$

$$J_\alpha \equiv [0 \ 0 \ 0 \ r_7 \ 0 \ 0 \ 0 \ r_8 \ 0 \ 0 \ 0 \ r_9 \ 0 \ 0 \ 0 \ t_z]^T \quad (3.5c)$$

By simple algebraic manipulations, the first term in the right hand side of (3.4), representing the contribution to the error model by the camera intrinsic error parameters, can then be written as

$$\text{vec}(dF T_n) = (J_x + sJ_y)df_x + J_\alpha d\alpha \quad (3.6)$$

More manipulations need to be performed to expand the second term in the right hand side of (3.4), representing the contribution to the error model by the robot kinematic error parameters. A main objective is to allow robot calibration researchers and practitioners to retain the original robot error models, and hence a large portion of the kinematic identification software. Define

$$\Delta \equiv T_n^{-1} dT_n \quad (3.7)$$

Δ , a 4x4 matrix, has the following structure,

$$\Delta = \begin{bmatrix} \Omega(\delta) & d \\ 0_{1 \times 3} & 0 \end{bmatrix} \quad (3.8)$$

where $\delta \equiv [\delta_x, \delta_y, \delta_z]^T$ and $d \equiv [d_x, d_y, d_z]^T$ are respectively the 3x1 rotational and positional error vectors of T_n , and $\Omega(\delta)$ is a skew-symmetric matrix,

$$\Omega(\delta) = \begin{bmatrix} 0 & \delta_z & -\delta_y \\ -\delta_z & 0 & \delta_x \\ \delta_y & -\delta_x & 0 \end{bmatrix}$$

Two steps are now needed to relate $\text{vec}(FdT_n)$ to the robot kinematic parameter error vector $d\rho$. First, $\text{vec}(FdT_n)$ is related to the pose error vector $[d^T, \delta^T]^T$ by a linear transformation. $\text{vec}(FdT_n)$ is then related to $d\rho$ by using an additional linear transformation relating $d\rho$ to $[d^T, \delta^T]^T$.

By (3.7),

$$FdT_n = FT_n \Delta \quad (3.9)$$

F is now decomposed into

$$F \equiv GH \quad \text{where} \quad G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \alpha & 0 \end{bmatrix} \quad (3.10a)$$

$$H = \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.10b)$$

Then from (3.9),

$$\text{vec}(FdT_n) = \text{vec}(FT_n \Delta) = \text{vec}(GHT_n \Delta) \quad (3.11a)$$

A crucial step in the derivation of the robot/camera error model is to invoke the Kronecker product of matrices. That is,

$$\text{vec}(GHT_n \Delta) = (I_{3 \times 3} \otimes G) \text{vec}(HT_n \Delta) = (I \otimes G) \begin{bmatrix} \text{vec} \begin{pmatrix} H_{3:3} R \Omega(\delta) \\ 0_{1 \times 3} \end{pmatrix} \\ H_{3:3} R d \\ 0 \end{bmatrix} \quad (3.11b)$$

where \otimes denotes a Kronecker product of two matrices [8].

In the error-model based robot calibration literature, the transformation from $d\rho$ to $[d^T, \delta^T]^T$ is available. More specifically, one has

$$\delta = \sum_{j=1}^P J_{\delta j} d\rho_j \quad (3.12a)$$

$$d = \sum_{j=1}^P J_{d j} d\rho_j \quad (3.12b)$$

where $J_{\delta i}$ and $J_{d i}$ are 3×1 vectors. The details of $J_{\delta i}$ and $J_{d i}$ in terms of a specific kinematic modeling convention can be found in the robot calibration literature. After substituting (3.12) into (3.11) and with some algebraic manipulations, one obtains

$$v e c(F d T_n) = (I \otimes G) \sum_{j=1}^P \begin{bmatrix} v e c \left(\begin{matrix} H_{3:3} R \Omega(J_{\delta j}) \\ 0_{1 \times 3} \\ H_{3:3} R J_{d j} \\ 0 \end{matrix} \right) \end{bmatrix} d\rho_j \quad (3.13)$$

Replacing ϕ in (2.6) by ϕ^* of (3.3) yields

$$C_i(\phi + d\phi) = 0 \quad i = 1, 2, \dots, m. \quad (3.14)$$

where again the superscript 0 is omitted. Substituting (3.6) and (3.13) into (3.4) and then substituting the result into (3.14) yields

$$C_i \phi = -C_i J_i d\alpha - C_i (V_x + V_y) d f_x - C_i (I \otimes G) \sum_{j=1}^P \begin{bmatrix} v e c \left(\begin{matrix} H_{3:3} R \Omega(J_{\delta j}) \\ 0_{1 \times 3} \\ H_{3:3} R J_{d j} \\ 0 \end{matrix} \right) \end{bmatrix} d\rho_j \quad (3.15)$$

$i = 1, 2, \dots, m.$

Let J_i be the $2 \times (p+2)$ coefficient matrix of (3.15), and

$$d\rho^{aug} \equiv [d\alpha \quad d f_x \quad d\rho^T]^T. \quad (3.16)$$

Then (3.15) can be rewritten as

$$C_i \phi = J_i d\rho^{aug} \quad i = 1, 2, \dots, m. \quad (3.17)$$

(3.17) provides the relationship between the measurement residual error vector $C\phi$ and the augmented parameter error vector $d\rho^{aug}$. J is termed the *Identification Jacobian* for robot-camera calibration.

4 Implementation Issues

A. Robot Parameters

A sufficient number of independent link parameters have to be used to express any variation

of the actual robot structure away from the nominal design. This number, for a serial manipulator consisting of rigid links connected by low pair joints, is $4N - 2P + 6$, where N is the number of degrees of freedom and P is the number of prismatic joints [9]. For example, in a PUMA 560 robot, the number of link parameters is 30. In other words, the dimension of the PUMA kinematic parameter vector ρ is 30.

Not all parameters in ρ need to be calibrated each time. It is very natural to accommodate different complexities of calibration with the approach proposed above. Next we discuss three levels of calibration.

As has been mentioned, the simplest level of calibration is to identify the camera parameters α , f_x and f_y together with the parameters that specify the hand/eye transformation A_n , assuming that the transformation from the world coordinate system to the robot hand coordinate system is known. This type of calibration is necessary whenever the relative pose of the camera with the robot is changed. We use six link parameters to specify the transformation A_n . Together with the two camera parameters, there are eight parameters to be estimated. That is, the dimension of $d\rho^{aug}$ given in Equation (3.16) is 8.

The second level of calibration is to identify the camera parameters together with the parameters that specify the hand/eye transformation A_n and the base/world transformation A_0 , assuming that the robot geometry is accurately known. This type of calibration is necessary whenever the camera changes its location with respect to the robot hand and the robot also changes its location with respect to an external reference object. Since four additional parameters in A_0 need to be identified, the dimension of ρ^{aug} is 12.

The third level, which is the most general, is to calibrate the entire robot-camera system. In this case, the dimension of ρ^{aug} is $4N - 2P + 8$, among which two are camera parameters.

B. Change of Reference Frame

$[d^T, \delta^T]^T$, the pose error vector of T_n defined in Section 3, is represented in the world coordinate system since T_n is defined as the transformation from the world coordinate system to the camera coordinate system. The robot error model given in Equation (3.12) shall be consistent with this convention. If the pose error vector is represented in the camera coordinate system, as has been the

case in some robot kinematic error models, the expression given in Equation (3.16a) has to be modified to accommodate the change of the reference coordinate system.

The transformation from the camera coordinate system to the world coordinate system is T_n^{-1} . Let d' and δ' be respectively the rotational and positional error vectors of T_n^{-1} . Also let

$$T_n^{-1} \equiv \begin{bmatrix} R' & t' \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

Then it can be shown that

$$\delta = -R^T \delta' \quad (4.1a)$$

$$d = -R \Omega(t') \delta' - R d' \quad (4.1b)$$

δ and d in Equation (3.11b) shall be substituted by Equation (4.1) whenever the robot error model (i.e. J_{δ_i} and J_{d_i}) is given in terms of the camera coordinate frame. Equations (3.12), (3.13), (3.15) and (3.16) shall also be modified accordingly.

C. Observability of the Unknown Parameters

In the robot calibration literature, the observability of the kinematic error parameter vector $d\rho$ is defined in terms of the Identification Jacobian. If the Identification Jacobian is full rank, the error parameter vector is said to be observable.

It is difficult to obtain analytical observability results since the structure of the Jacobian matrix in this case is very complex (refer to Equation (3.16a)). However, the Identification Jacobian derived in this paper can be used for optimal off-line search of robot measurement configurations, which can significantly improve calibration quality. Borm and Menq [10] defined observability measure for robot calibration using all singular values of the Identification Jacobian. Experimental studies were performed which demonstrated that a low number of well-selected measurements can relatively produce superior results, compared with identification done based on a set of large number of randomly selected measurement configurations.

D. Verification of the Calibration Results

It is difficult to obtain highly accurate

measurement set-up to serve as reference against which the accuracy performance of a robot-camera calibration task is assessed. In the absence of an accurate external reference device, one may use the following two approaches.

Approach I: 2D Image Plane Verification.

Assume that f_x , f_y , and ρ have been identified.

A set of robot configurations, which are different from those used for identification, is given. A set of world coordinates of the calibration points in each robot configuration is also given. The coordinates of the corresponding image points are measured. Using the identified camera and robot parameters together with the robot joint variables at each configuration, and the world coordinates of the calibration points, one can compute the predicted image coordinates of each image point (refer to Equation (2.2)). The Euclidian norm of the difference between the measured and computed image coordinates at each image point can be defined as a *2D calibration error*. 2D calibration errors are computed for all image points at all robot configurations, and finally the mean and standard deviation of the 2D calibration errors are computed.

Approach II: 3D World Coordinates Verification.

One may compute 3D world coordinates of each calibration point using the calibrated robot and camera parameters. This is possible by using more than one view, that is more than one robot configuration, of the same calibration point. Two views of an identical point are sufficient to compute its world coordinates using stereo triangulation. Using more than two views calls for a least squares fitting. The Euclidian norm of the difference between the computed world coordinates of the calibration point and its given world coordinates can be defined as a *3D calibration error*. One can compute the mean and standard deviation of the 3D calibration errors by repeating this procedure for all calibration points.

E. The use of a Single Calibration Point to Calibrate the Robot-Camera System

The proposed approach allows the user to use a single point to calibrate the system. A precision ball is placed at a location visible by the camera. The camera is moved by the robot arm. Since the camera can capture the image of the ball from any angles within the robot workspace, the variety of available camera orientations is very large. By mounting the ball on a rail, a x-y table, or a coordinate measuring machine, the measurable

robot workspace can be made even larger. This advantage is significant in real applications.

The minimum number of robot configurations needed for the calibration of the robot-camera system depends on 1) the number of unknown parameters, and 2) the number of calibration points measured at each camera snap shot. If at each robot measurement configuration only one image point is measured, (i.e., only one point is viewed by the camera), the minimum number of measurement configurations is approximately half of the number of parameters to be identified since each image point provides two equations. As the number of points measured in each configuration is increased, the number of configurations needed can be accordingly decreased.

Some simple remarks on the observability of the robot/camera parameters can be made for the set-up in which a single calibration point is used.

Case 1: The image of the calibration point is fixed at the image plane.

In the case that the image of the calibration point is fixed at the center of the image plane, $(X, Y) = (0, 0)$, and the matrix C given in Equation (2.5b) is reduced to the following form,

$$C = \begin{bmatrix} x_w & 0 & 0 & 0 & y_w & 0 & 0 & 0 & z_w & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & x_w & 0 & 0 & 0 & y_w & 0 & 0 & 0 & z_w & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The first column of the Identification Jacobian given in Equation (3.16a) is thus zero, regardless of the change of robot configurations. Consequently, the Jacobian is singular, therefore the robot/camera parameters are not observable.

Rather than fixing the calibration point at the center of the image, one may fix it somewhere else in the image plane. In such a case, the rank of the Identification Jacobian should be checked numerically.

Case 2: The calibration point is fixed.

If the fixed calibration point is defined as the origin of the world coordinate system, $(x_w, y_w, z_w) = (0, 0, 0)$. In this case, the Identification Jacobian is not full rank, which can be easily shown following the same procedure given in Case 1.

One may artificially shift the origin of the world coordinate system from the calibration point to another location. Again the rank of the Identification Jacobian can be checked by a numerical approach.

5. Experimental Results

The experiment system consisted of a Puma

560 robot, a CCD camera (Electrophysics, model number CCD1200), a 486 personal computer, an ITEX video imaging board with its driver software, a camera calibration board, and a coordinate measuring machine (CMM). The CMM was used to move the calibration board to different locations.

The CCD camera has 510H by 492V picture elements with 8.8x6.6 mm² sensing area. The camera calibration board is a glass plate painted with vapor deposited metallic chromium. It has 10x10 dot array points with center to center distance of 10 mm, and diameter (of each point) of 2 mm. The flatness of the calibration board is within ±0.003 mm and the center to center accuracy of the calibration points is within ±0.002 mm.

The vision algorithms for camera calibration were written in Microsoft-C. To estimate the image coordinates of a calibration point, an adaptive thresholding algorithm was devised. The algorithm first smoothed an image of the camera calibration board by repeatedly applying a 3x3 low-pass mask (three times). It then detected consecutively each calibration point, and computed the histogram within a window surrounding each point. The intersection of two Gaussian curves which fit the histogram was chosen as the threshold value for the calibration point. The centroid of the image of calibration point was selected as the estimate of its image coordinates. This procedure could yield image coordinates accuracy to within 1/5 of a pixel.

The modified CPC modeling convention [11] was used to represent the geometry of the Puma robot.

Performance comparison between the one-stage algorithm and a two-stage algorithm was conducted through experimentation. In the two-stage algorithm, Tsai's camera calibration technique was used to compute robot poses at each robot measurement configuration. Sample results are shown in Figure 2.

6. Conclusions

Calibration is at the very heart of creating a truly off-line programming environment for robots. Autonomous calibration of robot-camera system is of particular importance for robots that must function outside a controlled laboratory environment [2]. This paper presented a method for robot-camera system calibration which can be made autonomous. It has the following features:

1. The propagation errors which exist in multi-stage approaches are eliminated. This is reflected in the experimental results that clearly demonstrate the superiority of the one-

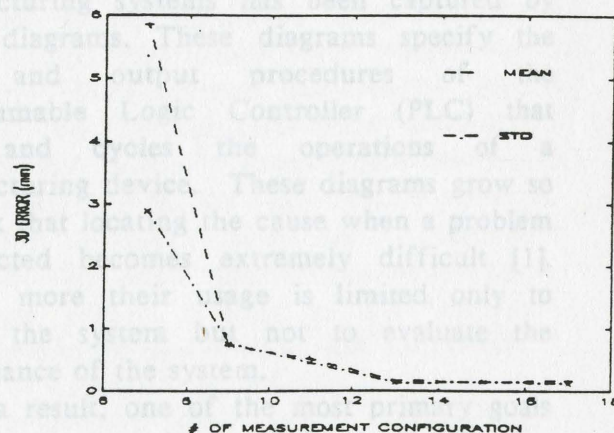
- stage approach over a two-stage approach in terms of accuracy performance.
2. The scheme uses a monocular camera, thus the field-of-view of the camera is larger than that of stereo cameras. Moreover, the processing speed of one camera system is faster.
3. Only two additional parameters are added to a robot calibration model to represent camera geometry in the system model.
4. Simulation and experimental results show that the proposed algorithm converges in very few iterations, as a good set of initial conditions for the robot and camera parameters is usually available.
5. The approach can be automated as only a single point needs to be tracked in space. A research is under way to automate the calibration process based on the proposed approach.

The major limitations of the proposed approach are:

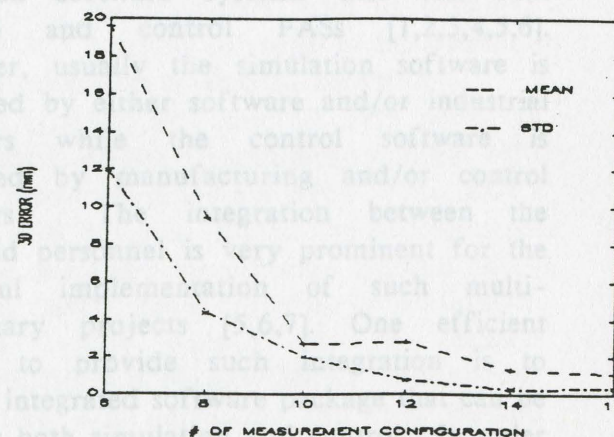
1. Since only one camera is used, the system cannot recover the absolute distance information of the scene.
2. If a tool is attached to the robot hand, the transformation from the tool to the camera has to be determined separately. This problem exists in all robotic systems equipped with a hand/eye configuration.

References

- [1] Puskorius, G.V. and L.A. Feldkamp, "Global Calibration of a Robot/Vision System," Proc. Int. Conf. on IEEE Robotics & Automation, Raleigh, NC, 1987, pp. 190-195.
- [2] Bennett, D. J. D. Geiger, and J. M. Hollerbach, "Autonomous Robot Calibration for Hand-Eye Coordination," *Int. J. Robotics Research*, Vol. 10, No. 5, Oct. 1991.
- [3] Tsai, R. Y., "A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE J. Robotics & Automation*, Vol. RA-3, No. 4, Aug. 1987 pp. 323-344.
- [4] Lenz, R. K and R. Tsai, "Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3D Machine Vision Metrology," *Proc. IEEE Int. Conf. Robotics & Automation*, 1987, pp. 68-75.
- [5] Marquardt, D. W. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *J. Soc. Indust. Appl. Math.*, Vol. 11, No. 2, June, 1963, pp. 431-441.
- [6] Wu, C.H., "A Kinematic CAD Tool for the Design and Control of Robot Manipulator," *Int. J. Robotics Research*, Vol. 3, No. 1, pp. 58-67.
- [7] Zhuang, H and Zvi S. Roth, "Robot Calibration Using the CPC Error Model," *J. Robotics & Computer-Integrated Manufacturing*, Vol. 9, No. 3, 1992, pp. 227-237.
- [8] Brockett, "Finite Dimensional Linear Systems," John Wiley and Sons, 1970.
- [9] Mooring, B. M., Z. S. Roth, and M. R. Driels, 1991, *Fundamentals of Manipulator Calibration*, John Wiley & Sons.
- [10] Borm, J. H., and C. H. Menq, "Determination of Optimal Measurement Configurations for Robot Calibration Based on Observability Measure," *Int. J. Robotics Research*, Vol. 10, No. 1, Feb. 1991, pp. 51-63.
- [11] Zhuang, H., Zvi S. Roth and K. Wang, "Robot Calibration Using a Modified Denavit-Hartenberg Model," *Proc. 4th Int. Symp. Robotics & Manufacturing*, Santa Fe, NM, Nov. 1992, pp.97-102.



(a) Result of one-stage algorithm



(b) Result of two-stage algorithm

Figure 2 Comparison of the one-stage algorithm and a two-stage algorithm

A SEQUENCE CONTROLLER BASED ON AUGMENTED TIMED PETRI NETS

Kurapati Venkatesh, Chenthilvel S. and Masory O.

Department of Mechanical Engineering
Florida Atlantic University
Boca Raton, Florida 33431

ABSTRACT

Software development for simulation and control of flexible automated systems (FASs) is an important task for its successful implementation. One of the topics in current research in FASs is the development of integrated software systems that can both simulate and control FASs. In this paper an integrated software system that is developed using Augmented Timed Petri Nets (ATPNs) is described. The software package allows the system designers to evaluate the system performance by simulation and to implement its controller using the same data base which describes the system configuration.

1. INTRODUCTION

A flexible automated system (FAS) consists of several concurrent units such as machines, robots, automated guided vehicles, programmable logic controllers, and computers which function asynchronously to meet the real time constraints of a production line and dynamically changing needs of the market. Because of its complexity, integrated software development for FMSs is very important to realize the full benefits of FMSs.

Simulation software generates results that aid in design, performance evaluation, and indepth analysis of the system. There are several popular packages like SIMAN, SLAM, CINEMA, EXCELL etc. for the simulation of manufacturing systems. However, these packages can not be used to control the system

and therefore other means are used to develop the system controller.

The typical functions of the control software are to monitor the system functioning and determine the states of different elements in the system with respect to real time. Traditionally, the sequence of operations executed by the control software of manufacturing systems has been captured by ladder diagrams. These diagrams specify the input and output procedures of the Programmable Logic Controller (PLC) that drive and cycles the operations of a manufacturing device. These diagrams grow so complex that locating the cause when a problem is detected becomes extremely difficult [1]. Further more their usage is limited only to control the system but not to evaluate the performance of the system.

As a result, one of the most primary goals of current research in FASs is to develop integrated software systems that can both simulate and control FASs [1,2,3,4,5,6]. Moreover, usually the simulation software is developed by either software and/or industrial engineers while the control software is developed by manufacturing and/or control engineers. The integration between the abovesaid personnel is very prominent for the successful implementation of such multi-disciplinary projects [5,6,7]. One efficient method to provide such integration is to develop integrated software package that can be used for both simulation and control. In order to develop such integrated software there is a need for integrated modeling tools that support

all the stages of system development, starting from its design to implementation.

In this paper Augmented Timed Petri Nets (ATPNs) are first time introduced as modeling tools to develop integrated software for simulation and control. The reasons for selecting Petri Nets (PNs) and extending them to ATPNs as a modeling tool are detailed in [8]. This paper concentrates on the application of ATPNs for control. An example, in which an industrial automated system has been both simulated and controlled, is provided.

2. PETRI NETS CONCEPTS

PNs are powerful modeling tools that are being recently applied to manufacturing systems [2,3,4,5,6,8]. The basic theory of PNs can be found in [11]. Graphically a PN is defined as a bipartite graph containing "places" (represented by circles) and "transitions" (represented by bars). Places and transitions are connected by "directed arcs" (represented by arcs with arrows). Places contain "tokens" (represented by dots). Places can model different entities comprising the system such as robots, and different intermediate states of the system entities such as "robot 1 loading machine 3". Transitions can model events/activities involved in the system such as "machine 1 finished processing". The basic constructs of PN modeling can provide the basic logic functions available in most Programmable Controllers as summarized in figure 1.

There exists several classes of PNs such as timed PNs, Colored PNs (CPNs), Stochastic PNs (SPNs), predicate/transition PNs (PPNs). CPNs, SPNs, PPNs are not easily understandable to specialists not in the area of PNs. Hence, TPNs that are easy to understand and aid in the integration of different people stated earlier are selected in the present study. However, conventional TPNs reported in the literature are limited only for simulation and performance evaluation purposes but not for controlling the system. In this paper TPNs are extended with

new constructs that are used for controlling the system and named as Augmented TPN.

A. Petri Net (PN):

PN - "N" is a 5 tuple, $N = (P, T, IN, OUT, M)$

where: $P = (p_1, p_2, \dots, p_n)$ is a set of places,
 $T = (t_1, t_2, \dots, t_n)$ is a set of transitions,
 $IN: (P \times T) \rightarrow S$
 $OUT: (T \times P) \rightarrow S$

are input and output functions defining directed arcs between places and transitions, where S is a set of all positive integers k:

If $k=1$ a directed arc is drawn without a label

If $k > 1$ a directed arc is drawn with label k.

If $k = 0$ no arc is drawn.

Input place set: I_t - The input place set of transition " t_i " is $I_t = \{ P / (p \in_i, t_i) \in IN \}$

Output place set: O_t - The output place set of transition " t_i " is $O_t = \{ P / t_i, p_i \in OUT \}$

Marking: M - Marking of a PN is a mapping from the set P to $Q = (0, 1, 2, \dots)$, i.e. $M: P \rightarrow Q$ means that M inputs tokens to every place, $M_i = M(p_i)$. Q indicates the number of tokens in place p_i .

B. Timed PN:

A timed PN (TPN) is an improvement over PN described earlier and includes a set of firing durations D. The TPN is formally defined as $TPN = (P, T, IN, OUT, M, D)$ where $D: T \rightarrow \{0, R^+\}$, where $\{0, R^+\}$ is the set of all non-negative real numbers. A transition has an associated deterministic firing time defined by two events: "start firing" and "end firing". In between these two events, the firing is in progress. The removal of tokens from a transition's input places(s) occurs at "start firing" and the placement of tokens on a transition's output places occurs at "end firing". While the firing of a transition is in progress, the time to end

firing, called the remaining firing time (R), decreases from firing duration to zero.

Instantaneous Description (ID): The state of the TPN can be defined by the instantaneous description, ID, which is a four-tuple:

$$ID = (M, F, R, AT)$$

where: M is a marking function, $M: P \rightarrow S$;

F is a selector function, $F: T \rightarrow (0,1)$;

The selector function is nothing but a firing vector (F-vector).

If $F(t_i)=1$, t_i is ready to fire

If $F(t_i)=0$, t_i is not ready to fire

R is remaining firing time function,

$R: T \rightarrow (0, R^+)$ is a cumulatively decreasing time function.

AT is the active time duration function

AT: $T \rightarrow (0, AT^+)$ is a cumulatively increasing time function.

With ID, it is possible to determine the state and performance of the system at any time. For example, using AT, the utilization of the system elements can be determined.

C. Augmented timed PN:

An augmented TPN (ATPN) is an improvement over TPN and introduced at the first time in this paper. An ATPN is aimed for both control and simulation of the system. To control an automated system the controller has to read inputs and send outputs to the physical system. Hence, the standard TPN has to be augmented to accommodate these functions by including two tuples namely,

1) Input signal vector (ISV) that is intended to read the state of the input signals from digital input interface.

2) Output signal vector (OSV) that is intended to send output signals through digital output interface.

An ATPN is a 8 tuple and defined as:

$$ATPN = (P, T, IN, OUT, M, D, ISV, OSV)$$

ISV is a mapping from set P to $S=(0,1,2,...)$ i.e. $ISV: P \rightarrow S$ where ISV associates attributes to every place, $S_i=S(p_i)$. S_i is an attribute associated with place p_i and represents the input channel associated with place p_i . For example, if place p_i models a limit switch, the ATPN reads the status of that switch from the digital input interface through the channel number represented by S_i . S_i is the second attribute of p_i , the first attribute being the initial marking $M(p_i)$.

OSV a mapping from set T to $O=(0,1,2,...)$ i.e. $OSV: T \rightarrow O$ where OSV associates attributes to every transition, $O_i=O(t_i)$. O_i is the attribute associated to a transition t_i which represents the binary number that is to be sent to the digital output interface. For example, t_i may be modeling the activity "send signal to actuate solenoid A". Each solenoid is activated by writing a specific binary number on to the digital output interface. During execution of the program, the ATPN writes the number O_i to digital output interface to actuate solenoid A. O_i is the second attribute of t_i , the first attribute being the transition duration, $D(t_i)$.

Transition firing: A transition t_j of an ATPN is said to be enabled in a marking M:

$$\text{iff } M(p_i) > IN(p_i, t_j) \text{ and } S(p_i) = 1 \forall P_i \in I_t(t_j).$$

Enabled in a marking M, t_j fires and results in a new marking M according to equation:

$$M(p_i) = M(p_i) + OUT(p_i, t_j) - IN(p_i, t_j) \forall P_i \in P$$

After firing, t_j writes $O(t_j)$ to the digital output interface. Here, M is said to be reachable from M.

3. SOFTWARE DESCRIPTION

The software package developed to execute ATPNs is written in C++ and has five major modules with their functions described below.

A. Read_Petri_net: This module reads an input file that specifies the structure of the PNM; transition timing durations and output signal vector; initial marking and the input signal vector. The structure of the PNM means the connectivity between places and transitions including the weights on the connecting arcs.

B. Enabled_transitions: This module generates an F-vector as an output vector. It scans the whole PNM at each instant of time and finds the transitions that are ready to fire. Thus, the F-vector indicates the transitions that are enabled to fire with respect to real-time. The F-vector is the input for modules "Conflict", "New_marking", and "Main".

C. Conflict: This module determines the transitions that are in conflict and stops the program execution until the conflict is resolved. Once the conflict is resolved, the program execution is resumed. A conflict in PNM results when an element is shared by two other elements of the system (e.g. a single robot serving two machines that demand service at the same time). In such cases, the module detects the conflicts and resolves it by disabling one of the two transitions that are enabled at the same time.

D. Minimum_time: This module scans the whole PNM and detects the transition that has minimum time to fire. As there can be more than one transition with minimum time, the outputs from this module are both the number of transitions with minimum time and their identity.

E. New_marking: This module contains two submodules: 1) "Read_marking" checks for the second attributes of all places that are inputs for enabled transitions. 2) "Update_marking" fires the transitions and changes the current marking.

Read_marking uses F-vector as input. If a transition t_i is enabled it checks for the second attribute of all input places of t_i . If the second attribute of an input place is high it sets the variable "flag" corresponding to that place as TRUE. After flags corresponding to all the

input places of t_i are set to TRUE, it removes the tokens from these places and sends a signal to "update_marking" to fire t_i .

After receiving the signal corresponding to the transition to be fired from "read_marking", "update_marking" sends the second attribute of the transition to be fired to the digital output interface and deposits tokens in all the output places of the transition fired.

F. Main: This module coordinates the functioning of above modules and generates a status report of the system elements. The report is stored in an output file which is updated whenever a transition is fired in the PNM. Whenever an event occurs in the system, the output file is appended by the time at which the event occurred, marking of the PNM, F-vector, R-vector, and AT-vector.

4. EXAMPLE

For illustration an automatic machine, shown in figure 2, that produces parts from a reel of plastic tape, is considered. Piston A is used to index the tape, piston B drives a punch and piston C cuts the tape. The required sequence for this application is :

ST, 2 [A+,{ A-,B+},B-], 2 (A+,A-), C+,C-

For this system, the assignment of input and output channels is shown in tables 1 and 2. In table 1, BNA (BND) indicates the binary number that has to be sent to digital output interface to activate (deactivate) the corresponding solenoid or light. Figure 3 shows the Petri Net controller for this system, and table 3 the corresponding input file.

For example, in figure 3 for transition 2 (t_2), the timing duration is 1 time unit and hence the first attribute of t_2 is 1. For the same transition, the second attribute is 1. This is because t_2 models the activity "activate solenoid A". By referring to the table 1, the binary number that is to be sent to the digital output interface to activate solenoid A is 1.

The "Read_marking" module performs three functions: 1) If the second attribute of a place is less than 11 (because the number of inputs for this system is 11) it reads the digital input interface; if it is greater than 11 it does not read digital input interface and skips to check the attribute of next place; 2) If the second attribute of a place is less than 11 and if the number of channel assigned to the corresponding place (from table 1) is TRUE it sets the variable "flag" corresponding to that place as TRUE; and 3) The above two steps are repeated until all "flags" corresponding to input places of a transition enabled are set TRUE. Then it removes tokens from the input places of a transition and sends the signal to "update_marking" to fire the enabled transition.

In figure 3 the first attribute of place 1 (p1), used to model "start", is 1 indicating that there is one token present in p1 at the start. Switch 1 (SW1) is assigned as the start button. Transition 1 is the output transition for p1, and in order to activate this transition, which models the activity "push SW1", SW1 has to be pushed. Now, referring to the table 2, the corresponding attribute for SW1 is 8. Hence, the second attribute for p1 is 8. At the time "zero", the system will not start until SW1 is pressed by the operator. Then, the "read_marking" reads the channel 8, sets the flag corresponding to p1 to TRUE, sends a signal to "update_marking" to fire t1 and to remove the tokens from the input places of t1. Once t1 is fired, the number 1 (the second attribute of t1) is sent to the digital output interface activating the solenoid A and tokens are deposited in the output places of t1.

Table 4 shows the typical output results for the PNM shown in figure 3. The information in the output file is very much useful for both simulation and control purposes. By looking at the marking of the PNM the status of the elements in the system such as "solenoid limit switch b0 is closed", "piston A moving forward", "the number of cycles finished", etc. can be easily known. By looking at the R-vector, the remaining time to finish a transition

can be found. By looking at the AT-vector, the utilization of the system can be found. For example, The information in the output file would be of much help to the control engineer to diagnose the system functioning with respect to real time. Thus the information obtained in the output file is of immense help to both system designers and control engineers.

5. CONCLUSIONS

A software package usable for controlling of automated systems, based on Petri Nets, is described. The advantages of this approach and the software structure are discussed. In addition, the control of an industrial automated system has been illustrated. Future research aims are to upgrade the package to an object-oriented software, to include graphics for animation of the system, and to extend its capabilities for modeling complex systems.

6. REFERENCES

1. J.K. Chaar, R.A. Voltz, and E.S. Davidson, "An Integrated Approach to Developing Manufacturing Control Software", Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, April 1991.
2. S. Godavari, M.C. Zhou, and N. Levy, "Software Integration in Design of Flexible Manufacturing Systems", Proceedings of 1991 International Symposium on Intelligent Control, 13-15, August 1991, Arlington, VI.
3. M. Zhou, and M.C. Leu, "Modeling and Performance Analysis of Flexible PCB Assembly Station Using Petri Nets", Transactions of the ASME, Vol. 113, December, 1991.
4. M.C. Zhou, F. DiCesare, and D. Rudolph, "Control of a Flexible Manufacturing System Using Petri Nets", 11th IFAC World Congress, Automatic Control in the Service of Mankind, Tallinn, Estonia, USSR, August 13-17, 1990.
5. J.M. Proth and F. Vernadat, "Discrete Manufacturing Systems: From Specification to Evaluation", The Second International Conference on Automation Technology, July 1992.
6. Kurapati Venkatesh, O.V.K. Chetty and V. Radhakrishnan, "Software Development for Fully Automated Industries", Proceedings of the International Conference on Design Automation and Computer Integrated Manufacturing, PSG College of Technology, Coimbatore, India, January 1990.
7. P.Y. Huang, and M. Sakurai, 1990, "Factory automation: the Japanese experience", IEEE Transactions on Engineering Management, Vol. 37, No. 2, May, 103-108.
8. Kurapati Venkatesh, O.V.K. Chetty, and K. R. Raju, 1990, "Simulating Flexible Automated Forming and Assembly systems", International Journal of Material Processing and Technology, Vol. 24, 453-462.

Solenoid or Light	Channel #	BNA	BND
A	0	1	-1
B	1	2	-2
C	2	4	-4
Green	4	16	-16
Red	5	32	-32

Table 1: The outputs assignment.

Channel #	0	1	2	4	5	6	8	10
Switch	a1	b1	c1	a0	b0	c0	sw1	ES

Table 2. The inputs assignment.

Transitions Input Places

```

10000000000000000000000000000000
0100100001000010000100001000010
00100000000000000000000000000000
000100000000000000000000000010000
00001110000000000000000000000000
00000001000000000000000000000000
00000000100000000000000000000000
0100100000000000000000000100100
00010000000100000000000000000000
00000000000010000000000000000000
01000000000002100000000000000000
00000000000000001000000000000000
00000000000000001000000000000000
00000000000000001000000000000000
00000000002000000000000000000000
000000000000000000000000010000

```

Transitions Output Places

```

01000000000000000000000000000000
00101000000000100000000000000000
00010000000000000000000000000000
00000110000000000000000000000000
00000001000000000000000000000000
01000000100000000000000000000000
00001000011000000000000000000000
00100000000100000000000000000000
00000000000100000000000000000000
0100100000000100000000000100
00000000000000010000000000000000
00000000000000001000000000000000
00000000000000001000000000000000
0100000000000000100012020000
000000000000000000002000000
0000000000000000000000000000200

```

D-Vectr 1111111111111111

OSV 01010-201-1040-4000

Marki ng 10001000010000100002023100

ISV 8 4 12 0 5 12 12 1 12 12 12 12 12

12 6 12 2 12 12 12 12 12 12 12 10

Table 3: Input file structure.

```

TIME: 1
MARKING: 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0
          0 0 0 2 0 2 2 1 2 0
F-VECTOR: 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
R-VECTOR: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
AT-VECTOR: 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
TIME: 2
MARKING: 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1
          0 0 0 1 0 2 1 1 3 0
F-VECTOR: 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
R-VECTOR: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
AT-VECTOR: 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2
TIME: 3
MARKING: 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1
          0 0 0 1 0 2 0 1 5 0
F-VECTOR: 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
R-VECTOR: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
AI-VECTOR: 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 3

```

Table 4: Sample of output.

Symbol/Construct	Meaning
	PLACE Models a condition or the status of a system element
	TRANSITION Models an activity
	DIRECTED ARC Models the flow of information or material.
	Taken in a Place Models the truth value of the condition associated with the place. If there is a token in place, the condition modeled is fulfilled.
	Logical AND If A = true and B = true and C = true then D = true
	Logical OR If A = true or B = true or C = true then D = true
	Concurrency If A = true and B = true then C = true and D = true and E = true
	Time delay If A = true then delay "1 time units": B = true
	Synchronization If A = true then delay "11 time units": D = true If B = true and C = true then delay "12 time units": E = true If D = true and E = true then delay "13 time units": F = true Here, operations with the different timings are synchronized to trigger third operation.
	Hierarchical modeling Here, Petri Net consisting of G, H, I, and J models the system at higher level. The Petri Net consisting of A, B, C, etc. models the system at lower level.

Figure 1: Basic PN constructs.

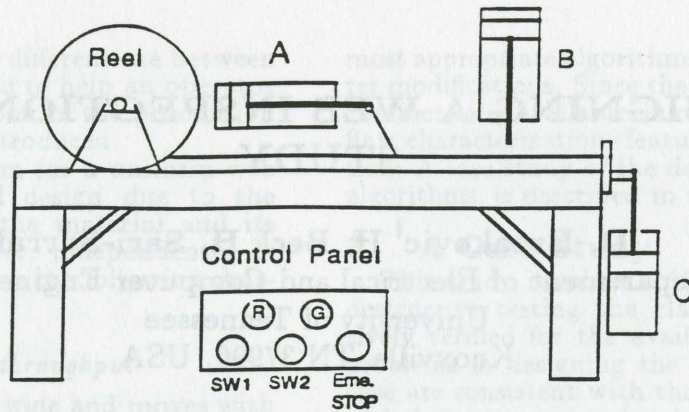


Figure 2: Schematic of the example hardware.

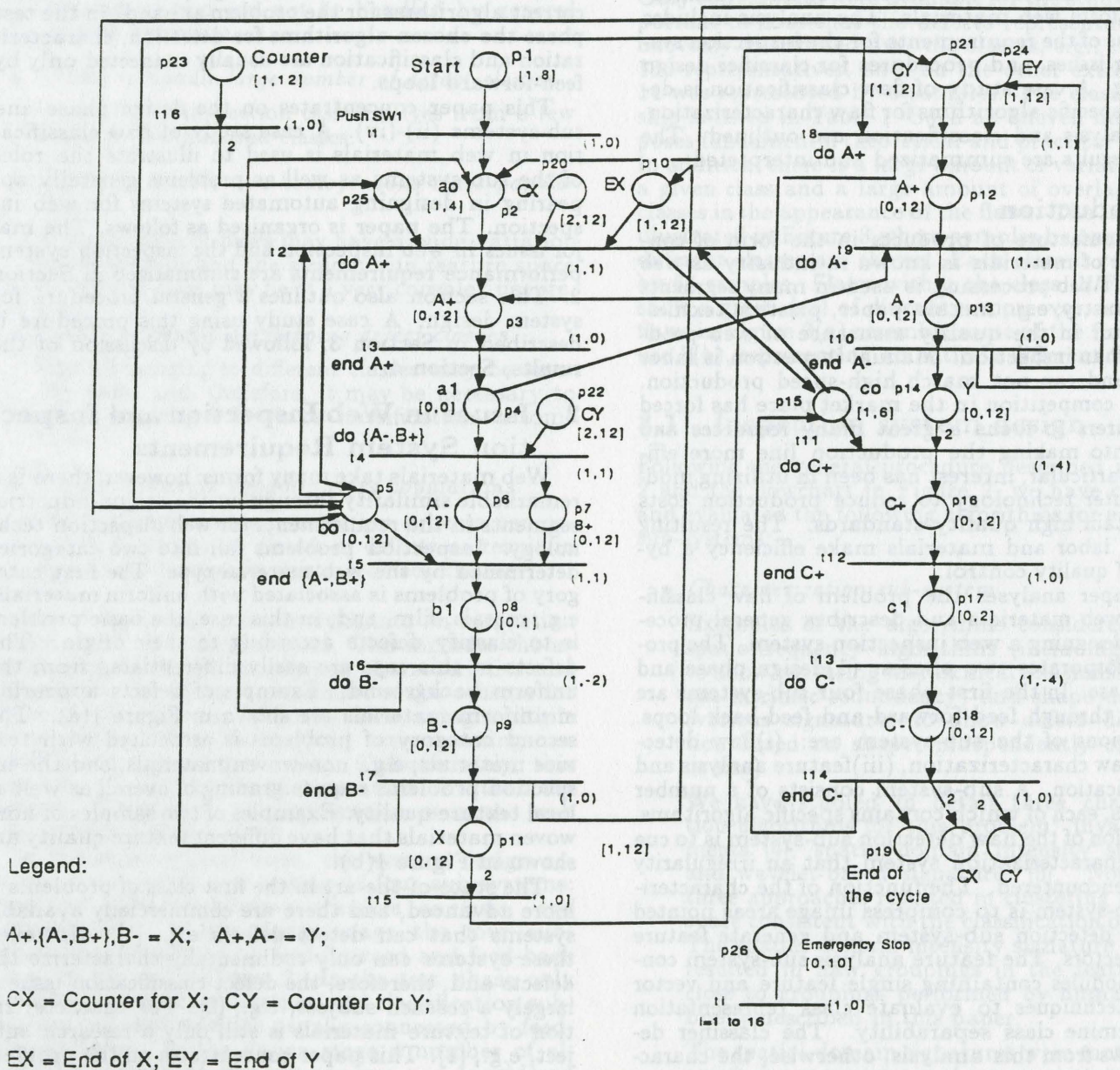


Figure 3: Petri Net Controller for the example.

ISSUES IN DESIGNING A WEB INSPECTION SYSTEM: CASE STUDY

D. Brzakovic¹ H. Beck H. Sari-Sarraf

Department of Electrical and Computer Engineering
University of Tennessee
Knoxville, TN 37996, USA

Abstract

This paper analyses the problem of flaw classification in uniform web materials. The analysis includes descriptions of the requirements for the inspection system, sensor issues, and procedures for classifier design and testing. A case study of flaw classification is detailed and specific algorithms for flaw characterization, feature analysis and classification are outlined. The obtained results are summarized and interpreted.

1 Introduction

The manufacture of products in the form of continuous rolls of materials is known in industry as web processing. Web processing is used in many segments of U.S. industry, e.g., metals, paper, plastics, textiles. A key factor in the quality assurance of web products is human inspection. Manual inspection is labor intensive and can not match high-speed production. Increasing competition in the market place has forced manufacturers to focus a great many resources and energies into making the production line more efficient. In particular, interest has been in utilizing modern computer technology to reduce production costs and maintain high quality standards. The resulting savings in labor and materials make efficiency a by-product of quality control.

This paper analyses the problem of flaw classification in web materials and describes general procedures for designing a web inspection system. The procedure incorporates two phases: (1) design phase and (2) test phase. In the first phase four sub-systems are connected through feed-forward and feed-back loops. The functions of the sub-system are: (i) flaw detection, (ii) flaw characterization, (iii) feature analysis and (iv) classification. A sub-system consists of a number of modules, each of which contains specific algorithms. The function of the flaw detection sub-system is to cue the flaw characterization system that an irregularity has been encountered. The function of the characterization sub-system is to compress image areas pointed at by the detection sub-system and generate feature pattern vectors. The feature analysis sub-system consists of modules containing single feature and vector analysis techniques to evaluate class representation and determine class separability. The classifier design follows from this analysis; otherwise, the characterization sub-system is cued for additional measure-

ments. The classification sub-system performs recognition. The objective of the design phase is to choose correct algorithms for the problem at hand. In the test phase the chosen algorithms for detection, characterization and classification are usually connected only by feed-forward loops.

This paper concentrates on the design phase and sub-systems (ii)-(iv). A case study of flaw classification in web materials is used to illustrate the roles of the sub-systems as well as problems generally appearing in designing automated systems for web inspection. The paper is organized as follows. The major issues in web inspection and the inspection system performance requirements are summarized in Section 2. This section also outlines a general procedure for system design. A case study using this procedure is described in Section 3, followed by discussion of the results, Section 4.

2 Issues in Web Inspection and Inspection System Requirements

Web materials take many forms; however, there is a remarkable similarity throughout the major industrial segments in the requirements for web inspection technology. Inspection problems fall into two categories determined by the web material type. The first category of problems is associated with uniform materials, e.g., metals, film, and, in this case, the basic problem is to classify defects according to their origin. The defects in this case are easily differentiated from the uniform background. Example of defects appearing in uniform materials are shown in Figure 1(a). The second category of problems is associated with texture materials, e.g., non-woven materials, and the inspection problems require grading of overall as well as local texture quality. Examples of two samples of non-woven materials that have different texture quality are shown in Figure 1(b).

The state-of-the-art in the first class of problems is more advanced, and there are commercially available systems that can detect defects e.g., [3]. However, these systems can only rudimentally characterize the defects and, therefore, the defect classification issue is largely a research subject, e.g., [2]. The characterization of texture materials is still only a research subject, e.g., [1]. This paper concentrates on the problem of uniform web materials where the highest priority

¹Present address: Department of Electrical Engineering and Computer Science, Lehigh University, Bethlehem, PA 18015

technical need is for an ability to differentiate between several distinct types of flaws and to help an operator on the factory floor assess the point in the production process where flaws are being introduced.

Typically an inspection system for a uniform web material requires a customized design due to the unique characteristics both of the material and its manufacturing process. However, independently of the material, a system should have the following characteristics:

- *Ability to handle high data throughput*

A typical web is 8 – 10 feet wide and moves with speeds ranging from 600 to 6000 feet/minute. Consequently, the data throughput for 100 % inspection (when detecting flaws of mm size) is tremendous and can not be handled by general purpose hardware.

- *Ability to handle large number of defect classes*

Typically an inspection task involves from a few dozen to few hundred classes.

- *Ability to handle non-homogeneous class populations*

A single class of flaws may have a wide variation in appearance and the structure in feature space for a given class may be of a very complex nature.

- *Ability to handle overlapping defect classes*

Flaws belonging to different classes may be visually alike, and, therefore, it may be necessary to assign confidence levels to classification of some flaws.

- *Ability to handle dynamic class populations*

Small changes in the production process can result in entirely new classes of defects or result in significant changes in existing defect classes.

One of the major issues in designing an inspection system is the selection of sensors. This includes choosing a sensor with the appropriate characteristics and determining the optimal placement of the sensors for the production process. In many cases, using multiple sensor inputs, sensor fusion, and active sensing can simplify the classification task. Unfortunately, this issue has yet to capture the attention of either the research community or practitioners in the web inspection industry.

In the most general form, the system design requires development and testing of four sub-systems: *Detection, Characterization, Feature analysis, and Classification*. In the design stage the four sub-systems are connected by feed-forward and feed-back loops, as shown in Figure 2. In the test phase only the detection, characterization, and classification sub-systems are used and are usually connected by feed-forward loops. In the design stage a number of algorithms in each of the sub-systems are considered and, based on the characteristics of the problem, the

most appropriate algorithms are chosen, sometimes after modifications. Since the flaw detection systems are commercially available, in our work we concentrate on flaw characterization, feature analysis, and classification. A case study of the design phase, listing specific algorithms, is described in the next section.

3 A Case Study

This study involves a 15 class problem. Through destructive testing the class membership was positively verified for the available samples. The major problems in designing the inspection system in this case are consistent with those described in Section 2, and the most concerning issue is to determine if the information captured within the sensitivity band of the sensor can be related to the known classification. Only 628 samples were available for the study. It is important to note that the defects representatives were unevenly distributed over the classes, e.g., class 2 had 156 representatives, and, on the other extreme, class 11 was represented by 6 samples. The class membership is listed in Table 1. Such an uneven membership poses fundamental theoretical and practical problems. In addition, there is a large amount of variation within a given class and a large amount of overlap between classes in the appearance of the flaws. The last point is illustrated in Figure 3 where samples belonging to two different classes are shown. Each class is represented by 64 samples. Flaws within the class in Figure 3(a) show a great deal of diversity; moreover, some of the flaws in Figure 3(a) resemble some of the flaws in Figure 3(b), representing an entirely different flaw class.

3.1 Inspection system design

Following the general procedure described in Section 2 and the flowchart in Figure 2, we have considered and evaluated the following algorithms for each of the sub-systems:

- *Characterization sub-system*

Examples of the algorithms considered include spatio-frequency algorithms (including wavelets to allow capturing hierarchical relationships without intrinsic redundancy) and shape descriptors (various signatures to allow defect discrimination based on shape, independently of position or size).

We have studied in detail three characterization algorithms for this problem: invariant moments [4], spatial-domain intensity signatures, and wavelet-based signatures [5]. None of the three approaches resulted in clustering in feature space consistent with the classification objectives. The spatial-domain intensity signature approach resulted in flaw groupings in the feature space close to groupings performed by humans and is briefly described in this paper.

The spatial-domain signatures were motivated by our desire to capture in the same signature shape, orientation, and intensity variations of a defect.

For an image, $f(x, y)$, an element s_{θ_i} of signature S is defined as

$$s_{\theta_i} = \frac{1}{K} \sum_{r=1}^K f(x_0 + r \cos \theta_i, y_0 + r \sin \theta_i), \quad (1)$$

where (x_0, y_0) is the center of gravity, $0 \leq \theta_i \leq 180^\circ$ varying in predetermined steps δ , and K is determined by the physical edges of the image region considered. An n element signature is defined as $S = (s_0, s_\delta, s_{2\delta}, \dots, s_{(n-1)\delta})$, where $(n-1)\delta = 180^\circ$. We have used $\delta = 5$ in our experiments, thus acquiring 72 element pattern vectors. Since the acquired signatures/pattern vectors did not immediately show distinct features for further data reduction, we have subjected the entire signature to the feature analysis sub-system. It should be noted that the backgrounds were normalized prior to signature generation in order to counter-effect differences in image acquisition conditions.

- *Feature analysis sub-system*

The algorithms considered included, among others, clustering, dimensionality reduction, hierarchical structures for reducing the complexity of the analysis task, and strategies for combining different representation schemes (e.g., symbolic and quantitative).

The specific choices of steps in the analysis were based on the pattern vector being a one dimensional signal representation of a two dimensional signal. The first step was to generate basic statistics. Based on this, two classes represented by a single vector in the original data set were discarded from the data set. In the next step, a measure of cluster validity was used in an iterative search for the optimal number of clusters inherent in the data based on the K-means algorithm. There is a mixed class representation in the majority of the clusters. A further search on individual classes revealed that some of the classes contained as many as 15 clusters, and even the most sparsely represented classes had members distributed among 3 clusters. Table 2 shows the distribution of class population among the obtained clusters. The images associated with the clustered vectors were examined for visual homogeneity, showing an immediate separation of the flaws into two separate groups: large, bright flaws and small, low contrast flaws. Thus, at the first pass, there appears to be the necessary relationship between the characterization and the visual characteristics of the flaws. Examples of the obtained clusters are shown in Figure 4.

The next step in the analysis was to determine the amount of resolution necessary to separate the existing clusters into more homogeneous groups based on visual characteristics. The technique utilized for this was a visually guided variant of the ISODATA clustering algorithm. The general

strategy is to incorporate the ISODATA features of cluster deletion, splitting, and merging, guided by the visual information in the images associated with the clustered vectors. The obtained clusters contained fairly homogeneous populations. It should be noted that even when further increasing the degree of resolution the cluster membership clearly shows that there is no separability using the available data set. Also, there becomes a point where the increasing degree of resolution becomes nonsensical.

- *Classifier sub-system*

The classification algorithms considered draw from single, hierarchical, and hybrid approaches, based on deterministic, stochastic and fuzzy pattern recognition principles. Decision rules can be constructed from different data representation schemes, e.g., symbolic or quantitative, and can incorporate multiple representation schemes into the decision rules.

The classifier design generally falls directly out of the analysis of the features. The preceding discussion, however, indicates that there is little correlation between the classes and the visual information; therefore, it would be fruitless to design a classifier for this task. A non-parametric classifier could be designed, backprop or Parzen windows, that would produce very good results for a train A - test A trial; however, the results would be very misleading.

4 Discussion

This paper describes the major issues in designing an inspection system for uniform web materials. The design phase considers four sub-systems: the detection, characterization, feature analysis and classification sub-systems. The performance of the characterization sub-system is evaluated by the feature analysis sub-system (in context of the specific classification problem), and an appropriate classifier is designed or chosen based on this analysis. The results obtained in the case study indicate that there is little correlation between the destructive analysis grading of the flaws and the information encoded by the flaw characterization methods. There is, however, a strong correlation between the flaw characterization and the appearance of the flaws. Similar results were obtained using different characterization algorithms, leading us to the conclusion that in this particular case either the chosen sensor was not appropriate for the task at hand, or that it was not placed on the appropriate position in the production process.

References

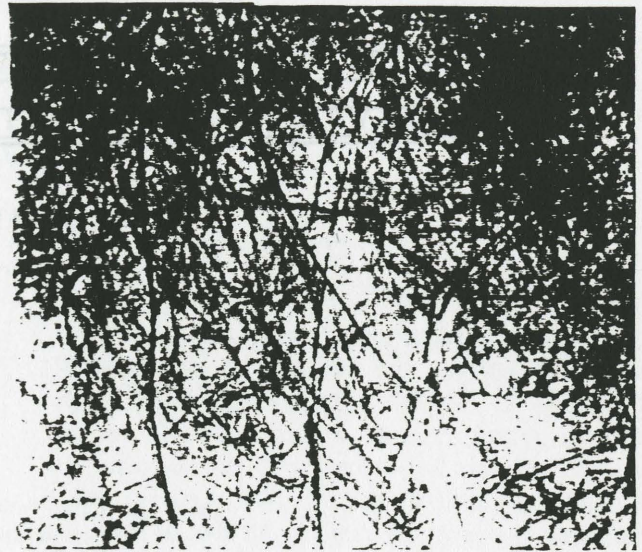
- [1] Brzakovic, D., Beck, H., and Sufi, N. "Flaw Classification in Texture materials," *Advances in Image Processing*, Y. Mahdavih and R.C. Gonzalez (Eds.), SPIE Optical Engineering Press, Bellingham, WA 1992.
- [2] Brzakovic, D., Beck, H., and Shanmugam, P., "Defect Characterization and Classification for

Web Materials," Proc. of IEEE Conference on Signal Processing, Speech, and Acoustics, Minneapolis, MN, 1993.

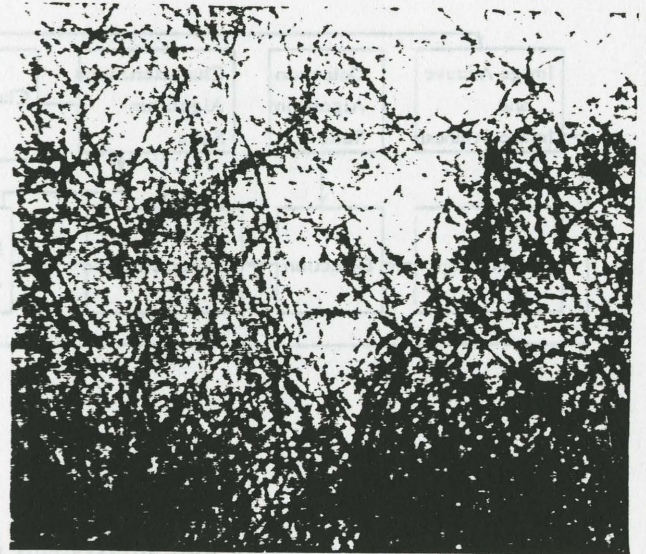
- [3] Guary, J.L., "A Machine Vision System for Real-Time Inspection of Moving Web," Proc. of SPIE: Intelligent Robotics and Computer Vision 8, Systems and Applications, Phil., Pa, 1989.
- [4] Hu, M.K. "Visual Pattern Recognition by Moment Invariants," *IRE Trans. Info. Theory*, vol. IT-8, pp. 179-187, 1962.
- [5] Sari-Sarraf, H. and Brzakovic, D. "2-D Signal Classification by Multiscale Wavelet Representation." Proc. of SPIE's Technical Program on Image Processing and Signal Processing, San Diego, CA, 1992.

ACKNOWLEDGEMENT

The work described in this paper has been funded in part by the Web Inspection Consortium, and the Center for Measurement and Control at the University of Tennessee.



(a)



(b)

Figure 1: Examples of different web materials: (a) uniform material and examples of flaws belonging to a single class—total of 64 flaws, (b) two samples of textured material that have distinctly different overall texture characteristics.

For an image, $f(x, y)$, an element s_i of signature S is defined as



The algorithms considered included, among others, the following:

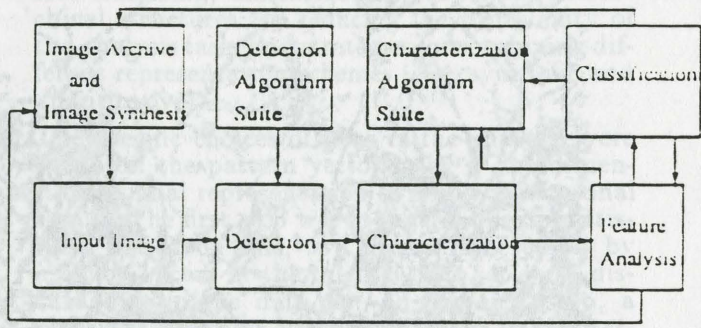
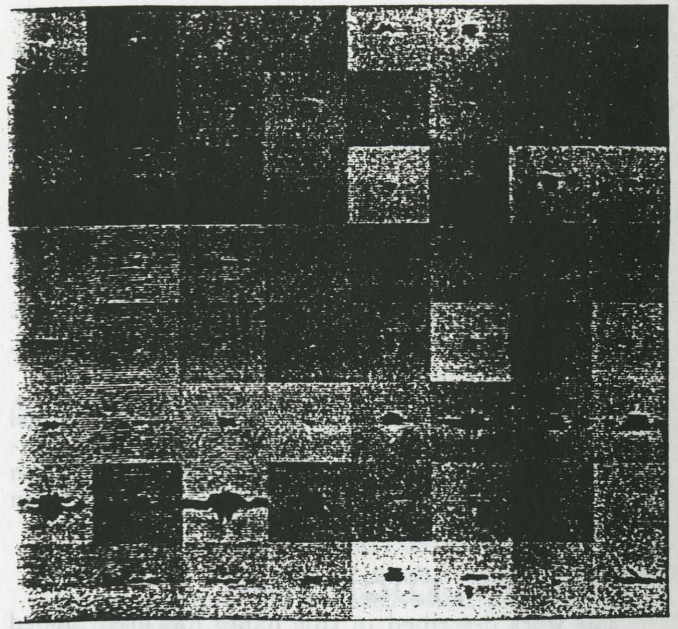
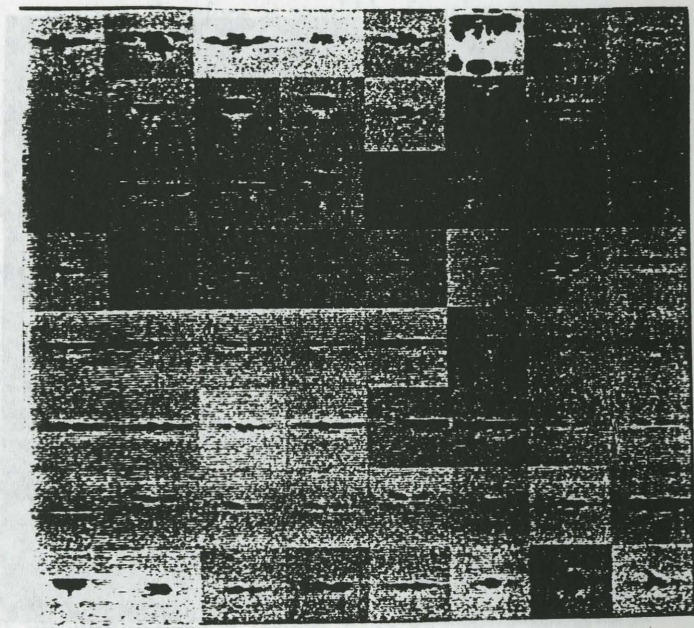


Figure 2: The interaction of four sub-systems in the design phase of a web inspection system.

The images associated with the clustered vectors were examined for visual homogeneity, showing an immediate separation of the flaws into two separate groups: large, bright flaws and small, low contrast flaws. Thus, at the first pass, there appears to be the necessary relationship between the size of the flaw and its contrast. The next step in the analysis was to separate the existing clusters into more homogeneous groups based on visual characteristics. The technique utilized for this was a visually guided variant of the ISODATA clustering algorithm. The general



(a)



(b)

Figure 3: Two class example illustrating inter-class diversity and intra-class similarity in the case study: (a) class 1, (b) class 2. Each class is represented by 64 samples.

class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
count	81	156	47	105	75	19	11	58	12	17	6	32	7	1	1

Table 1: Class membership for the case study

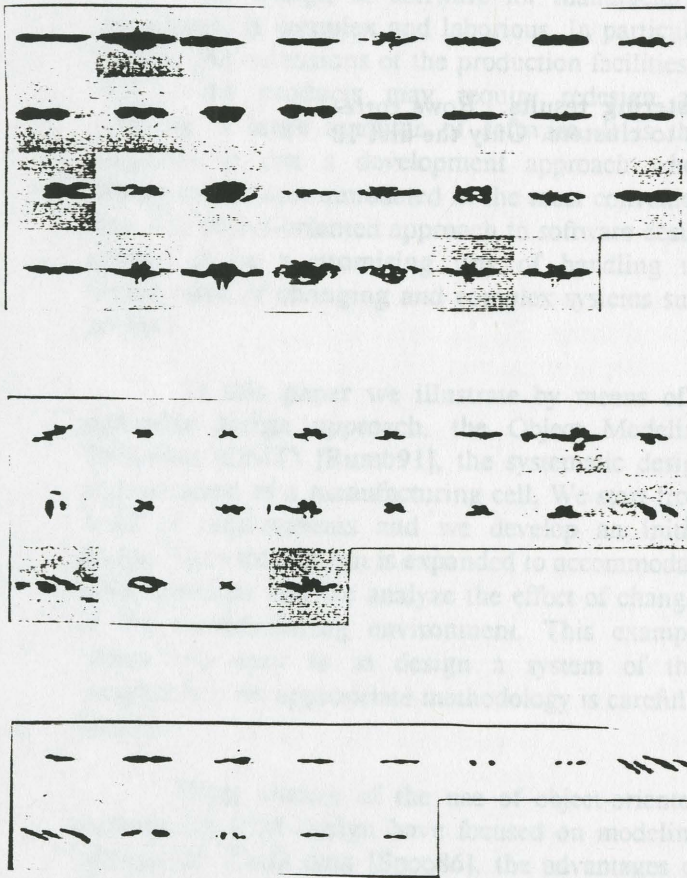


Figure 4: Examples of three obtained clusters that correlate close to the visual characteristics of flaws.

Section II discusses basic background on FMSs and on object-oriented design, introducing the Object Modeling Technique (OMT) of [Rumba91]. Section III introduces a specific system used as reference for our development and we show how to integrate the OMT. Section IV tests our method to design these systems and other functions. We evaluate these results in section V, while section VI presents some conclusions.

II. BACKGROUND

2.1. Flexible Manufacturing Systems

A flexible manufacturing system (FMS) is formed by a group of machine tools and material handling devices. Individual units of the system are controlled by their own numerical control unit, while the system is usually supervised by an executive computer. An FMS system usually has random scheduling capabilities. Since the type of products and their production processes change as the product type changes, to design and implement the software which controls this type of systems is a complicated job.

A typical FMS system contains workstations, load and unload stations, workpiece transport equipment, pallets, fixtures, tools, tool transport, robots, buffer storage at workstations, etc. Different products, in general, require different processes and equipment associated with the manufacturing processes and material handling needs.

2.2. OMT Concepts

Object-oriented modeling and design have been proposed as a way to promote better understanding of requirements, cleaner design and more maintainable systems. These advantages are accomplished through many unique features found in an object-oriented system. We will not define these

0	0	2	0	9	4	0	5	0	17	0	2	11	3	8	6	4	2	0	8
7	1	27	2	16	6	1	6	10	4	0	2	5	12	0	6	31	7	1	12
0	0	0	0	0	0	0	0	0	12	0	28	6	0	0	1	0	0	0	0
8	0	0	0	0	2	0	0	0	43	2	7	9	1	2	18	3	0	10	0
1	6	0	0	0	0	1	0	3	3	13	26	10	1	3	2	1	1	0	4
0	0	0	0	0	0	1	0	0	3	0	7	6	1	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	3	0	2	0	0	0	0	0	0	5	0
0	0	0	0	3	0	0	0	0	7	0	24	20	1	0	1	0	1	1	0
0	0	0	0	0	0	0	0	0	4	0	2	4	0	0	2	0	0	0	0
1	0	2	0	1	0	0	0	0	4	0	0	1	1	0	5	0	1	0	1
0	0	0	0	0	0	0	0	1	2	0	1	0	0	0	2	0	0	0	0
0	0	0	0	0	0	0	0	0	11	0	18	1	0	0	2	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	4	1	0	0	0	0	0	0	1

Table 2: First level clustering results. Rows correspond to classes columns to clusters. Only the first 13 classes are included.

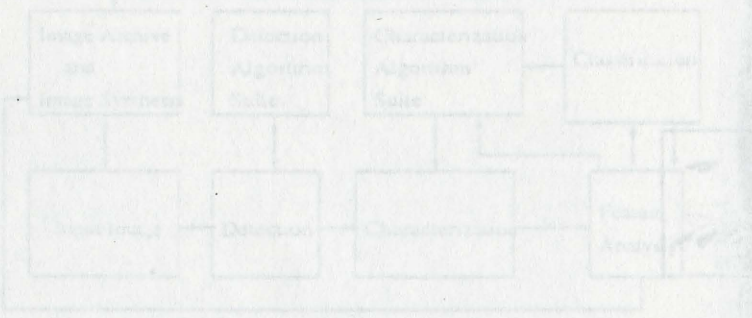


Figure 2: The interaction of four sub-systems in the design phase of a web inspection system.

OBJECT-ORIENTED DESIGN OF FLEXIBLE MANUFACTURING SYSTEMS

E. B. Fernandez* and C.P. Han **

*Department of Computer Science and Engineering

**Department of Mechanical Engineering

Florida Atlantic University

Boca Raton, FL 33431

I. INTRODUCTION

The design of software for manufacturing cell controls is complex and laborious. In particular, changes and extensions of the production facilities as well as the products may require redesign and rewriting of large amounts of software. It is then important to use a development approach where change can be accommodated in the most convenient way. The object-oriented approach to software design appears to be a promising way of handling the development of changing and complex systems such as these.

In this paper we illustrate by means of a particular design approach, the Object Modeling Technique (OMT) [Rumb91], the systematic design and extension of a manufacturing cell. We start from a set of requirements and we develop an initial design. Then this design is expanded to accommodate other functions and we analyze the effect of changes in the manufacturing environment. This example shows how easy is to design a system of this complexity if an appropriate methodology is carefully applied.

Other studies of the use of object-oriented methods for CIM design have focused on modeling mechanical CAD data [Spoo86], the advantages of C++ as a support for this type of environment [Stur91], diagnosis [Grah91], and the use of Ada and Petri nets to model dynamic aspects [Kura92]. The study that comes closest to our work is [HsuC92], but that paper focuses on more general aspects of an object-oriented architecture. Here we show a detailed design example to exhibit the modeling power of this approach and to demonstrate how easy is to perform local changes or broad extensions to the initial design.

Section II discusses basic background on FMSs and on object-oriented design, introducing the Object Modeling Technique (OMT) of [Rumb91]. Section III introduces a specific system used as reference for our development and we show how to model it using OMT. Section IV tests our design to see how to add other functions. We evaluate these aspects in section V, while section VI presents some conclusions.

II. BACKGROUND

2.1 Flexible Manufacturing Systems

A flexible manufacturing system (FMS) is formed by a group of machine tools and material handling devices. Individual units of the system are controlled by their own numerical control unit, while the system is usually supervised by an executive computer. An FMS system usually has random scheduling capabilities. Since the type of products and their production processes change as the product type changes, to design and implement the software which controls this type of systems is a complicated job.

A typical FMS system contains workstations, load and unload stations, workpiece transport equipment, pallets, fixtures, tools, tool transport, robots, buffer storage at workstations, etc. Different products, in general, require different processes and equipment associated with the manufacturing processes and material handling needs.

2.2 OMT Concepts

Object-oriented modeling and design have been proposed as a way to promote better understanding of requirements, cleaner design and more maintainable systems. Those advantages are accomplished through many unique features found in an object-oriented system. We will not define these

here in detail since there is a considerable literature on this subject [Booc91, Mona92, Rumb91].

Some key ideas are central to the mechanisms of the object-oriented approach. The fundamental construct is the *object*, which contains both a data structure and a collection of related procedures, sometimes called *methods*. The only way in which objects interact with each other is by sending each other messages or by calls to their interface. Objects with the same data structure (attributes) and behavior (methods) can be grouped into a class. Each object is called an instance of the class. A central feature of object-oriented systems is *inheritance*. Inheritance is a mechanism whereby one class of objects can be defined as a special case of another class, automatically including the methods and attributes of that class. The special cases of a class are known as subclasses. The same operation may behave differently on different classes - this is called *polymorphism*. The data within an object can be accessed only through its methods (*encapsulation*).

2.3 Object Modeling Technique (OMT)

As a general object model we use Rumbaugh et al.'s Object Modeling Technique (OMT), a specification and design approach for object-oriented systems [Rumb91]. Figure 1 shows the definition of a class with three separate areas: the *class name*, the *class attributes*, and the *operations* or *methods* of the class. In the following figures some of these details will be omitted if they are not of interest in their specific context.

The OMT model allows three basic types of associations between classes.

- **Generalization** implies the definition of a superclass that collects the common characteristics of several subclasses. It describes an "is-a" association between a subclass and its superclass. For example, in Figure 2 the superclass **Person** is a generalization of the classes **Faculty**, **student** and **Staff**; **Student**, in turn, is a generalization of **Foreign-Student**. The symbol for generalization is a triangle.
- **Aggregation (Composition)** implies the description of a class in terms of its constituent parts. The concept of aggregation defines an "is-a-part-of" relationship between a subclass and its

superclass. For example, Figure 3 shows a university composed of colleges, which in turn are made up of departments. Aggregation is denoted by a diamond. The black dots indicate multiplicity, e.g. a university is composed of several colleges, a college is composed of several departments.

- **Relationship** describes how objects belonging to different classes relate to each other. The OMT symbol for association is a line connecting two classes labeled with the association name. Relationships can be binary, ternary, or even higher order, and can have any number of attributes. For example, **Student** can enroll in courses, and **Faculty** teach courses (Figure 4). Relationships can have attributes, e.g. grade in Figure 4.

The complete modeling methodology consists of three orthogonal stages:

1. **Object Model** : shows the static data in the system. It divides the application into object classes. The relationships among those classes are described by the class structure. The behavior of the objects are defined by the methods or operations associated with the object classes.
2. **Dynamic Model** : shows the time-dependent behavior of the system. It shows the way the system behaves with internal and external events. An event trace diagram explains the event transformations between system classes and external effects.
3. **Functional Model** : shows how output values are derived from input values, or how to process the data flow in the system during each event or action.

III. CASE STUDY: A FLEXIBLE MANUFACTURING SYSTEM

In this section, we first develop a FMS which consists of robots, assembly stations, storage devices and conveyors. A particular assembly sequence is defined for a selected product. The object structure, dynamic model, and state model are developed for the initial system using the OMT method. Then the system is modified to have different layouts and to handle different products. We show

how little change is needed in our original design to adapt the software for the new system.

3.1 Initial Definition

Consider the manufacturing system shown in Figure 5. *SI* is a storage bin from where Robot *R1* picks up parts. The parts are deposited in a conveyor belt *CB1*, able of start-stop operation. When camera *C1* detects a part, robot *R2* picks it up and deposits it in rotary table *T*. Here Robot *R3* drills a hole in the part and paints it. Assume that not all robots of type *R3* can handle this type of part (special tools are needed). When finished, *R3* signals *R2* to put the part in conveyor belt *CB2* which is moving at constant speed and takes it to the packing section. There are *t* work cells.

Figure 6 is an OMT diagram for this system. Some attributes and operations are included for each object. The complete manufacturing system is described by class **Manuf_System**. Operations in this class indicate the possibility of activating or deactivating the complete system. There could be specific instances (objects) of the manufacturing system tailored to handle different types of parts. The complete system is shown as composed of three units or subsystems. One of these is the assembly cell. This cell can have its own local control and can be activated or deactivated by the manufacturing system. Note that there are in general several instances of each cell (*t* in this example) for each instance of the manufacturing system. The other subsystem is the distribution unit that describes the common equipment used to load and unload the assembly cells. In general there is only one distribution unit for the whole manufacturing system.

The next aggregation level describes the components of these units. The distribution unit is composed of one storage unit, 2 conveyor belts, and a robot. The assembly cell is composed of 2 robots, 2 cameras, and one table. There are 2 types of robots in the complete system: load robots, which can only grasp objects, and assembly robots, which can perform some specific operations (but cannot grasp parts). These two types of robots are generalized to **Robot**, which factors out common characteristics such as serial number, manufacturer, etc.

The relationships indicate joint coordination of units, for example a camera works with the load robot to signal it when it should pick up a part from

the conveyor belt *CB1*. Parts are related to the assembly cells. The *many-to-many* relationship here indicates that a part can be handled by many cells and that a cell can handle different parts.

Additionally, the following remarks apply to this design:

- Can-assemble can be deleted if one assumes all stations can assemble any part.
- Cameras need to be related to their robots, the other units are related by their physical position, i.e. a given load robot "knows" where to pick up its part because of the physical structure of the cell.
- One can generalize **Robot**, **Conv_belt**, and **Table** to **Machinery** but it is not important for this application. (It would be useful for a maintenance application).
- The sequencing of the system is in the implementation of the activate operation. For example, the activate operation in **Manuf_system** would first initialize the system, then activate the assembly cell and periodically check status. It would also close the system in case of a malfunction.

3.2 Dynamic Model of the Initial Definition

Figure 7 shows the dynamic model of this system. This is a composed of subdiagrams for each unit. We show here only the robots because their behavior is more complex than the other units. Robot *R1* repeats a sequence of picking-move-place to deposit parts on the conveyor belt. If the bin becomes empty it goes to a wait or idle state. Robot *R2* starts in the idle state until informed that a part is available on *CB1* when it starts a sequence of pickup-move-place that deposits the part on the table. It goes then into a wait state until the part is completed when it repeats the grip cycle to put the part in *CB2*. Robot *R3* waits for a part to be placed on the table after which it paints it and drills a hole. When finished it returns to its wait state.

Exceptional conditions can be conveniently shown in this model. When a robot drops a part we assume that assembly can continue normally (except that we may have lost some time). However, when

there is a mechanical malfunction in a robot we go to a final state which deactivates the complete cell.

Figure 8 shows a timing diagram of the interaction between units. This is convenient to study synchronization aspects.

IV. ADAPTING TO CHANGING CONDITIONS

There are many sources of changes in an industrial environment. We may need to change the configuration of the assembly cells for cost or efficiency reasons. We may want to change the operations performed by the cell, we may need to accommodate a larger variety of parts, etc. In order to operate more effectively we may also want to expand the computer controls to include other applications such as maintenance, inventory, order processing, etc. To be precise we will call the ability to adapt to changing conditions *modifiability*, and the ability to extend the realm of applications *extensibility*. By using the example of section III we will show that the object-oriented approach provides for both modifiability and extensibility.

Possible changes to the initial system are:

1. Make *R3* a more advanced robot able to grip and move its arm in addition to performing work on pieces.
2. Add an additional storage bin so that *R1* can pick up pieces from it when the first parts bin is empty.
3. Add another robot *R4* to accelerate the work of *R3*, e.g. *R3* could paint a part while *R4* drills a hole in it.

Possible extensions are:

1. Add a maintenance application that keeps track of maintenance schedules, technicians assigned to machines, etc.
2. Add an inventory application that will keep a list of available parts and will order parts from the lowest cost supplier when their inventory is low.

The changes can be accommodated in the following way:

1. Now robots of type *R3* (advanced-robot) become a subclass of hand robots since they have all the functions of those plus some additional functions (Figure 9).
2. This requires to add another instance of storage-bin and change the state diagram of *R1* to pick up parts from the second bin once the first one becomes empty.
3. This change requires to add another instance of *Assembly_Robot*.

The extensions can be handled in the following way:

1. Generalize all the machines to a superclass *Machinery* and include there all the information needed for maintenance, e.g. serial number, manufacturer, etc. Describe technicians as a subclass of a class of employees and indicate by means of a relationship who is in charge of what machine. All this is shown in Figure 10. In this figure we assume that a technician can handle more than one machine and a machine can be handled by more than one technician.
2. This is shown in Figure 11. We only show here the relationship between suppliers and parts. We leave as an exercise for the reader to complete this solution.

V. DISCUSSION

This example (and many in [Rumb91]) shows the modeling power of the object approach. Modeling can be performed at the logical level of the application, leaving out implementation details which are encapsulated in each object. A clear advantage of this is that implementation changes do not affect the logical level.

A very valuable practical advantage is the easiness to accommodate changes. The 3 cases shown here required minimal changes or no changes to the OMT class diagram. As shown in the extension examples existing classes can be combined with new classes to add new applications. This implies a gradual growth instead of the need to start from scratch as it happens in other methodologies.

From a CIM viewpoint the state diagrams, while conceptually clear to define object interactions,

are not convenient when timing aspects must be considered. Also conditions such as deadlock, starvation, etc., necessary to evaluate a concurrent system such as this, cannot be conveniently studied. Petri nets, stochastic Petri nets, and other approaches can be used to complement this analysis [Kura92].

There exist a good number of development tools to help building class diagrams. The authors of OMT have developed OMTTool and several CASE tools include support for object-oriented development. These tools can convert the OMT class diagram into a set of C++ definitions.

Another convenient aspect of this approach is the ability to describe exception states in the state diagram. This is useful to analyze error and abnormal conditions of any type.

VI. CONCLUSIONS

We believe we have made a strong case for the use of object-oriented methods in CIM. As indicated earlier, other studies have not focused in modeling aspects. While we used a specific approach, i.e. OMT, any similar approach, e.g. [Booc91, Mona92] would be appropriate.

It would be also possible to extend this approach to simulation. One must then make all the implicit relationships explicit, e.g. the actual physical position of a machine cannot be used as an implicit relationship as discussed above. The limitations of the method, i.e. lack of process structure, timing data, etc. must be taken into consideration.

References

- [Booc91] G. Booch, *Object-oriented design with applications*, The Benjamin/Cummings Publ. Co., 1991.
- [Grah91] J.H. Graham, S.M. Alexander, and W.Y. Lee, "Object-oriented software for diagnosis of manufacturing systems," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, April 1991, pp. 1966-1971.
- [HsuC92] C.L. Hsu, "Flexible manufacturing system controller software development by object-oriented programming,"

Proceedings of the Second International Conference on Automation Technology, July 4-6, 1992, Taipei, Taiwan, R.O.C., pp. 53-59.

- [Kura92] V. Kurapati, and E.B. Fernandez, "Object-oriented simulation and control of flexible manufacturing systems using Petri nets and Ada," FAU Tech. Rept. TR-CSE-92-33, Nov. 1992.
- [Mona92] D.E. Monarchi, and G.I. Puhr, "A research typology for object-oriented analysis and design," *Comm. of the ACM*, 35, 9, Sept. 1992, pp. 35-47.
- [Rumb91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-oriented modeling and design*, Prentice Hall, Englewood Cliffs, NY, 1991.
- [Stur91] M.C. Sturzenbecker, "Building an object-oriented environment for distributed manufacturing software," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, April 1991, pp. 1972-1978.
- [Spoo86] D.L. Spooner, M.A. Milicia, and D.B. Faatz, "Modeling mechanical CAD data with data abstraction and object-oriented techniques," *Proceedings of the International Conference on Data Engineering*, IEEE 1986, pp. 416-424.

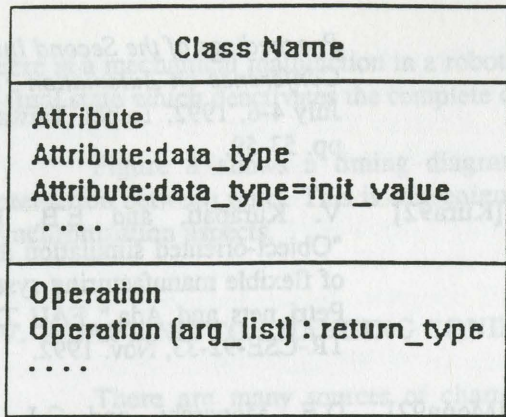


Figure 1 A Class Definition in OMT

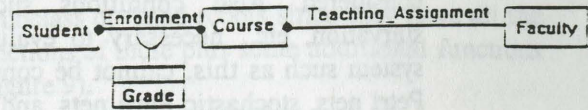


Figure 4 OMT Diagram for Association

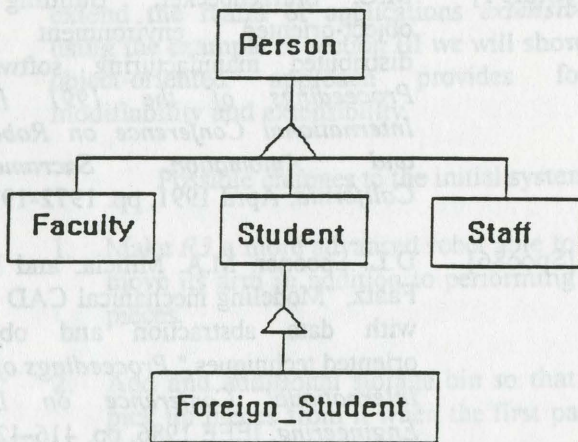


Figure 2 OMT Representation of Generalization

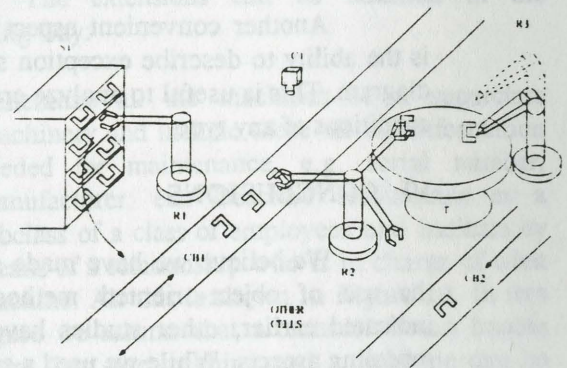
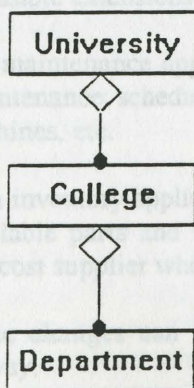


Figure 5 The Example FMS

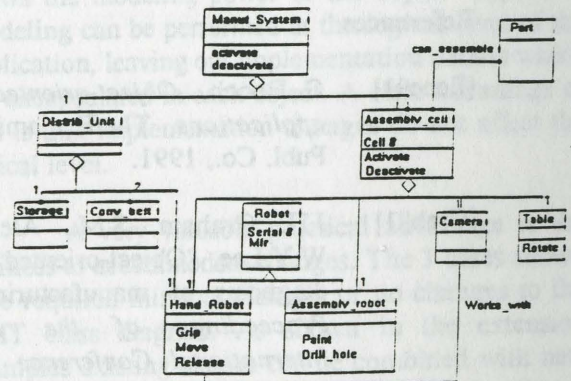


Figure 6 OMT Class Description of the Example

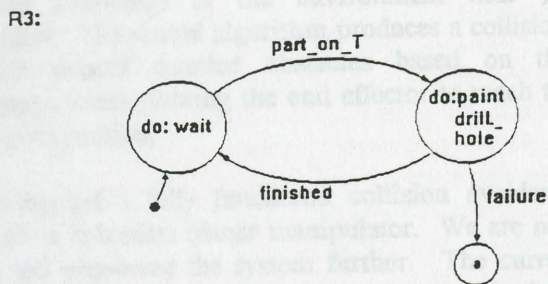
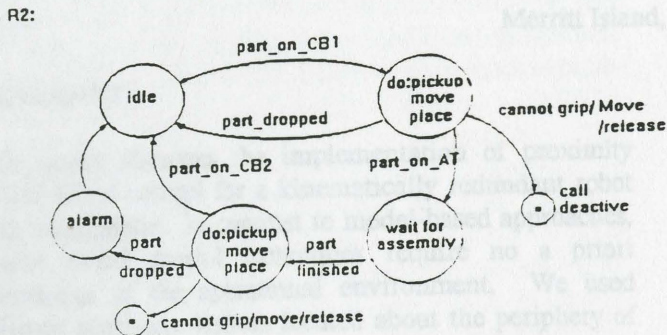
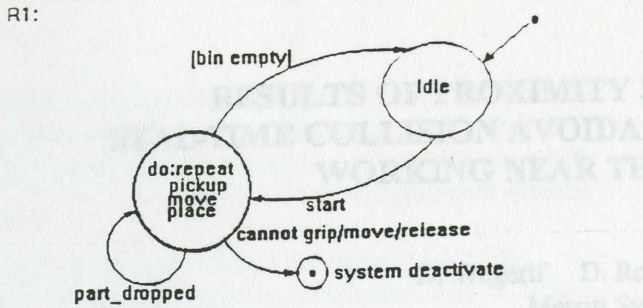


Figure 7 State Diagram of Robots in the Sample System

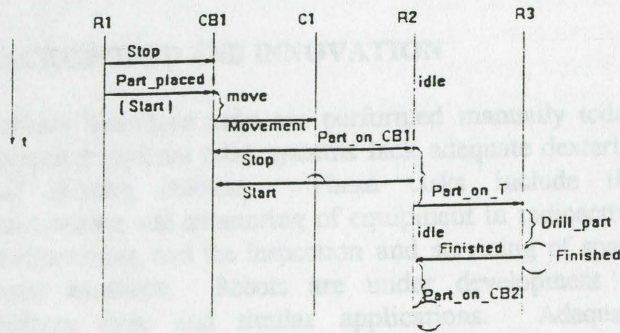


Figure 8 Timing Diagram for Example

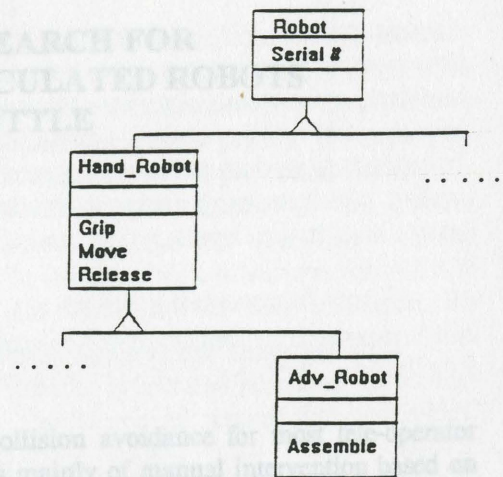


Figure 9 New Robot Hierarchy After a Change

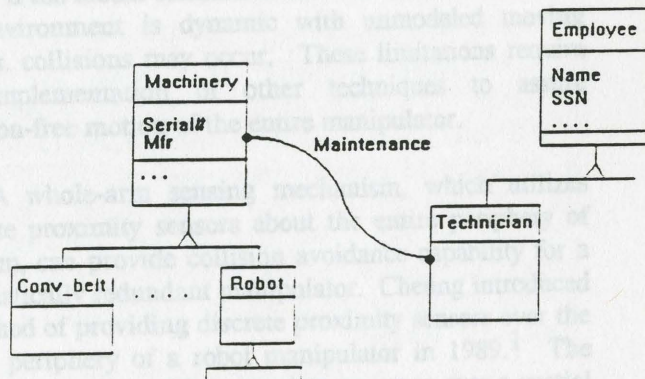


Figure 10 Maintenance Application

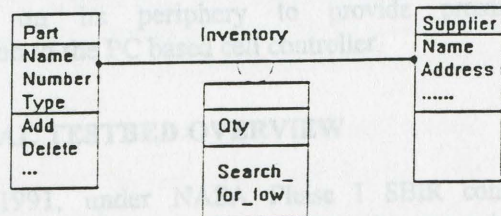


Figure 11 Inventory Classes

RESULTS OF PROXIMITY SENSING RESEARCH FOR REAL-TIME COLLISION AVOIDANCE OF ARTICULATED ROBOTS WORKING NEAR THE SPACE SHUTTLE

D. Wegerif D. Rosinski W. Parton
Merritt Systems Inc.
P. O. Box 2103
Merritt Island, Florida 32954

ABSTRACT

This paper discusses the implementation of proximity sensor based control for a kinematically redundant robot arm manipulator. In contrast to model-based approaches, sensor based control techniques require no a priori knowledge of the operational environment. We used infrared proximity sensors located about the periphery of a three degree of freedom planar mechanism to provide real time knowledge of the environment near the manipulator. The control algorithm produces a collision-free path around detected obstacles based on this information, while allowing the end effector to reach the desired goal position.

We constructed a fully functional collision avoidance system for a redundant planar manipulator. We are now testing and improving the system further. The current testbed incorporates a PUMA-600 robot manipulator operating in a plane with a sensor "skin" composed of 14 infrared sensorCells about its periphery. Each sensorCell incorporates eight sensing elements, a distributed processing electronics system, and supporting communications hardware providing real-time control of the robot system. A standard desktop computer serves as the process controller. This work has been extended to redundant spatial manipulators under a NASA Phase II SBIR research grant.

BACKGROUND AND INNOVATION

Various hazardous tasks are performed manually today because traditional robot systems lack adequate dexterity and sensing abilities. These tasks include the maintenance and monitoring of equipment in radioactive environments, and the inspection and servicing of space flight hardware. Robots are under development to perform these and similar applications. Adequate collision avoidance techniques to fully utilize the capabilities of these robots have not yet been developed, however.

Existing collision avoidance for most tele-operator systems consists mainly of manual intervention based on visual observation of the robot. Constrained environments, however, preclude direct view in some configurations. This prevents an operator from detecting obstacles visually. Autonomous motion planners using model based control strategies were developed, however, they provide collision-free motions only if the model accurately matches the actual working environment of the robot. If the model contains even small inaccuracies, or if the environment is dynamic with unmodeled moving objects, collisions may occur. These limitations require the implementation of other techniques to assure collision-free motion of the entire manipulator.

A whole-arm sensing mechanism, which utilizes discrete proximity sensors about the entire periphery of the arm, can provide collision avoidance capability for a kinematically redundant manipulator. Cheung introduced a method of providing discrete proximity sensors over the entire periphery of a robot manipulator in 1989.¹ The original system provided complete coverage over a spatial positioning (3 revolute) robot. In 1991, MSI extended this capability to a kinematically redundant planar robot, and successfully demonstrated a proximity sensor-based control methodology on a redundant planar manipulator.² A motion planning algorithm, developed by MSI, utilizes the kinematic redundancy of an articulated robot to avoid obstacles while maintaining the desired end-effector trajectory. This capability is currently being demonstrated on a new testbed incorporating a PUMA 600 robot manipulator and 14 infrared sensorCells mounted on its periphery to provide proximity information to the PC based cell controller.

ORIGINAL TESTBED OVERVIEW

In 1991, under NASA Phase I SBIR contract NAS10-11754, MSI investigated proximity sensor-based motion control for kinematically redundant robot

manipulators. We accomplished this through the development of a demonstration testbed, a graphical user interface, and a control algorithm that combined real-time sensor data with desired end-effector positioning commands to produce the necessary joint velocities. This section will describe the original development platform, sensor system, user interface, and control algorithm.

Hitachi Demonstration Testbed

MSI assembled the original testbed to verify the ability of proximity sensors to control a kinematically redundant manipulator in real time. MSI selected a planar robot, the Hitachi 4010H, SCARA-type manipulator, as the Phase I demonstration platform. As with most SCARA manipulators, the 4010H is a serial RRRP (R -- revolute, P -- prismatic). With a link attached to the wrist joint, a total of three revolute links in a plane were available. The robot was able to meet any specified position of the end-effector in the workspace with an infinite number of configurations of the three links.

A 33 megahertz clock-rate, 80386 microprocessor-based, personal computer (PC) served as the cell controller. A standard RS-232 serial port operating at a rate of 4800 baud supported communications to the robot. A dedicated 16 channel analog-to-digital (A/D) card in the PC converted the output of the sensor system into useful digital data.³ We used the four available digital output channels and one of the D/A channels to provide a five-bit address to the proximity sensor control circuits mounted on the robot.

The system software was written in Borland Turbo C, with five major software routines: 1) the main program, which included the robot control algorithm; 2) D/A & A/D card control software, which scanned the sensors and listed the location and magnitude of the highest sensor reading; 3) robot control software, which provided serial communication between the robot and PC, and converted the output of the main algorithm joint command from radians to stepper motor positional counts; 4) the supporting program for the main algorithm, which provided general vector and matrix manipulation functions; and 5) the user interface, with a graphical representation of the robot work space.

Sensor Skin and Electronics

We limited the objectives of this development project to demonstrating the ability of a proximity sensor-based approach for a kinematically redundant robot manipulator, and did not include the development of a

specialized sensor system. MSI selected reflected infrared energy as the sensing media. We used optically reflective materials as obstacles. Previous research done by Cheung was the basis for the sensor system.¹

MSI built sensing circuits for each of the three robot links. Each circuit card accommodated up to 32 sensor pairs, consisting of an IRED emitter and a PIN diode receiver. The circuit contained both an analog portion for amplifying and filtering the reflected light energy, and a digital portion for sequentially operating each of the IREDs.

While Cheung used a flexible Dacron mounting material, we chose a standard 40-conductor ribbon cable with 40-pin female connectors attached at approximately every 50 mm (two inches). The ribbon cable allowed a simple method of attaching the sensors to the robot links. We inserted an IRED and PIN diode into each connector. This provided complete flexibility in the location of the sensors along the robot periphery, and also permitted IREDs and PIN diodes to be easily changed or reconfigured to different addresses. We addressed the IREDs by bending the leads to fit in the appropriate socket of a female connector. This provided a simple, low-cost method of fabricating a strip of sensor pairs. The sensor strip was attached to the perimeter of the robot manipulator using double-sided adhesive tape.

User Interface

The user interface contained two elements. The first was a menu, which allowed the user to initialize the system and command specific responses from the robot and sensors. This enabled the user to observe raw sensor data, calibrate the sensors, and review the calibrated sensor data. The second element, the graphical interface, provided the user with an overhead view of the current configuration and the desired end-effector position. For the graphical display, line segments represented each link of the robot. An operator commanded the desired end-effector position of the robot by first moving the on-screen cursor with the mouse to the desired position within the robot workspace, and then depressing the left button on the mouse to fix the desired position. Obstacle detection illuminated a visual cue on the display at the point representing the sensor location on the robot. The graphics refreshed each cycle of the software.

Control Algorithm Development

MSI investigated several control techniques, including both closed-form and heuristic algorithms, before selecting an approach. The ideal control algorithm

would enable the robot to detect obstacles, avoid collisions, and maintain a desired end-effector trajectory without any prior knowledge of the work space. The algorithm also had to be sufficiently compact and efficient for use in real-time on a PC-based cell controller.

The control algorithm implemented, based upon the 1985 work of Maciejewski and Klein,⁵ achieved the two goals of maintaining a desired end-effector velocity and providing obstacle avoidance in a dynamic environment. To do this, the algorithm combined the desired end-effector velocity with a calculated obstacle velocity to obtain the required manipulator joint velocities. The original algorithm gave highest priority to achieving the desired end-effector position, while using the available degree of freedom for obstacle avoidance. Our modification to their algorithm instead placed top priority on avoiding the obstacle, allowing movement to the desired end-effector position only if the path was clear.

As a result, the implemented algorithm supported three modes of operation, depending on the location of the closest obstacle to the arm. In Mode 1, when no objects were within range of the sensors, the arm simply moved to the desired end-effector position. If an object came into sensor range, the arm entered Mode 2, where the arm moved to the desired end-effector position, while avoiding the obstacle using the redundant degree of freedom. Finally, if the obstacle came too close, the arm switched to Mode 3, where it would move all joints necessary to avoid a collision. The switching between modes occurred transparently and smoothly, without any action by the operator. We based switching solely on the current sensor data. This led to an intuitively agreeable operation of the arm. A description of the algorithm follows below.

The end-effector velocity, x' , is related to the joint velocities, denoted by the n -dimensional vector Q' , where n is the number of degrees-of-freedom, by the equation

$$x' = JQ' \quad (1)$$

where J is the Jacobian matrix. In the case of redundant manipulators, the inverse of J is not defined because it is rectangular. Instead we calculate the pseudo inverse, J^+ , using the following equation:

$$J^+ = J^T (J J^T)^{-1} \quad (2)$$

The general solution to equation 1 is given by

$$Q' = J^+ x' + (I - J^+ J) z \quad (3)$$

where I is an n by n identity matrix, and z is an arbitrary vector in q' - space. Thus, we decompose the resultant joint angle rates into a combination of a least squares solution of the minimum norm, plus a homogeneous solution created by the action of the projection operator $I - J^+ J$ (which describes the redundancy of the system) mapping an arbitrary z' into the null space of the transformation. By the application of various functions of q to compute the vector z , we can configure the manipulator to achieve some desirable secondary criterion under the constraint of the specified end-effector velocity.⁵

The obstacle avoidance approach identifies, for each period in time, the point on the robot that is closest to an obstacle. We call this the obstacle point, and assigned to it is a desired directional velocity component away from the obstacle surface.⁵ This method is ideally suited for robot systems incorporating numerous proximity sensors about their periphery, where each individual sensor provides a discrete value representing the distance of that portion of the arm to the obstacle. By monitoring all of the sensors, it is relatively simple to determine the part of the robot that is closest to an obstacle at any time. The primary goal of the specified end-effector velocity and the secondary goal of obstacle avoidance are thus described by the equations

$$J_e Q' = x_e \quad (4)$$

$$J_o Q' = x_o \quad (5)$$

where: J_e - end-effector Jacobian
 J_o - obstacle point Jacobian
 x_e - specified end-effector velocity
 x_o - specified obstacle point velocity

We give the set of solutions to exactly satisfy the desired end-effector goal in the form of equation 3. Substituting this solution into equation 5, the result is

$$x_o = J_o Q' = J_o (J_e^+ x_e' + (I - J_e^+ J_e) z) = J_o J_e^+ x_e' + J_o (I - J_e^+ J_e) z \quad (6)$$

which can be solved for the desired homogeneous solution. Solving the above equation for z leads to the following equation:

$$z = [J_o (I - J_e^+ J_e)]^+ (x_o' - J_o J_e^+ x_e') \quad (7)$$

We substitute this result back into equation 3 to provide the desired solution satisfying both goals under the

constraints imposed by the number of available degrees-of-freedom, yielding:

$$Q' = J_e^+ x_e' + [J_0 (I - J_e^+ J_e)]^+ (x_0' - J_0 J_e^+ x_e') \quad (8)$$

Finally, adding a term to move the arm away from obstacles, and placing top priority on avoiding obstacles yields:

$$Q' = J_e^+ x_e' + [J_0 (I - J_e^+ J_e)]^+ (x_0' - J_0 J_e^+ x_e') + k J_0 (x_0' - J_0 J_e^+ x_e') \quad (9)$$

Each of the terms in the preceding formulation has a physical interpretation. In the redundant case, the pseudo-inverse solution of the desired end-effector trajectory $J_e^+ x_e'$ guarantees the exact desired end-effector velocity and minimum joint velocity norm. We use this matrix to transform the desired obstacle point motion from Cartesian obstacle velocity space into the best available solution in the joint velocity space, again through the use of the pseudo-inverse. The second component sacrifices the minimum norm solution to satisfy a different goal, that of obstacle avoidance using the redundancy. The matrix, composed of the obstacle Jacobian multiplied by projection operator $J_0 (I - J_e^+ J_e)$, represents the degrees-of-freedom available to move the link's obstacle point without moving the end-effector. We scale the third component by the parameter k to force the arm away from any sufficiently close obstacles, abandoning the desired end-effector position if necessary in order to avoid a collision.

We find the parameter k in equation 9 by setting a threshold on the sensor data from the skin. If the sensor reading associated with the obstacle point exceeds this threshold, we gradually increase the parameter k in proportion to the sensor reading, until we drive the arm away from the object. The vector describing the desired obstacle point motion is the commanded motion, x_0' , obtained from the sensor data, modified by subtracting the motion caused at the obstacle point by satisfying the end-effector velocity constraint $J_0 J_e^+ x_e'$.

We calculate the specified end-effector velocity, x_e , by taking the difference between the current and desired end-effector position. After determining which sensor on the manipulator has the highest reflected energy signal, we set the magnitude of the specified obstacle point velocity in proportion to this sensor's measured value. The direction of x_0 equals the sum of the joint angles of the arm, plus or minus 90 degrees, depending upon which side of the arm the sensor is fastened. Currently, we use

only the single point closest to the arm in the equation to develop desired joint velocities. In some instances, this may cause the manipulator to oscillate between two configurations. This would occur if we drive the arm out of the range of one obstacle, only to be left closer to another obstacle. During the next cycle, the manipulator might then be driven back toward the original obstacle. We can reduce this effect by monitoring several sensors simultaneously, which represents obstacles close to the manipulator, and adding their respective components to the homogeneous solution:

$$Q' = J_e^+ x_e' + \sum_{i=1}^n [J_0 (I - J_e^+ J_e)]^+ (x_i' - J_0 J_e^+ x_e') \quad (10)$$

The algorithm presented here meets all of the requirements developed for this application, and it is extendible to highly redundant planar or spatial robot manipulators. We demonstrated two control methods during system testing: repel and teleoperation. The repel condition maintains a desired end-effector position while the manipulator is exposed to dynamically moving objects. To accomplish this, we maintain the desired end-effector position while the links of the manipulator move to avoid colliding with the objects. In the teleoperator condition, meanwhile, the operator commands in real-time the desired end-effector position of the robot, and the robot moves to the desired end-effector position while avoiding obstacles detected by the sensitive skin. We successfully demonstrated the system in both conditions, with multiple, dynamic objects in both cases.

Results of Original Testbed

We built and demonstrated a fully functional collision avoidance system for kinematically redundant robot manipulators. The results indicated that it was feasible to use proximity sensor-based control of redundant manipulators in constrained environments. The infrared sensor system demonstrated a sensing range of approximately 8 cm (3 inches), and an ability to detect a wide variety of objects. The sensing technique depended on measurements of the amplitude of the energy reflected back from the object to the sensor-receiver. The cell controller polled the entire sensor system at an update rate of approximately 3 Hz, sampling each of the 49 sensor pairs 30 times and averaging the readings, in order to reduce noise in the system. The resulting sample rate proved adequate for the test robot.⁶

Thus, this novel sensor-based motion planning approach for kinematically redundant robots provided real-time guaranteed safe motion around detected

obstacles in a cluttered, partially modeled, or dynamic environment, while allowing the end-effector to reach the desired position in the workspace. Certain limitations of the sensing system were evident, however, including: a low sensor update rate; limited detection range, inability to detect some materials; and, a high level of noise in the analog circuit. Successful commercial implementation of this technique would require an improved method of detecting obstacles and processing sensor data.

The limitations of the sensing capabilities led to the development of the sensorCell, described in section 4, to provide a greatly enhanced method for obtaining the proximity data. The development of a new testbed incorporating a closed-loop servo controlled robot, an improved implementation of the control software using an object oriented language (Borland C++), an improved cell controller (a 50 MHz 486 PC) and 14 infrared sensorCells is providing an excellent demonstration platform.

RELATED RESEARCH

MSI completed a related investigation into improved sensing capabilities entitled Sensor Technology for Robotic Obstacle Avoidance, under NASA Phase I SBIR contract NAS10-11860 in July 1992.⁴ The primary purpose of this project was to evaluate the ability of several different sensing media to detect materials found on or near the Space Transportation System space shuttles. We constructed and tested infrared, capacitive, and ultrasonic sensors. Improved methods of obtaining and processing sensor data resulted in the development of a distributed processing electronics system, termed a sensorCell.

Sensor Capability Study

During this project, we constructed and tested fully functional infrared (IR), ultrasonic, and capacitive proximity sensors. We developed a special test fixture and software to collect data on the detection capabilities of the three sensor types. The design of the fixture allowed for testing material samples in both the X (displacement) and Y (direction) coordinates. This provided performance data on both range and directional sensing capability. We designed and built a prototype distributed processing electronics system to support operation of each sensor.

The testbed infrared sensor was able to detect all of the materials tested at a minimum range of 10 cm (4 inches). MSI observed a broad angle of sensitivity, which would support overlapping coverage in a multi-sensorCell

environment. The later version of the sensorCell was also able to detect ceramic coated space shuttle insulation material at approximately 20 cm (8 inches), also an improvement over the testbed configuration.

The ultrasonic sensor likewise was able to detect each of the material samples. Minimum detection range was 8.25 cm (3.5 inches), with a detection zone width about 10 cm (4 inches). The width of the target detection beam increased with the distance to the target, indicating a conical beam shape, as expected. The data also showed that the sensor easily detected the tile over the maximum range of the testbed, approximately 50 cm (20 inches). However, the magnitude of the response signal, indicated by the number of counts, did not vary significantly, i.e., the sensors positively detected the target, but could not make an accurate determination of relative range.

Although able to detect all test materials, the capacitive sensor, based on the capaciflector,⁷ exhibited the most limited detection range of the test systems. The range varied from 3.75 to 7.5 cm (1.5 to 3 inches). Within these ranges, the magnitude of the measured changes was extremely low and could present detection problems in noisy environments. The material properties of the target materials and/or the specific implementation technique may have caused this relatively poor performance. Also, the size and composition of our test materials may not be conducive to long range capacitive detection.

Sensor System Hardware Improvements

The sensorCell evolved from the need to minimize the amount of robot cell controller computational overhead caused by collision avoidance data handling and analysis. Distributing the processing intelligence over many dedicated micro controllers appeared to provide an optimal solution. We sought to minimize cost, leading to multiple sensors sharing the same microcontroller hardware. We also required an efficient method of reducing and transmitting the resulting data due to the large number of sensors to instrument an entire robot manipulator. Additionally, we deemed as necessary a robust communication capability between sensorCells and the host computer, support for multiple sensor media, and a high level of fault tolerance.

The equipment designed and built during the project satisfied all of the above requirements. We selected reflected IR light amplitude as the sensing technique for the prototype sensorCells. Localization of the sensory hardware to each sensorCell limited analog signal distribution to the sensorCell itself. We converted the

analog object proximity information to a digital form extremely resistant to transmission noise. As a result, we eliminated previous problems with signal noise between the sensors and robot cell controller.

The use of digital communications between sensorCells, and between the sensorCells and the controller, also allowed development of a standard for the presentation of proximity information. The standard interface made the sensory mechanism of the individual sensorCells transparent to the robot controller, enabling the concurrent use of different sensing media on neighboring sensorCells. The result is that each sensorCell appears generic to the controller system. We developed a novel communications architecture to pass commands and information between the controller and the sensorCells. This design allows the controller to command all of the sensorCells simultaneously, and permits immediate communication of proximity alarms to the controller from the sensorCells. Additionally, each sensorCell has a unique address, even though it may be otherwise identical to all other sensorCells. Controller coordination of scanning and identification of the source of proximity information and alarms requires this unique address. The design supports a two-dimensional array of sensorCells, 32 cells wide by 32 cells long with a maximum of 1023 sensorCells in a single system.

Under the Phase I contract we constructed and tested three fully functional IR sensorCell prototypes. Each sensorCell contained two emitters and eight receivers arranged in two approximately square-shaped elements. Operational characteristics of individual sensorCells were measured. We observed significant detection range increases were observed over the initial development prototype version, which incorporated a single sensing element. The software and firmware successfully implemented the described communications protocol enabling redundant, fault tolerant data transmission between the sensorCells and the control computer. Scan time was 10 ms for a single board with no obstacles present. This corresponded to a single cell scanning frequency of 100 Hz in the preferred mode. The backup communications mode scan time was 12 ms.

CURRENT WORK

MSI is currently conducting work under NASA Phase II SBIR contract NAS10-11910 to develop and implement a fully functional proximity sensor skin and control system for a Robotics Research 1207 manipulator. This two-year effort includes: extension of the control algorithm to seven degrees of freedom; improvements in

sensor detection range; construction of a sensor skin containing a distributed array of infrared sensing devices and related electronics; and, installation and testing of the system on a Robotics Research 1207 device. We are presently performing capability testing and performance optimization using a new demonstration platform. It incorporates a PUMA 600 articulated robot and 14 upgraded sensorCells mounted about the robot's periphery. For test purposes, we configured the robot to provide a single kinematic redundancy with three axes in a vertical plane by fixing the position of the fourth axis so that axes' two, three and five are parallel. We added a short link to the tool mounting flange to provide a third planar link.

We installed an improved version of the sensorCell, designated revision B, for optimization and testing. Revision B devices provide improved sensing capability and increased control of the sensorCell system. Improvements include: restructuring of the sensorCell command set, modification of the analog circuit, and a reduction in the current draw of each sensorCell. We modified the initial command set to handle up to 16 sensors on each SensorCell. We retained the ability to support addressing of 1023 sensorCells, however, we introduced the concept of the sensorCell 'set'. We can now assign each sensorCell to one of 256 possible sets. Sets can be enabled or disabled to allow, for example, only sensorCells facing in the direction of travel to be perform scans. This innovation greatly improves the effective scan rate. The revised instruction set permits complete sensing operation using only one of the two busses in the event of a failure. This adds another degree of redundancy and makes the system more fault tolerant.

Optimization of the analog circuitry component improved performance and ensures that each sensorCell operates within prescribed standards. We made other changes to the analog to digital converter circuitry to improve sensitivity. We have also implemented a detailed circuit testing and acceptance program. We test and adjust each sensorCell to provide uniform performance. These changes increased detection range substantially. The sensorCells now consistently detect standard reflective targets at up to 35 cm (14 inches).

We also implemented hardware changes to reduce the sensorCell current requirements. The first generation sensorCell employed an older communication IC that resulted in a current draw of approximately 200 milliamps per device. We replaced these components with a more efficient communication IC. The current sensorCells require approximately 25 ma at 7 volts for

operation. The entire 120 sensor skin requires less than 400 ma in contrast to the 3 A required by the earlier devices.

A standard 80486 PC now serves as the cell controller. The control system uses two serial ports to transfer sensor data, commands to the sensorCells, and communications between the PUMA and the PC. The robot downloads current joint positions to the cell controller each command cycle. The cell controller uses desired end-effector position and the sensor data in combination with the current robot position to create the new desired joint commands. The cell controller sends joint commands to the robot every command cycle. The configuration provides a real-time update rate. We plan to mount additional sensorCells to the robot to provide sensing capability out of the plane and enable spatial motion control of the robot. The goal of this evolutionary process is to extend the sensing techniques to spatially redundant robots such as the Robotics Research 1207.

As of this writing, the testbed system performs approximately 7 complete sensor skin samples per second over our 120 sensor skin. Testing is in progress to implement a methodology of multiple scans. Simultaneously scanning multiple sets of sensorCells effectively reduces the time to cover the entire systems of sensors. We are also investigating the feasibility of other scanning methods using heuristics to select optimum sensing patterns for operating conditions. Interference between sensorCells is currently the limiting factor to increased scanning frequencies.

MSI and the Department of Energy, through the Oak Ridge National Laboratory, are discussing funding in fiscal year 1993 to investigate application of this technology to robotic missions in the Environmental Restoration and Decontamination and Decommissioning areas. A possible task would be the construction of a system based on the PUMA prototype for use as a technology demonstration device. MSI is pursuing this work through the Robotics and Process Systems Division at the Oak Ridge National Laboratory.

MSI has also proposed to NASA the development of a collision avoidance system for a serpentine robot manipulator currently under development by Foster Miller as a deliverable for a Phase II SBIR to begin in 1993. The proposed system would consist of a flexible skin of distributed intelligent sensors based on the sensorCell described above. Each intelligent sensor would be a standalone sensing and processing device. We proposed optimizing sensor media composition and distribution for

the Shuttle Processing work environment by the performance of an extensive sensor test program.

TECHNOLOGY APPLICATIONS

Currently, four robotic systems are under development for applications at the Kennedy Space Center: the Automated Radiator Inspection Device (ARID), the HEPA Filter Certification Robot (HFCR), the Robotic Tile Processing System (RTPS), and a serpentine robot to perform pre-flight close-out tasks. Because collisions between the robot and flight hardware are unacceptable, whole arm sensing technologies are necessary. In many cases, a proximity sensor system may be the preferred technique for preventing collisions. Similar sensing technology is currently under development at the Goddard Space Flight Center for future autonomous satellite servicing⁷ and for hazardous and radioactive materials handling systems for the Department of Energy at the Sandia National Laboratories and Oak Ridge National Laboratory.⁸ MSI has submitted a Phase II SBIR proposal to NASA for development of a functional proximity-sensor skin for the serpentine robot being constructed by Foster-Miller Inc.

We have identified several large commercial applications for non-tactile, proximity sensor-based obstacle detection and avoidance systems. These applications include the seven vertical case preparation systems designed and built by Vadeko International Inc. for the NASA Solid Rocket Motor Manufacturing Facility in Iuka, Mississippi. A system for the robotic painting and maintenance facility currently being designed to service United States Air Force aircraft is also under consideration. This later system requires a collision avoidance system to ensure that the robot does not strike critical flight surfaces of advanced aircraft. Vadeko identified MSI as the source of the collision avoidance function for this system.

Other commercial applications exist in manufacturing. The widest application may provide enhancements to a large number of existing robotic work cells. Instead of the current intrusion protection techniques (fences, pressure sensitive pads) used to restrict humans from being struck by robots, the robots could be instrumented with sensorCells to allow humans to work safely in the same area. Integrated man and robot work cells would utilize the dexterity, finesse, and intelligence of a human with the strength, accuracy, and endurance of a robot. This capability has the potential to greatly improve the productivity of numerous manufacturing processes.

CONCLUSIONS

The results of the research show that it is feasible to use proximity sensor-based control of redundant manipulators in constrained environments. We base our conclusion on the successful demonstration of a fully functional planar testbed developed by MSI. We have demonstrated two control methods, repel and teleoperation. The control algorithm is extendible to both planar and spatial manipulators with any number of redundancies. It converges asymptotically to the desired endpoint position and remains stable until a sensor detects another, closer object. The manipulator then moves to avoid the closest object while maintaining the desired end-effector position.

Existing proximity sensing technology appears adequate to support construction of a practical collision avoidance system for installation on both new and existing robot manipulators. We found three proximity sensing techniques, infrared, capacitive, and ultrasonic, capable of consistently detecting Shuttle Orbiter materials. Successful deployment requires several technical enhancements. Ultrasonic sensing exhibited exceptional absolute detection ranges, but lacked sufficient resolution to allow relative range determination. Infrared reflected amplitude provided good relative range information, but over a limited absolute range. We found the capacitive technique limited in both areas. The test results indicate that an integrated infrared and acoustic proximity sensing system would be the most effective means of detecting materials found on or near the Orbiter for ground-based applications.

We have demonstrated the sensorCell distributed processing concept to be a feasible method for controlling distributed proximity sensors. Using infrared media, these sensorCells presently have an effective detection range of over 30 cm. Each sensorCell can perform an autonomous scan of all 8 sensors in 10 milliseconds, corresponding to an effective sensor scan frequency of 800 Hz. We have achieved a scan frequency of approximately 7 Hz for a complete sensor skin of 120 sensors. We expect innovative techniques to allow demonstration of operational frequencies of 24 to 48 Hz in the near future. Extensive testing of second generation sensorCells continues to identify improvements in design and operation.

ACKNOWLEDGMENTS

This work was performed under NASA SBIR contracts NAS10-11754, NAS10-11860 and NAS10-

11910. We wish to thank the following NASA employees for both technical and administrative support: Todd Graham, Robert Lewis, Gabor Tamasi, Eric Rhodes and Jim Aliberti. Additionally, we would like to thank Mr. Jim Whitehead for his thorough technical reviews.

REFERENCES

1. E. CHEUNG, "Real-Time Motion Planning for Whole Sensitive Robot Arm Manipulators," Ph.D. Dissertation, Yale University, New Haven, CT., (1990).
2. D. WEGERIF, et.al., "Sensor Based Whole Arm Obstacle Avoidance for Redundant Robot Arm Manipulators," Phase I Final Report for NASA Contract NAS10-11754, Merritt Systems, Inc., Merritt Island, FL, (1991).
3. *Keithly Metrabyte Corporation User Guide for the DAS-16F*, Revision C, Taunton, MA, (August 1990).
4. B. PARTON, et.al., "Sensor Technology for Robotic Obstacle Avoidance," Phase I Final Report for NASA Contract NAS10-11860, Merritt Systems Inc., Merritt Island, FL, (1992).
5. A. MACIEJEWSKI, C. KLEIN, "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments," *The International Journal of Robotics Research*, Vol. 4, No. 3, MIT Press, 1985.
6. D. ROSINSKI AND D. WEGERIF, "Sensor-Based Whole-Arm Obstacle Avoidance for Kinematically Redundant Robots," SPIE OE/Technology '92, Boston, MA, (November 1992).
7. J. VRANNISH, R. McCONNEL, S. MAHALINGAM, "'Capiciflector' Collision Avoidance Sensors for Robots," Draft Copy, Goddard Space Flight Center, MD, (1991).
8. A. WHITENBERD, et.al., "Sensor-Based Whole-Arm Obstacle Avoidance for Unstructured Environments," ANS/ENS Conference, Chicago, IL, (Nov. 1992).

This paper has been accepted for presentation at the Fifth Topical Meeting on Robotics and Remote Handling, sponsored by the American Nuclear Society.

AUTONOMOUS OBSTACLE AVOIDANCE USING VISUAL FIXATION AND LOOMING*

Kunal Joarder and Daniel Raviv

Robotics Center and Department of Electrical Engineering
Florida Atlantic University
Boca Raton, Florida 33431

ABSTRACT

This paper deals with a vision-based method for avoiding obstacles using visual looming and fixating motion. Visual looming is the result of expansion of objects in the retina. Usually, this is due to a decreasing distance between the observer and the object. An increasing looming value signifies an increasing threat of collision with the object. Also, a purposive control of visual fixation at the objects in front of the moving camera simplifies the visual task of avoiding them. Using these two basic concepts we implemented real time obstacle avoidance in a tight perception-action loop. 3D space in front of the camera is divided into zones with various degrees of looming-based threat of collision. For each obstacle seen by a fixating camera, looming and its time derivative are calculated directly from the 2D image. Depending on the threat posed by an obstacle, a course change may be required. This looming based approach is simple, independent of the size of the 3D object and its range and involves simple quantitative measurements. Results show a camera on a robot arm navigating between obstacles.

1. INTRODUCTION

How do we navigate in a world full of stationary and moving obstacles? How do we avoid collisions with furniture and moving people when we walk from one end of the room to the other? How does a driver manipulate his speed and the course of direction during driving a car in a heavy traffic?

We argue that the visual looming effect, i.e., the expansion of objects' size in the retina is essential to all the above. Usually, an object's size as projected on the retina expands if the distance between the observer and the object decreases. This visual looming carries an indication of a possible collision with that object. The observer may react defensively to reduce this visual threat. Following this approach it is possible to explore navigation problems, in particular the obstacle avoidance problem, for an autonomous mobile robot. For navigation, reconstruction of 3D world is computationally expensive and may not be needed. In many situations a task-dependent, tight 2D based perception-action loop can be adopted [1-4].

The problem of obstacle avoidance by autonomous mobile robots has received various approaches from researchers in the computer vision area. Usually the task is to drive the robot from a starting position to a destination through multiple obstacles. In some cases an on-board sensor is not used and the environment is assumed to be known. An approach of global optimization to generate a

trajectory as proposed by Gilbert[5], Shin[6], and Kim and Shin[7] requires intensive computation. Introduction of artificial potential fields that attract the robot to the goal and repulse the robot from obstacles is novel. Khatib[8], Hogan[9], Krogh[10,11], Newman and Hogan[12] and Volpe and Khosla[13] adopted this line. However it is difficult to implement this approach in the conventionally steered mobile robots. Nelson and Aloimonos[14] introduced an approach, in which the flow field divergence is used as a qualitative measurement for obstacle avoidance.

In this paper, we use the visual parameter *looming* in servoing a camera attached to the end-effector of a six-degree-of-freedom robot through multiple spherical obstacles. This parameter is calculated *directly* from a sequence of 2D images obtained from an on-board camera. It is shown that looming, a vital visual cue, when used actively by continuously changing the points of attention, can make it possible to navigate in a fairly complex environment. Our approach does not assume any knowledge on the velocity of the camera and the size and range of the obstacle. Also as mentioned previously looming is a measurable parameter. Hence the idea is simple and readily implementable.

First, we provide a summary of the previous results on looming including definition and methods of measuring it. Visual fixation and looming as a function of time is covered next. Finally, the obstacle avoidance algorithm and its implementation are discussed followed by results.

* This work was supported in part by a grant to Florida Atlantic University from the National Science Foundation, Division of Information, Robotics and Intelligent Systems, Grant # IRI-9115939.

2. LOOMING - SUMMARY OF PREVIOUS RESULTS

Looming deals with expansion of the object's image on the retina. Usually this expansion is caused by a decreasing distance over some period of time between the observer and the object. Looming has been studied mostly qualitatively by the psychologists[15-19]. Recently a quantitative approach to looming has been presented by Raviv[20,21]. The looming value of an object is proportional to the relative translational velocity between the observer and the object divided by the distance itself. This measurable variable can be extracted *directly* from a sequence of 2-D images using optical flow or relative change in area of the image of the object. The concept of looming is also similar to Nelson and Aloimonos' Flow Field Divergence[14].

2.1 Mathematical Definition of Looming :

The looming value L of a very small 3D object is defined as the negative value of the time derivative of the relative distance between the observer and the center of the object divided by the relative distance[21] :

$$L = - \frac{\frac{dR}{dt}}{R}$$

The negative sign is used to signify image expansion with positive looming. The unit by which L is measured is [time⁻¹].

2.2 Equal Looming Surfaces :

Are there points in 3-D space that result in the same looming value L for any motion of the camera? It has been shown [20,21] that such points for a particular looming value lie on a sphere. The center of the sphere is located on the instantaneous translational vector and the observer lies on a point of this sphere. A larger translational vector will produce a larger sphere. In our approach objects lying on an Equal Looming Sphere are considered to pose the same threat since they share the same looming value.

2.3 Positive, Negative and Zero Looming Surfaces :

Some of these equal looming surfaces correspond to positive values of looming, some correspond to negative values of looming and there is a plane (i.e., a sphere with infinite diameter) that correspond to zero value of looming. Figure 1 shows the three types of surfaces. The plane that passes through the pinhole point of the camera and is perpendicular to the instantaneous translational vector contains points that produce zero looming. Points on the hemisphere in front of the moving camera, i.e., in front of the zero looming plane produce positive values of looming and likewise points on the

hemisphere behind the moving camera, i.e., behind the zero looming plane produce negative looming values. Naturally, objects lying on the zero or negative looming surfaces do not carry any threat of collision.

2.4 Measurement of Looming Using Optical Flow :

Using the well-known optical flow constraint equation [22], an expression of looming has been derived for a general six-degree-of-freedom motion of a camera [21].

Assume a stationary environment, with a spherical coordinate system $(R\theta\phi)$ attached to a moving camera (Figure 2). Let the instantaneous angular velocity vector ω be represented as $(A, B, C)^T$ where T denotes transpose. Let $s_\theta = \sin\theta$, $c_\theta = \cos\theta$, $s_\phi = \sin\phi$, $c_\phi = \cos\phi$ and I_θ , I_ϕ and I_t denote partial derivative of the brightness I with respect to θ , ϕ and time respectively. Then the expression for looming can be written as :

$$L = \frac{(c_\phi c_\theta c_\phi c_\theta + c_\phi s_\theta c_\phi s_\theta + s_\phi s_\phi)(-c_\phi I_t - c_\phi (Bc_\theta - As_\theta)I_\phi + (Cc_\phi - s_\phi (Bs_\theta + Ac_\theta))I_\theta)}{(c_\phi c_\theta s_\theta - c_\phi s_\theta c_\theta)I_\theta + c_\phi (c_\phi c_\theta s_\theta c_\theta + c_\phi s_\theta s_\phi s_\theta - s_\phi c_\phi)I_\phi}$$

where the translation vector t forms $\theta = \theta_t$ and $\phi = \phi_t$ angles in the $R-\theta-\phi$ coordinates.

Hence, given the location of the pixel in the image, the instantaneous rotational direction of motion, the direction of the instantaneous translational vector and the spatial and temporal intensity changes, the looming value of the corresponding point in 3D can be obtained.

2.5 Measurement of Looming from Relative Rate of Expansion :

The relative rate of expansion of a small object in the image is proportional to its looming value. This concept, discussed in detail in [21], shows that the looming can be calculated from the projected area of a 3D object as :

$$L = - \frac{\frac{dR}{dt}}{R} \approx \frac{\frac{dA}{dt}}{2A}$$

where A is the projected area of a 3D ball on a spherical image and R is the range of the ball from the camera. This idea of relative change of the projected area is used in our approach in obstacle avoidance.

3. VISUAL FIXATION AND LOOMING - AN OVERVIEW

3.1 Fixation :

Visual fixation is actively manipulating the imaging system by directing the attention to some specific points. These points in 3D space carry an immediate relationship with the task being performed. The fixation point may be stationary or in motion with the imaging system. By changing fixation or the point of attention an autonomous system can reduce a vast 3D space into a small working domain and therefore by reducing the computational complexity it can simplify the visual tasks. Fixation obviates a very high resolution wide field of view imaging system. Moreover, fixation and a logarithmic retina simplifies the calculation of looming [20,21]. The motivation comes from the well known process in which alternate saccadic eye movement and fixation are witnessed in human obstacle avoidance [23].

3.2 Looming and Changing Fixation as a Function of Time :

It has been already mentioned that equal looming surfaces are spheres. The size of each sphere is proportional to the instantaneous translational velocity. As a camera approaches a point, the equal looming spheres containing this point shrink, producing higher values of looming. The centers of the spheres lie on a line along the instantaneous velocity vector (not necessarily on the optical axis) and the observer lies on a point on the sphere surface. Also it has been shown that the spheres are independent of the instantaneous rotational parameters [20,21]. Now, Figure 3a illustrates the situation where a camera moves through multiple stationary objects, fixating at them and calculating each obstacle's looming value.

The camera, while making the translational motion rotates and fixates at all the four objects A, B, C and D. The corresponding looming values are calculated. At time instant t_1 the looming values for all the four objects are positive and the looming value of B gets its maximum. Similarly, at t_2 the looming value for the obstacle C gets the maximum while the looming values for A and D increase and that for B decreases. The looming values are plotted in Figure 3b. Note that the looming values of points A, B and C at t_3 are the same since they lie on an equal looming sphere. Also note that at this instant of time the derivative dL/dt indicates each curve's rate of change which signifies how close an object is with respect to the translational vector. dL/dt is the maximum for D which is directly on the path of the camera. The graph also shows zero and negative looming values at some time instants.

4. THE OBSTACLE AVOIDANCE ALGORITHM

4.1 The Task :

The task to be performed is the following :

A camera moving in a constant translational velocity has to reach a visible goal in an environment filled with several obstacles. The camera reaches the goal when the looming value of the object exceeds a certain value. The camera is assumed to be a point.

4.2 The Concept :

The space around the camera has been divided into three zones depending on two threshold looming values:

- Safety Zone
- Mild Risk Zone
- High Risk Zone

This is illustrated in Figure 4. Note that these zones are moving with the camera and their physical size may expand or shrink depending on the speed of the camera. While moving, the camera pans and fixates at each visible obstacle. The looming value is calculated for each obstacle using the method of relative rate of change of area of the projected image (section 2.5).

The idea used is that varying values of looming signify varying degrees of threat of collision, higher looming values indicating higher degrees. The looming value of each obstacle is compared with the two threshold values and a risk assignment is performed. Refer to Figure 5. Obstacle A carries a high risk of collision. Similarly, obstacle B and C pose mild and low risk respectively.

The camera needs to alter its course to avert possible collision with an obstacle only if the obstacle lies in the High Risk Zone. While implementing this alteration, the course is refined taking into consideration the obstacles lying in the mild threat zone too. After changing the course, the camera again looks at all the visible obstacles lying in the $\pm 90^\circ$ from the instantaneous translational vector and decides the degrees of threat coming from the obstacles. If necessary, it changes its course again. If two objects enter the High Risk Zone at the same time the algorithm takes into account the derivative of the looming value. Figures 6,7 and 8 illustrate these sequences. The camera advances towards the obstacle A until the obstacle enters the High Risk Zone. After changing the course to avoid collision it pans and finds a new path through B and C. This path is safe. Hence it continues in this direction. The algorithm uses the following notations :

L_{high} = High threshold value of the looming.

For any obstacle if the looming value exceeds this value, then the obstacle lies inside the High Risk Zone and the camera should alter its course to avoid any collision.

$(L_{mild} < L_{high})$ = Mild threshold value of the looming

If the looming value for any obstacle is in the range of L_{high} and L_{mild} , then that obstacle is in the

Mild Risk Zone. It is not needed to alter the course of the camera, but if it is altering anyway because of some high risk obstacle, it should refine its new course so that this obstacle posing mild threat does not become a high risk threat after the camera changes its course.

α = safe passing angle.

This is a clearance angle that the camera should maintain whenever it passes through two obstacles that it is trying to avoid.

4.3 The Basic Algorithm :

step 1. Initial Path : A straight line that connects the start and the destination point is the **original path**. The line along the translational motion vector is the **current path**. Initially **current path** and the **original path** are the same. Start moving along the **original path**.

step 2. Fixation : Rotate the camera through $\pm 90^\circ$ (relative to the current path) fixating at each visible object.

step 3. Calculation of Looming : Using the 'relative rate of expansion' method calculate the looming L and dL/dt of each obstacle.

step 4. Risk Assignment and Course Deviation Decision : Sort the looming values in descending order and compare the values with L_{high} and L_{mid} . If the highest looming value is in the High Risk Zone then alter the course. That obstacle represents the maximum threat. If there are multiple obstacles with the same looming value, then they lie on the equal looming sphere. Look at their time derivative dL/dt . We consider only positive dL/dt since in this case the looming value will increase as opposed to negative value of dL/dt where the looming value decreases. The obstacle with the highest dL/dt lies closest to the **current path** and poses the maximum threat. If the highest value of the looming is in the range $L_{mid} \leq L < L_{high}$ then it is not necessary to change the course. If the highest value is $L < L_{mid}$ then the path is naturally safe.

Hence, if $L < L_{high}$ then go to *step 7*

else

continue.

step 5. Changing of Heading Vector : Altering the course is necessary to avoid any collision with the obstacles inside the High Risk Zone. While altering the course ignore the obstacles lying in the safe zone.

Rotate the camera and fixate at the obstacle posing maximum threat (i.e. maximum looming). Measure this fixating angle from the **current path**. Also measure the fixating angles for other obstacles in the high and mild risk zones. Choose a path that is closest to the **current path** and maintains the safe passing angle α .

step 6. Translation : Move in this altered path for a unit time. Then connect the end of the translational motion vector with the destination by a straight line. This becomes the **current path**.

step 7. Checking the Destination : If the tip of the translation vector is not within a specified looming value with respect to the destination point, then go to *step 2* else the destination has been reached, so stop.

Note : Stay very close to the current path to avoid excessive change in the steering angle.

5. IMPLEMENTATION

5.1 The flight simulator :

The implementation has been done on a miniature, six-degree-of-freedom vision-based flight simulator. The simulator allows autonomous vision-based navigation in a miniature environment. This simulator has been obtained by modifying an IBM Clean Room Gantry Robot (Figure 9). The modification is such that the robot's main controller is not being used, instead it is controlled directly by an 80486 based personal computer that supplies analog signals to the six control loops each corresponding to a *velocity* input. This new control configuration of the robot allows smooth motion of the different axes.

The visual input to the computer is obtained from a miniature camera located at the end of the end-effector of the robot. Visual data from this miniature camera is processed in the PC-based vision processor. As a result, pitch, yaw, roll and speed signals are generated and sent to control the robot.

5.2 Experimental Setup :

There are several white plastic balls placed in a dark black background as shown in Figure 10. Currently the camera uses only the roll and thus pans only on the horizontal x-z plane. The fixating angle can vary from 90° positive to 90° degrees negative. The simulator looks at the sequence of images captured by the camera, continuously changes the fixating angle of it and moves through the white balls by avoiding collisions.

6. RESULTS

A threshold has been used to convert the sequence of captured gray level images into binary levels. This simplified the task of identifying the white obstacles. Figures 11a, 12a and 13a show the position of the simulator at three time instants. The robot is clearly advancing by avoiding the obstacles. Figures 11b, 12b and 13b depict the obstacles as seen by the camera while the robot was altering its course. As the robot moves towards an obstacle the change of projected area increases prompting it to alter the course when the obstacle enters into the High Risk Zone.

7. CONCLUSIONS AND FUTURE WORK

We have argued that visual looming as a cue, together with active fixation of the camera, are useful tools for obstacle avoidance. The results substantiate that argument. Our next phase of work will cover the following areas :

- calculation of looming in more unstructured scene
- considering the finite size of the camera
- incorporating speed change capabilities of the camera
- using optical flow

In the actual implementation of our idea we have maintained a somewhat structured environment with white balls in dark background, for ease of locating the obstacles and measuring the looming values. In real world scene naturally the world will not be full of white 3-D balls. However, most of the real world objects and the backgrounds have textures which can be used to calculate the looming values. In that case, separating obstacles from the background and calculating the looming for each of these obstacles become two very crucial and challenging problems.

REFERENCES

- [1] Aloimonos, J., Weiss, I. and Bandyopadhyay A., "Active Vision". *Intl. Journal of Computer Vision*, 1, 4, pp. 333-356, 1988.
- [2] Aloimonos, J. "Purposive and Qualitative Active Vision", *Proc. DARPA Image Understanding Workshop*, pp. 816-828, September 1990.
- [3] Ballard, D.H., "Animate Vision", *Artificial Intelligence*, 48, pp. 57-86, 1991.
- [4] Raviv, D. and Herman M., "Visual Servoing for Robot Vehicles using Relevant 2-D Image Cues", *A book chapter to be published*.
- [5] Gilbert, E.G. and Johnson, D.W., "Distance Functions and Their Applications to Robot Path Planning in the Presence of Obstacles", *IEEE J. Robotics Automat.*, vol. RA-1, No. 1, pp. 21-30, March 1985.
- [6] Shin, K.G. and McKay, N.D., "Minimum-time Control of Robotic Manipulators with Geometric Path Constraints", *IEEE Trans. Automat. Contr.*, vol. AC-30, No. 6, pp. 531-541, June 1985.
- [7] Kim, B.K., and Shin, K.G., "Minimum-time Path Planning for Robot Arms and their Dynamics", *IEEE Trans. Syst. Man Cybern.*, vol SMC-15, No. 2, pp. 213-223, March/April 1985.
- [8] Khatib, O., "Real-time Obstacle Avoidance for Manipulators and Mobile Robots", *The Int., J. Robotics Res.*, vol. 5, No. 1, 1986.
- [9] Hogan, N., "Impedance Control : An Approach to Manipulation", *J. Dyn. Syst. Measurement Contr.* vol. 107, pp.1-24, March 1985.
- [10] Krogh, B.H., "Feedback Obstacle Avoidance Control", *21st Allerton Conf. Commun. Contr. Computing*, Urbana, IL, pp. 325-334, October 1983.
- [11] -, "A Generalized Potential Field Approach to Obstacle Avoidance Control", *Proc. Robotics Int. Robotics Res. Conf.*, Bethlehem, PA, August 1984.
- [12] Newman, W.S. and Hogan N., "High Speed Robot Control and Obstacle Avoidance using Dynamic Potential Functions", *Proc. Robotics Automat.*, IEEE, Raleigh. NC, pp. 14-22, 1987.
- [13] Volpe, R. and Khosla, P., "Artificial Potentials with Elliptical Isopotential Contours for Obstacle Avoidance", *Proc. 26th Conf. Decision Contr.*, IEEE. Los Angeles, CA, December 1987.
- [14] Nelson, R.C., and Aloimonos, J., "Obstacle Avoidance using Flow Field Divergence", *IEEE Trans. on PAMI*, 11, No. 10, October 1989.
- [15] Alderson, G.J.K., Sully, D.L. and Sully, H.G., "An Operational Analysis of a One Handed Catching Task Using High Speed Photography", *J. Motor Behav.* 6, pp. 217-226, 1974.
- [16] Beek, P.J., "Perception-action Coupling in the Young Infant : An Appraisal of Von Hofsten's Research Program", *In M.G. Wade and H.T.A. Whiting (Eds.), Motor Development in Children : Aspects of Coordination and Control*, Dordrecht, The Netherlands: Martinus-Nijhoff, pp. 187-196, 1986.
- [17] Bootsma, R.J., *The Timing of Rapid Interceptive Actions*, Amsterdam: Free University Press, 1988.
- [18] Lee, D.N., "A Theory of Visual Control of Braking Based on Information about Time-to-collision", *Perception*, 5, pp. 437-459, 1976.
- [19] Gibson, J.J., *The Ecological Approach to Visual Perception*, Cornell University Press, Ithaca, N.Y., 1966.
- [20] Raviv, D., "A Quantitative Approach to Looming", National Institute of Standards and Technology Tech. Report, NISTIR-4808, April 1992.
- [21] Raviv, D., "Visual looming", *Proc., SPIE Conf. on Intel. Robots and Comp. Vision XI : Algorithms, Techniques and Active Vision*, Boston, MA, November, 1992.
- [22] Horn, B.K.P. and Schunck B.G., "Determining Optical Flow", *Artificial Intelligence*, 17, pp. 185-204, 1981.
- [23] Swain, M.J. and Stricker, M., (Eds.) *Promising Directions in Active Vision*, NSF Active Vision Workshop, August 1991.

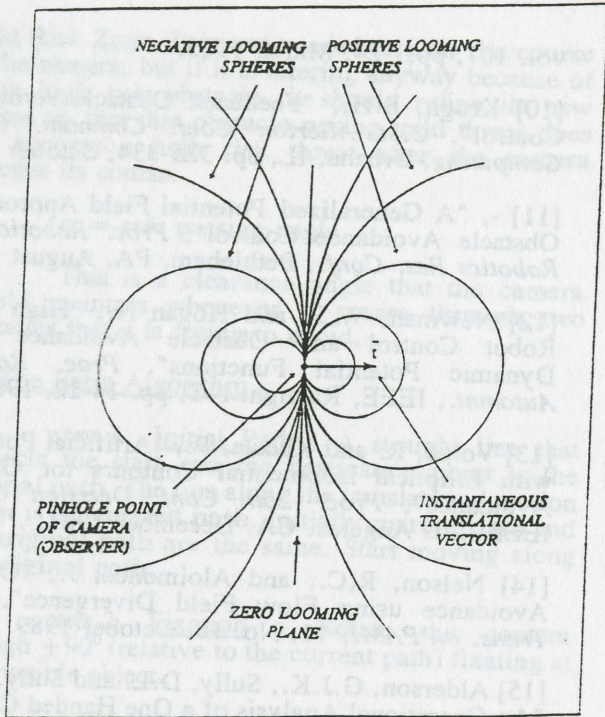


Figure 1: Positive and Negative Looming Spheres and Zero Looming Plane

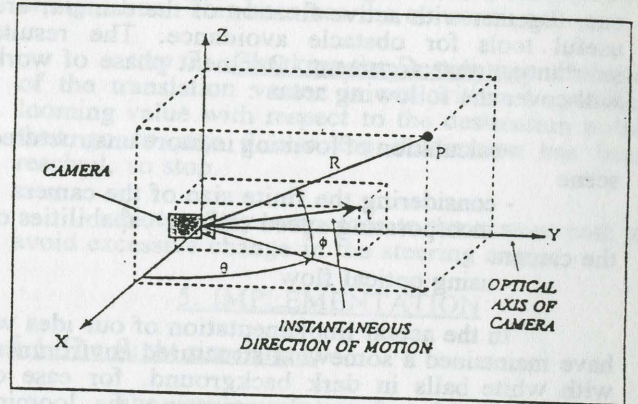


Figure 2: Camera Coordinate System

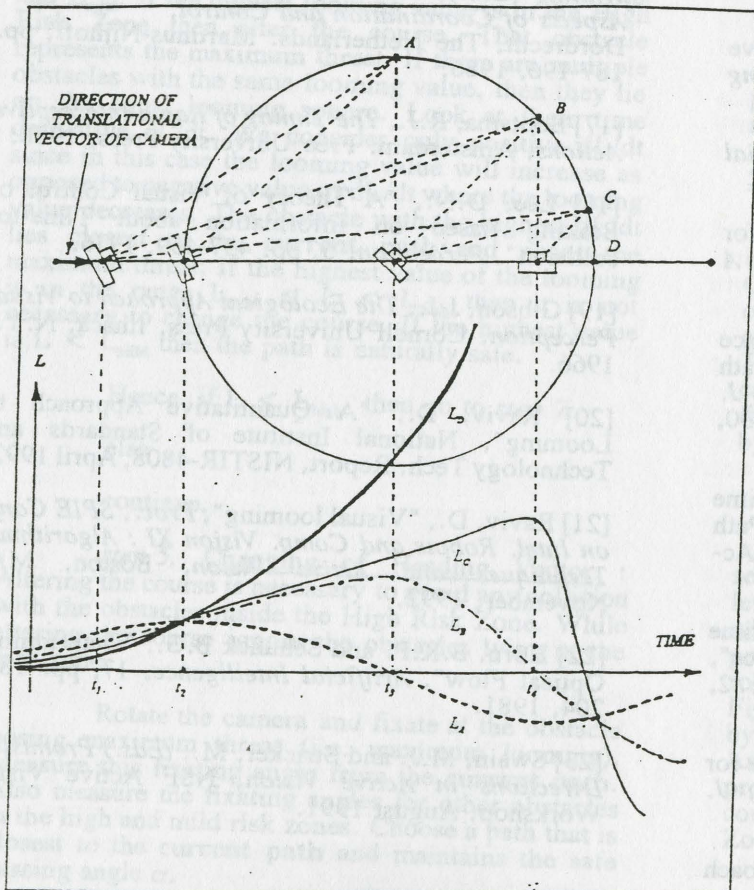


Figure 3: Looming and Fixation as a Function of Time

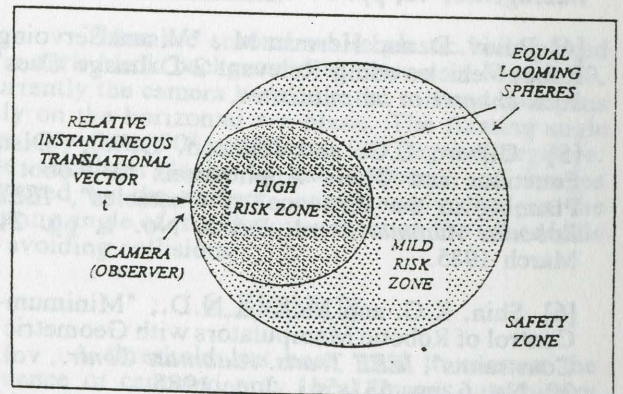


Figure 4: Different Looming Zones

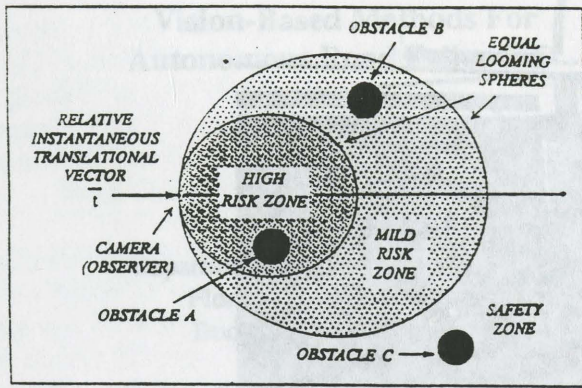


Figure 5: Obstacles in Different Looming Zones

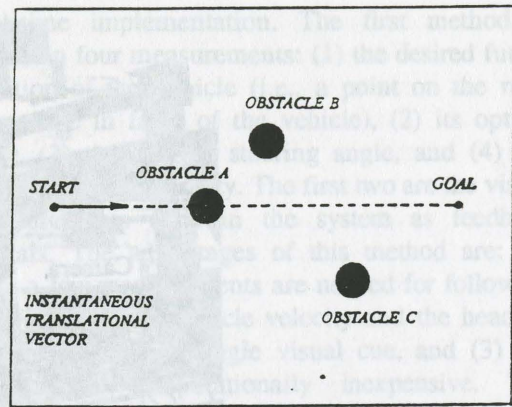


Figure 6: Obstacle A to be Avoided

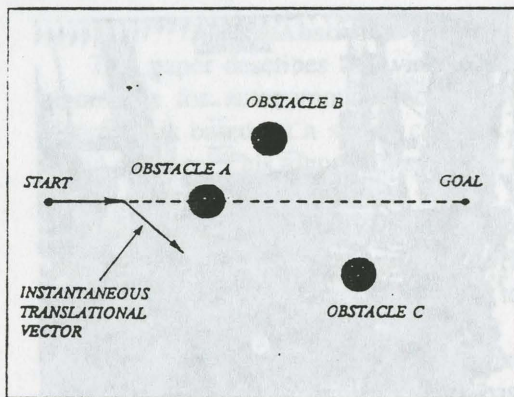


Figure 7: Changed Translational Vector

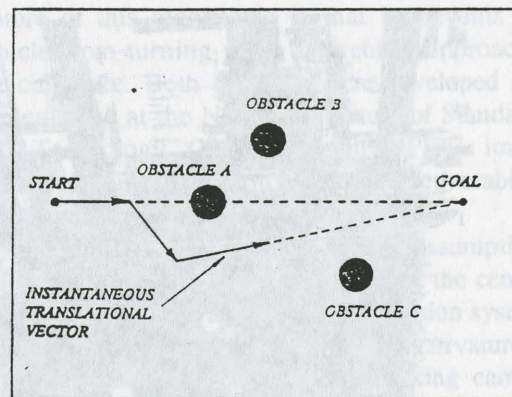


Figure 8: Changed Heading Vector

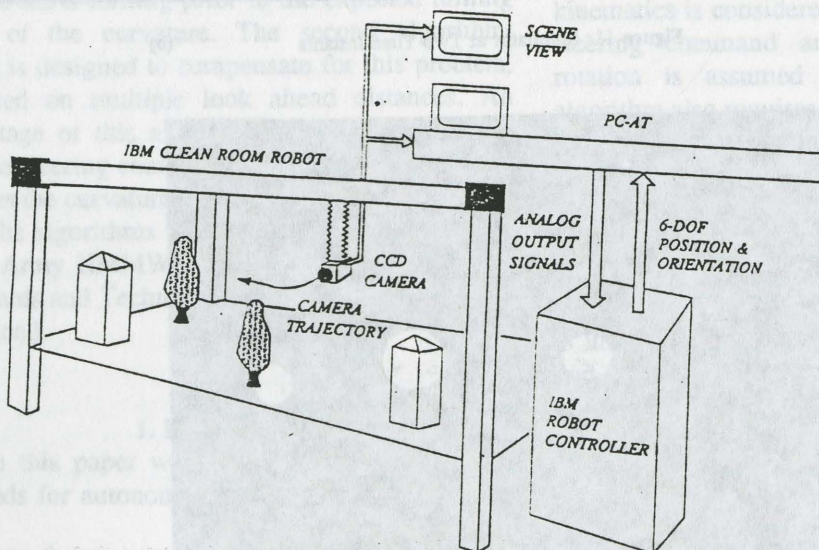


Figure 9: Flight Simulator

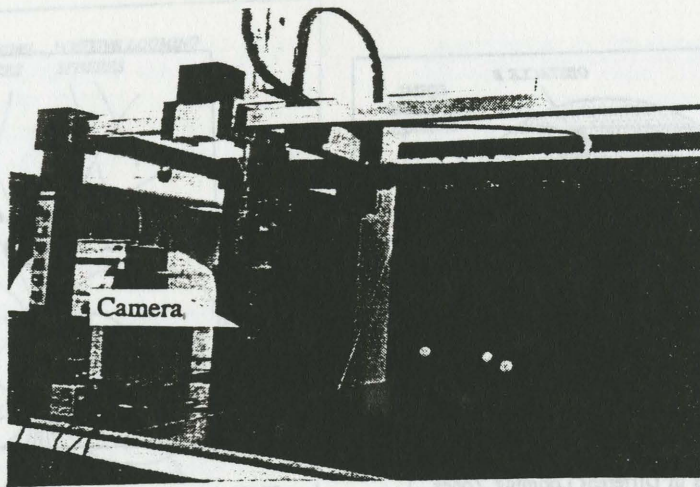
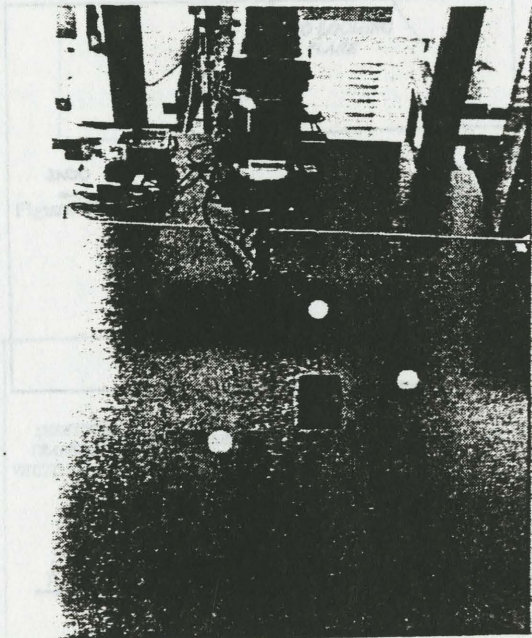
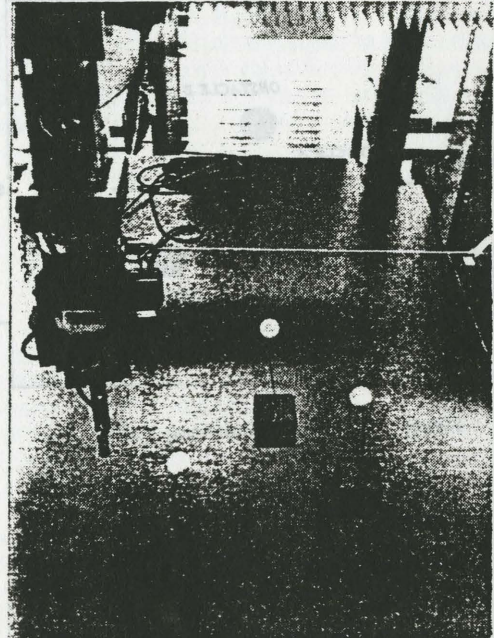


Figure 10: Experimental Setup



(a)



(b)

Figure 11: Simulator at Two Time Instants

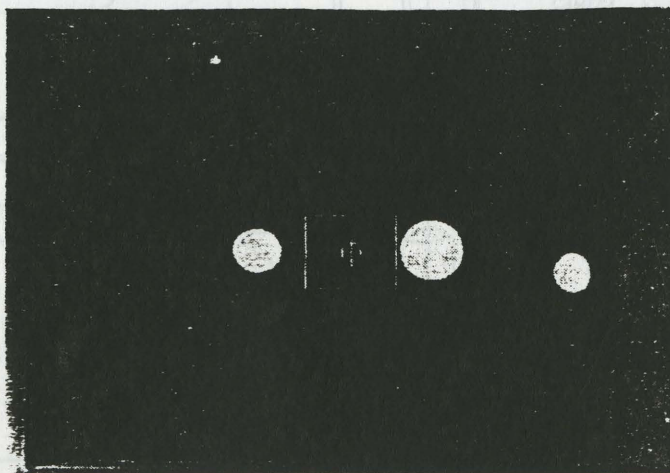


Figure 12: An Image Taken from the Moving Camera

Vision-Based Methods For Autonomous Road Following*

Huseyin Hakan Yakali
Daniel Raviv

Robotics Center and
Department of Electrical Engineering
Florida Atlantic University,
Boca Raton, Florida 33431
and
Robot Systems Division
National Institute of Standards and Technology
(NIST) Bldg 220, Rm B124
Gaithersburg, MD 20899

Abstract

This paper describes two vision-based control algorithms for autonomous road following. One algorithm is based on a single constant-time look ahead distance. This algorithm selects a goal point on the road centerline and the vehicle speed and the radius of rotation are corrected in such a way that the vehicle will be at the point after T seconds. Since the look-ahead distance is chosen to be time-based, the selection of the goal point is speed-dependent. Steering and velocity commands are generated in order to reach this point based on the maximum lateral acceleration allowed and the location of the point relative to the vehicle. This method is based on the assumption that there are no abrupt changes in the curvature of the road centerline. However, this assumption may not be realistic, e.g., when the curvature changes from a straight road to a curved one. As a result, the vehicle starts turning prior to the expected turning point of the curvature. The second algorithm, which is designed to compensate for this problem, is based on multiple look ahead distances. An advantage of this algorithm is that it delays the vehicle steering command until the vehicle actually reaches the curvature.

The algorithms were partially implemented on a US Army HMMWV at the National Institute of Standards and Technology (NIST) in Gaithersburg, Maryland.

1. Introduction

In this paper we report on new vision-based methods for autonomous road following and their

real-time implementation. The first method is based on four measurements: (1) the desired future location of the vehicle (i.e., a point on the road centerline in front of the vehicle), (2) its optical flow, (3) the current steering angle, and (4) the current vehicle velocity. The first two are the visual measurements used in the system as feedback signals. The advantages of this method are: (1) only a few measurements are needed for following the road, (2) the vehicle velocity and the heading are adjusted by a single visual cue, and (3) the method is computationally inexpensive. The second method is based on two image measurements which are equivalent to the locations of the two points on the road centerline at two different look ahead distances. The unique feature of this algorithms is that it prevents the vehicle from turning until the vehicle approaches the curvature. Both methods were developed and implemented at the National Institute of Standards and Technology (NIST) using a PIPE image processor and a US Army High Maneuverability Mobility Wheel Vehicle (HMMWV).

The algorithms are based on the assumptions that the road is part of a planar surface, the centerline of the road is available from the vision system, and there are no abrupt changes in the curvature of the road centerline. The forward looking camera parameters and its position/orientation on the vehicle are assumed to be known. As a result there is a known one-to-one mapping between the image plane and the road surface. Thus the methods can be described in either 2D image plane or 2D road coordinates. Both algorithms require the knowledge of the vehicle velocity and assume relatively low speed of the vehicle i.e., only the kinematics is considered. The relation between the steering command and the vehicle radius of rotation is assumed to be known. The first algorithm also requires the knowledge of the radius of rotation of the vehicle.

Several different methods for autonomous road following have been developed in the past. Dickmanns and Graefe [1] used monocular vision in their system. A window is opened for every important road feature in the image. Then, a single filter is utilized to track these features in a 4-D (space and time) world representation. The interpretation of the images is done in a 4-D world model instead of in the image coordinate system. The Martin-Marietta Group [2] has achieved results for selecting the road/non-road regions by using

*This work was supported in part by a grant to Florida Atlantic University from the National Science Foundation, Division of Information, Robotics and Intelligent Systems, Grant #IRI-9115939.

colored images. The Navlab, at Carnegie-Mellon University [3], used color images for detection of road/non-road regions. Once the road edges are detected, the vanishing point is used to generate steering commands. A range scanner is utilized for obstacle detection and terrain analysis. Work in road following has also been done at the University of Maryland [4] and at Ford Motor Company [5]. In most of the systems mentioned above, a 3-D representation of the world is recovered in order to generate steering commands. Raviv and Herman [6] suggested simpler methods for road following.

2. The First Method

2.1. The Approach

After selecting a goal point on the road centerline, the vehicle speed and the radius of rotation are corrected in such a way that the vehicle will be at the point after T seconds. Since a look-ahead distance is chosen to be time-based, the selection of the goal point is speed dependent. Velocity and steering commands are generated in order to reach the look-ahead point based on the maximum lateral acceleration allowed and the location of the point relative to the vehicle, respectively.

2.2. The Effect of Speed, Radius of Rotation and Lateral Acceleration

Figure 1 is a top view of the planar surface on which the vehicle is moving. Point A is the current location of the vehicle and the y-axis is the longitudinal axis of the vehicle. The arc ABC is the future path of the vehicle for the current speed and the radius of rotation. Point B is a future location of the vehicle after T seconds. Clearly the location of point B is dependent on both radius of rotation and the speed of the vehicle. Under the current conditions, the arc AB traveled by the vehicle can be computed by

$$l = V \cdot T = 2 \cdot \alpha \cdot R \quad (1)$$

or the angle α is:

$$\alpha = \frac{V \cdot T}{2 \cdot R} \quad (2)$$

where V is the vehicle speed, R is the radius of rotation and α is the angle between y-axis and the AB straight line.

2.2.1. Constant Velocity Curves

Figure 2 is a top view of all possible locations

of point B (as described in Figure 1) for a constant velocity and different radii of rotation. The vehicle location is at (0,0). Figure 3 shows a family of constant velocity curves each of which corresponds to a value of velocity. The closer the curve to the location of the vehicle, the lower the speed.

2.2.2. Constant Radius of Rotation Curves

Figure 4 shows a family of constant radius of rotation curves. This means that for any speed and a constant steering command the vehicle will follow a curve as shown in Figure 4. Clearly, these curves are circles.

2.2.3. Constant Lateral Acceleration Curves

In order to maintain a lateral acceleration less than a specific value, the following has to be satisfied:

$$\frac{V^2}{|R|} \leq a_{\max} \quad (3)$$

where a_{\max} is the maximum level of allowed acceleration. Equation (3) can be used for finding the maximum value of allowed V_{\max} for given radius of rotation and maximum acceleration.

$$V_{\max} = \sqrt{a_{\max} \cdot |R|} \quad (4)$$

Figure 5 shows a pair of curves which satisfy Equation (4). These curves are called constant lateral acceleration curves for a_{\max} . Figure 6 shows three different pairs of lateral acceleration curves. The closer the curves to the y-axis, the lower the lateral acceleration and the *safer* the vehicle.

2.2.4. Combining The Above Curves

In Figure 7, the three families of curves, i.e., the constant velocity (V), the constant radius of rotation (R) and constant maximum lateral acceleration curves (a_{\max}), are superimposed. Next we explain how to use these curves for generating driving commands.

2.3. Vehicle Control Using the Acceleration Constraint

The goal is to control both the vehicle steering and speed in such a way that a T-second-look-ahead point will stay on both the centerline of the road and the constant acceleration curves. In reality this may not be achieved due to noise and the dynamics of the vehicle. Therefore we try to keep this point as close as possible to the constant acceleration curve.

Refer to Figure 8. Let R_c be the current radius of rotation and V_c the current vehicle velocity. After T second the vehicle will be at P_4 . Assume that V_c^2/R_c is a_{max} , the maximum acceleration allowed. This means that the vehicle's T -second-look-ahead point is on the border of the region which is between the constant acceleration curves and is not allowed to be outside of this region. Assume now that the goal point is P_3 i.e. we have to generate control commands such that the vehicle will reach P_3 point after T seconds. Note that P_1 , P_2 , P_3 and P_4 points in Figure 8 are in reality very close to each other, since we assume no discontinuities in the curvature of the centerline.

Ideally, to go from P_4 to P_3 , we need to change both steering and speed commands and keep the T -second-look-ahead point as close as possible to P_4 P_3 curve. Speeding up first from P_4 to P_1 and then steering from P_1 to P_3 is not a good solution since P_1 will result acceleration higher than the allowed one. On the other hand changing the steering first, i.e., moving from P_4 to P_2 and then P_2 to P_3 by changing the speed is a better solution.

2.4. Velocity and Heading Adjustments

The above section explained the concept of velocity control based on T -second-look-ahead and constant acceleration. In this section we explain how to quantitatively generate speed commands. Substituting Equation (4) into Equation (2) yields

$$\alpha = \frac{T \cdot a_{th}}{2 \cdot V} \quad (5)$$

or

$$\alpha \cdot V = \frac{T \cdot a_{th}}{2} \quad (6)$$

By keeping T -second-look-ahead and the lateral acceleration constant, one can get the change in V as a result of change in α . Taking the derivative of Equation (7) with respect to time yields,

$$\alpha \cdot dV + V \cdot d\alpha = 0 \quad (7)$$

or

$$dV = -\frac{V}{\alpha} d\alpha \quad (8)$$

or

$$\frac{dV}{dt} = -\frac{V}{\alpha} \frac{d\alpha}{dt} \quad (8.a)$$

Similarly, in order to find the changes in vehicle radius of rotation, the vehicle velocity is solved from Equation (5) and then substituted back in Equation (2). The resulting equation is

$$dR = -\frac{2R}{\alpha} d\alpha \quad (9)$$

or

$$\frac{dR}{dt} = -\frac{2R}{\alpha} \frac{d\alpha}{dt} \quad (9.a)$$

Both the vehicle velocity and the vehicle radius of rotation are measurable from the vehicle, i.e., from the speedometer and the steering wheel of the vehicle, respectively. The only variable that is not measured from the vehicle is the α angle and has already been defined (Figure 1) as the angle between the vehicle axis and the target point measured from the center of the vehicle coordinate system.

Note that the required change in vehicle velocity dV and radius of rotation dR can be calculated by using α , $d\alpha$, V and R . This simplicity is the unique feature of this method.

2.5. Measurement of the α angle

The algorithm defined uses three different variables. Two of these variables, i.e., V and R , are measured from the vehicle. α is the only variable which is measured from the image. Based on the assumption that the road is locally flat, one to one mapping from the image plane to ground plane is possible. As a result one can find 3D world coordinates of an image point and calculate α angle.

2.6. Velocity and Heading Control Block Diagram:

Block diagram of the overall system is shown in Figure 9. Image processor block takes an image using the camera mounted on the vehicle and extracts the road edges from the image. The controller block has four tasks: (1) to calculate road centerline from the road edges, (2) select a new goal point using the road centerline and (3) calculate α angle and $\Delta\alpha$ for the new goal point and (4) calculate ΔV and ΔR by using the vehicle radius of rotation and the speed measurements from the vehicle, α angle and $\Delta\alpha$. α_i and $\alpha_{i-\Delta t}$ are the current and previous measurements of α , respectively.

3. The Second Method

3.1. Problem Statement & The Approach

The previous method is based on the assumption that there are no abrupt changes in the curvature of the road centerline. However, this assump-

tion may not be realistic e.g. when the curvature changes from a straight road to curved one. As a result the vehicle will start turning prior to the expected turning point of the curvature. We call this the *when-to-steer* problem.

The second method deals with when-to-steer problem. The goal is to delay the vehicle steering command until the vehicle actually reaches the curvature. This is done by (1) measuring road radius of curvature from several different distances in the image and (2) averaging these measurements.

3.2. Measuring Circular Road Radius

Refer to Figure 1. x , y , and α are defined in the preceding section. But this time arc ABC is defined to be the circular road centerline and its radius R is unknown.

In order to measure R , the road curve is searched by several (in this case three) horizontal lines, called search lines (Figure 10). The search lines are always parallel to the x -axis. Their distances from the x -axis are denoted by l_1 , l_2 and l_3 . D, E and F points are the intersection points of the search lines with the road curvature. By using y -axis and A, D, E, F points, three angles are defined: α_1 , α_2 , and α_3 . They are the angles of AD, AE, and AF lines relative to the y -axis, respectively. Then the radius of the circular road is calculated for every (l_i, α_i) , $i=1,2,3$, using the following equation.

$$R_i = \frac{l_i}{\sin(2\alpha_i)} \quad (10)$$

Note that as described in the preceding section, α_i angles can be measured from the image plane.

3.3. Calculation of Circular Road Radius

After measuring the road curvature radius for different search lines, the final road curvature is calculated by using weighted averaging.

$$R = \frac{\sum_{j=1}^3 w_j R_j}{\sum_{j=1}^3 w_j} \quad (11)$$

where w_j is the weight of that measurement.

Note that the radius of rotation calculated from the closer search line is more important than the further ones. As a result, the closer search lines get larger weight values. In our application, the distance of the search lines is a function of time

and the vehicle velocity.

$$l_i = V \cdot t_i \quad (12)$$

where V is the vehicle velocity and the t_i is the look ahead time. Using this relation we may choose weights as follows:

$$w_i = \frac{1}{t_i} \quad (13)$$

Substituting this equation back into the averaging equation will result

$$R = \frac{\sum_{j=1}^3 \frac{R_j}{t_j}}{\sum_{j=1}^3 \frac{1}{t_j}} \quad (14)$$

3.4. Implementation

A version of this method was implemented for two points and tested on the HMMWV. Results as can be seen on a videotape are very encouraging.

4. Conclusion

We have shown two vision-based road following control algorithms which are capable of steering a vehicle as well as controlling its velocity. The first algorithm uses a single visual cue and two measurements from the vehicle, velocity and the radius of rotation, to generate steering and velocity commands. The second algorithm is capable of preventing the vehicle from turning until the vehicle is in the curvature. This is achieved by comparing the two visual measurement of the road radius of rotation from two different distances. The results can be seen by the video.

5. References

1. Dickmanns, E.D., and Graefe, V., "Dynamic Monocular Machine Vision and Applications of Dynamic Monocular Machine Vision," Universitat der Bundeswehr, Muchen, July 1988.
2. Turk, M.A., Morgenthaller, D.G., Gremban, K.D., and Marra M., "VITS - A Vision System for Autonomous Land Vehicle Navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.10, No.3, May 1988.
3. Thorpe, C., Hebert, M.H., Kanade, T., and Shafer S.A., "Vision and Navigation for the

Carnegie-Mellon Navlab," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.10, No.3, 1988.

4. Waxman, A., LeMoigne, J., Davis, L., Srinivasan, B., Kushner, B., Liang, E., and Siddalingaiah, "A Visual Navigation System For Autonomous Land Vehicles," *IEEE Journal of Robotics and Automation*, Vol.3, 1987.
5. Kuan, D., Phipps, G., and Hsues, A.

"Autonomous Robotic Vehicle Road Following," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.10, No.5, pp.648-658, September 1988.

6. Raviv, D., and Herman, M., "A New Approach to Vision and Control For Road Following." *National Institute of Standards and Technology Internal Reports NISTIR 4476*, January 1991.

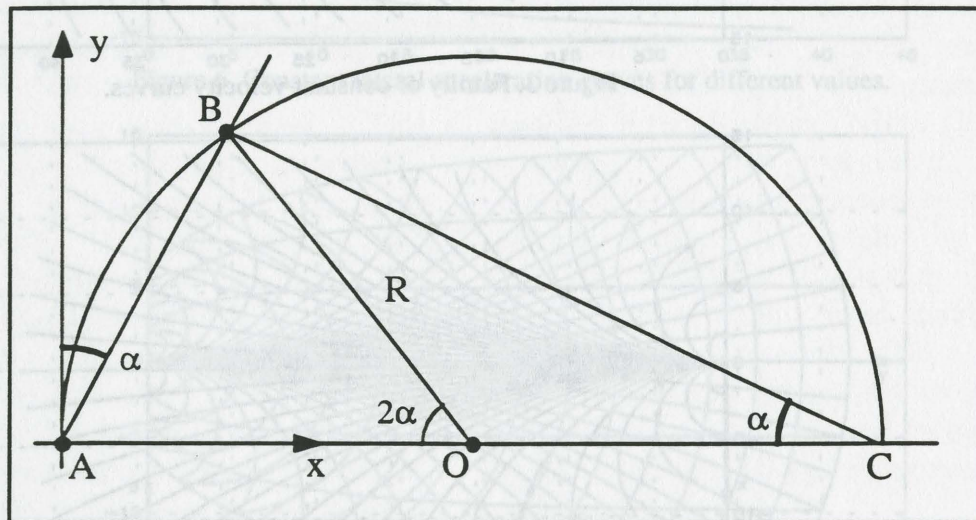


Figure 1. Constant Steering Circle.

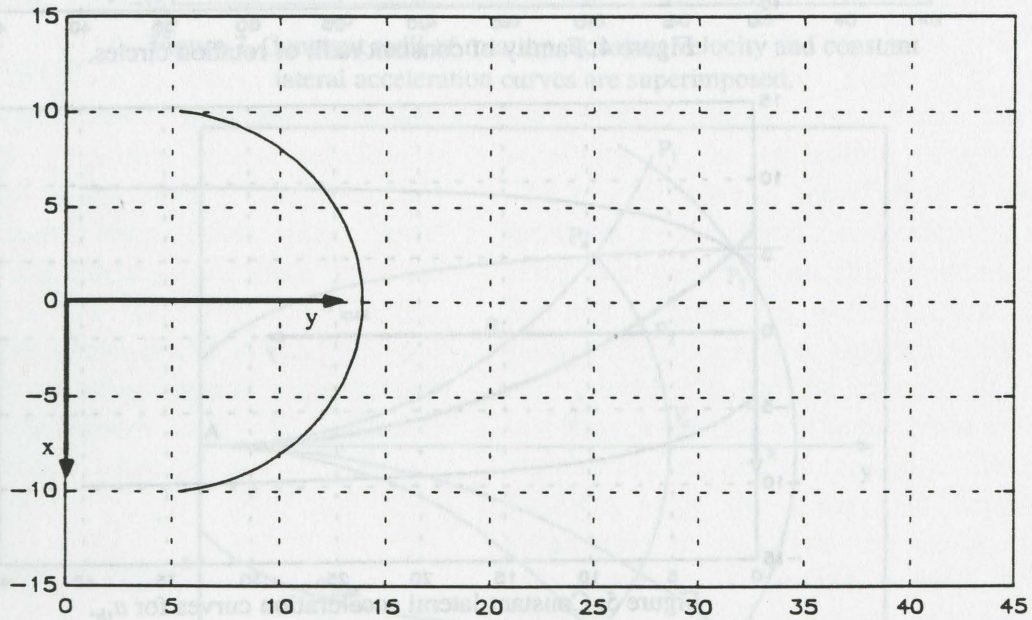


Figure 2. A constant velocity curve.

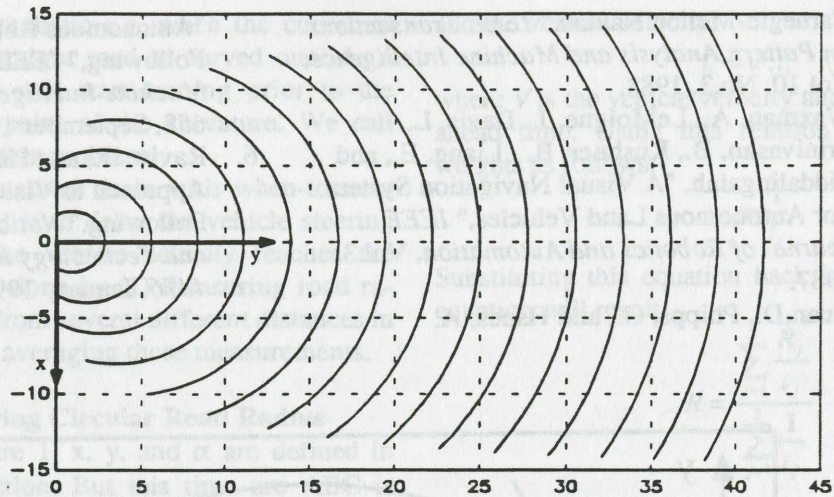


Figure 3. Family of constant velocity curves.

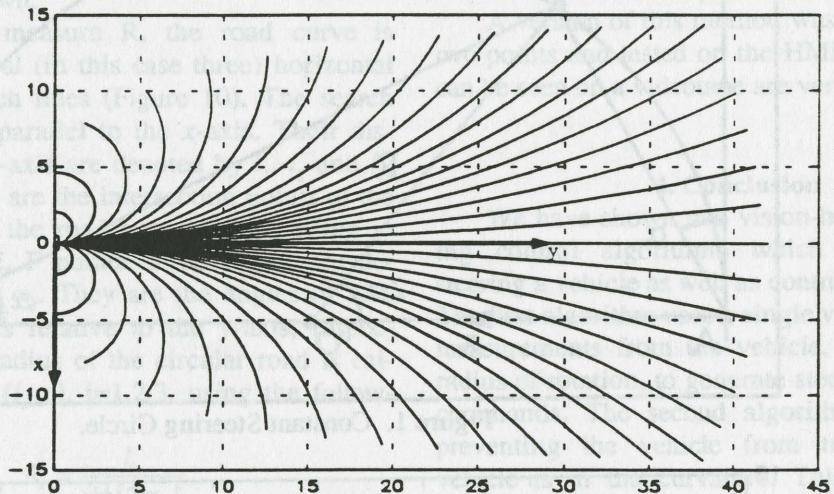


Figure 4. Family of constant radii of rotation circles.

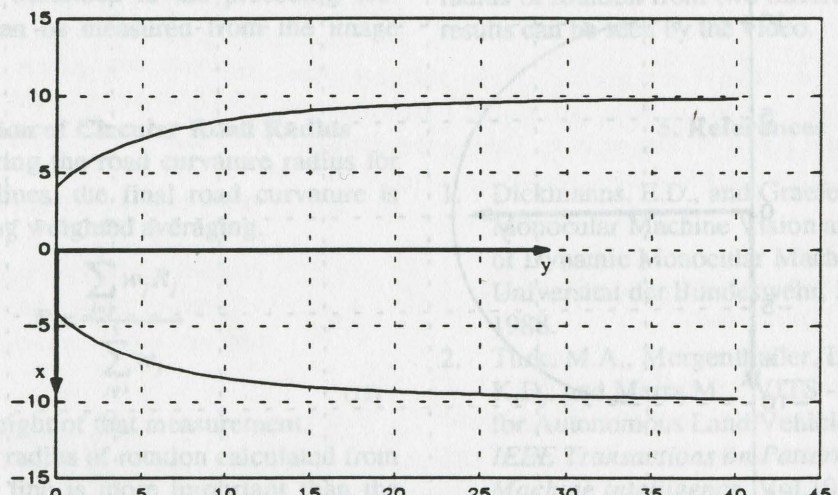


Figure 5. Constant lateral acceleration curves for a_{th} .

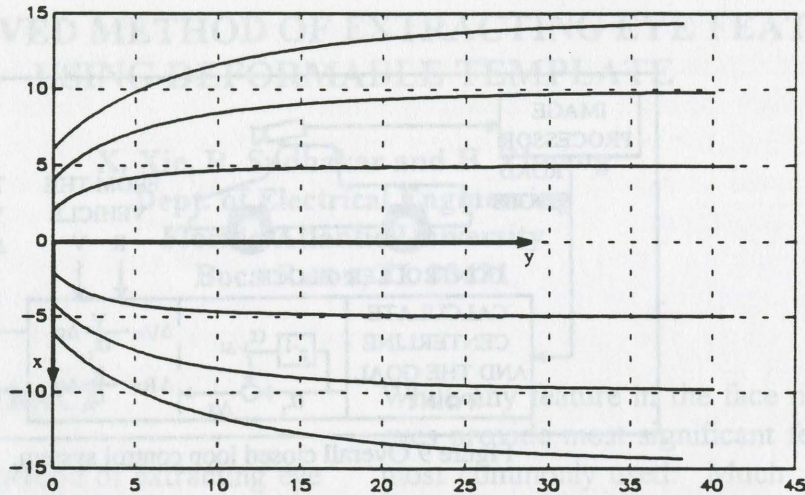


Figure 6. Constant lateral acceleration curves for different values.

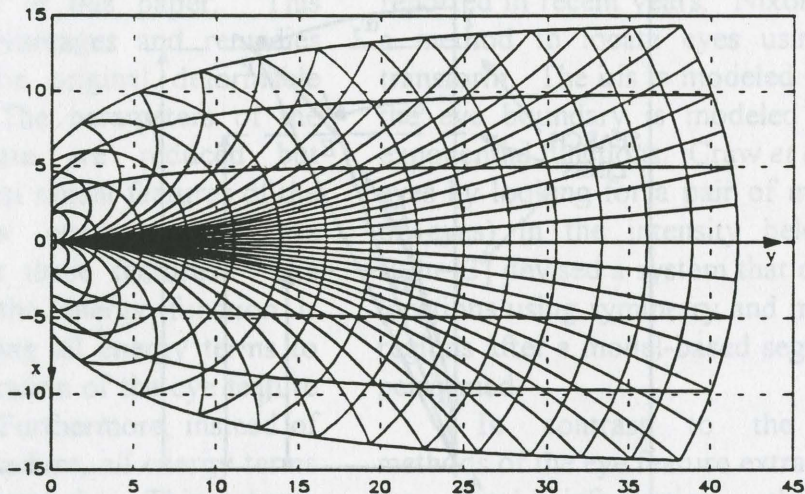


Figure 7. Constant radii of rotation, constant velocity and constant lateral acceleration curves are superimposed.

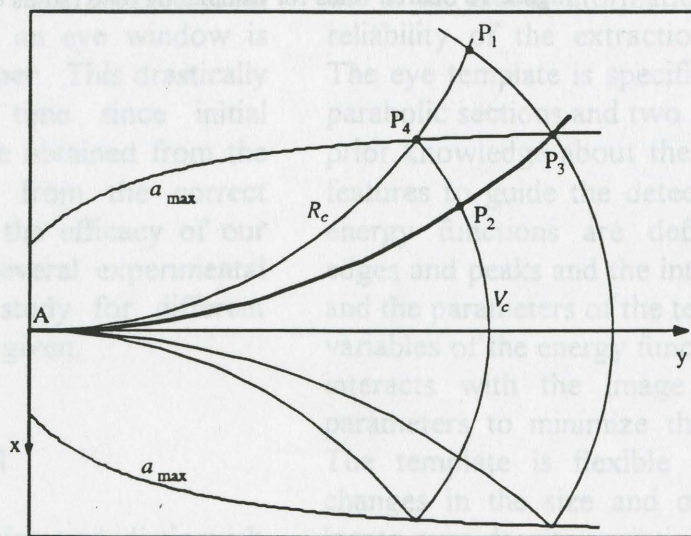


Figure 8. Vehicle control example.

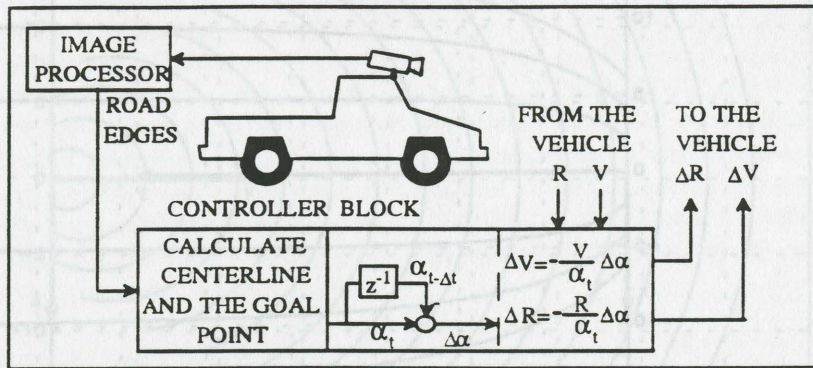


Figure 9 Overall closed loop control system.

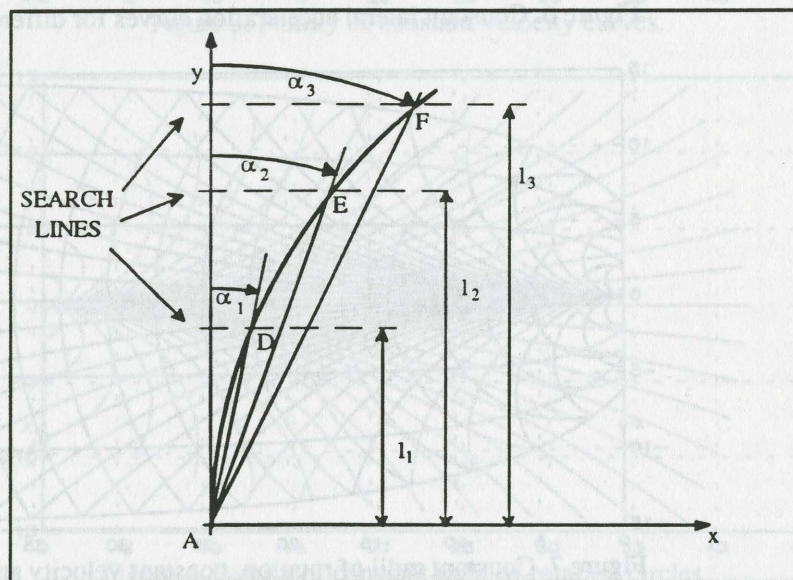


Figure 10 Search lines for calculating road radius of curvature.

AN IMPROVED METHOD OF EXTRACTING EYE FEATURES USING DEFORMABLE TEMPLATE

X. Xie, R. Sudhakar and H. Zhuang
Dept. of Electrical Engineering
Florida Atlantic University
Boca Raton, FL 33431

ABSTRACT

An improved method of extracting eye features from faces using modified eye template is described in this paper. This method retains all advantages and remedies some weakness of the original deformable template approach. The parameters of the modified eye template are reduced but concentrated in the most salient features of the eye and some new energy terms are introduced to capture these features. The weight schedule for the energy function is simplified by normalizing all energy terms to facilitate the implementation of the eye feature extraction algorithm. Furthermore, instead of using a sequential procedure, all energy terms are minimized simultaneously. This scheme helps the eye template to move to the global minimum location. A preprocessing procedure that constructs an eye window is briefly described in this paper. This drastically reduces the processing time since initial parameters of the template obtained from the preprocessing is not far from the correct values. We demonstrate the efficacy of our algorithm by presenting several experimental results. A comparison study for different processing schemes is also given.

1. INTRODUCTION

In order to recognize and distinguish faces, a recognition system must extract salient features from faces of human beings.

While any feature in the face may be located, eyes provide most significant features and are most commonly used. Much research work about the extraction of eye features has been reported in recent years. Nixon [1] described a method to locate eyes using the Hough transform. The iris is modeled by a circle and the eye boundary is modeled by a tailored exponential functions. Craw *et al.* [2] located eyes by looking for a pair of minima (centers of eyes) in the intensity below eyebrows. Buhr [3] devised a system that determined eye locations using symmetry and moments of the regions after a model-based segmentation was performed.

In contrast to the conventional methods of the eye feature extraction based on the local information, the deformable templates proposed by Yuille *et al.* [4] made use of the global information and improves the reliability of the extraction of eye features. The eye template is specified by a circle, two parabolic sections and two points that enable a prior knowledge about the expected shape of features to guide the detection process. The energy functions are defined with valleys, edges and peaks and the intensity of the image and the parameters of the template are also the variables of the energy function. The template interacts with the image by adjusting the parameters to minimize the energy function. The template is flexible enough to handle changes in the size and orientation. It can locate eyes despite variations in scale, tilt and lighting conditions.

Yuille *et al.* also devised a sequential scheme to minimize the energy function. This scheme is very flexible, but there are some problems. First, the weights for energy terms are determined by experiments and are difficult to generalize. Secondly, while the global information is used in the eye template, sequentially changing weights of the energy terms may cause the algorithm sometimes being trapped in local minima. Thirdly, the process time is likely to be high.

We propose some modifications to the deformable template algorithm in order to retain its advantages and avoid above problems. First, rather than using a sequential procedure, all the parameters of the template are changed simultaneously during the minimization process. This prevents some parameters of the eye template from being overly changed and helps the algorithm to converge to the global minimum since all conflicting factors in the energy terms are activated with the update of template parameters. Secondly, in order to overcome the headache issue of selecting weights for the energy terms, we normalize the value of each energy term to the range between 0 and 1 and only two different weights are assigned to the two types of energy terms (energies for the real image and energies for the template structures). Experimental results show that slightly changes of weights have little effect on final results. Thirdly, we eliminate certain parameters and add some energy terms to make the energy function more sensitive to the change of parameters.

The paper is organized as follows. In Section 2, we define the modified deformable template for eyes and formulate its energy function. We describe the preprocessing procedure of finding eye windows and the minimization algorithm in Section 3. In Section 4, we demonstrate the experimental results for the extraction of eye features. The paper ends with concluding remarks.

2. THE ENERGY FUNCTION FOR THE EYE TEMPLATE

For the purpose of an effective eye feature extraction, an eye template should consist of all the relevant eye features. A crucial step in performing feature extraction using the deformable template is to construct a proper energy function. In this section, we give a modified eye template and introduce some additional energy terms to strengthen the energy function.

The eye template proposed by Yuille *et al.* [4] consists of a circle separating the iris and the whites of the eye, two parabolas describing the contours of the upper and lower eyelids and two points located at the center of the whites of the eye. After analyzing parameters of this template and doing some experiments, we found that only using two points in the center of the whites of the eye is not sufficient to describe peak fields in the image and is insensitive to the change of eye parameters (such as eye orientation). The image peaks can be described by a region more effectively. Furthermore, if the eye moves to the extreme positions, one white area will disappear. This eye template can not accommodate such a situation. We modified this eye template by removing two points corresponding to the whites of the eye. The modified eye template is parameterized as shown in Fig. 1.

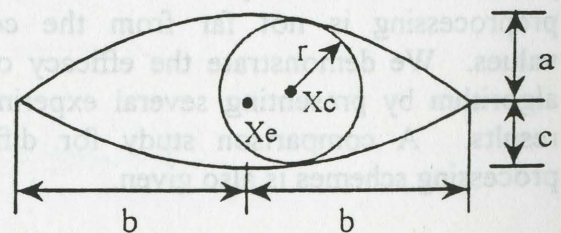


Fig.1 The modified eye template

In this template, 3 parameters (\bar{X}_c, r) correspond to a circle and 6 parameters ($\bar{X}_e, a, b, c, \theta$) correspond to the two

parabolas. The template interacts with the original image and the valley, peak and edge representations of the image.

Two types of the energy terms are defined for this template. One is used to account for the characteristics of the real image; another is used to describe the structures of the template.

The energy function E is the summation of the weighted square of the energy terms:

$$E = \sum_{i=1}^{10} W_i E_i^2$$

The definition of each energy term is given as follows:

(1) The energy term E_v is a measure of the fitness between the eye template and the valley field of the image.

$$E_1 = E_v = -\frac{1}{Area_{circle}} \sum_{i \in circle} b_i$$

where b_i is a binary value of a point in the valley field and summations are made over the circle area.

(2) The edge energy term E_e is used to match the template to the edge of the iris and the upper and lower eyelids.

$$E_2 = E_e = -\frac{1}{Length_{circle}} \sum_i b_i - \frac{1}{Length_{para-bound}} \sum_i b_i$$

where summations are made over the boundaries of the circle and the parabolas.

(3) The new energy term E_p is used to attract peaks in the whites of the eye by making summations over two small windows centered at the left and right whites of the eye, instead of using two points at centers of the whites.

$$E_3 = E_p = -\sum_{i=1}^2 \frac{1}{Area_{window_j}} \sum_{j \in window_j} b_j$$

(4) The new energy term E_c gives a measure for the correct orientation of the eye template.

$$E_4 = E_c = -\sum_{i=1}^2 \frac{1}{Area_{window_j}} \sum_{j \in window_j} b_j$$

where the summations are made over the two small windows, centered at two intersections of two parabolas, corresponding to two eye corners.

(5) The energy term for the intensity inside the circle is defined as

$$E_5 = E_{ic} = -\frac{1}{Area_{circle}} \sum_{i \in circle} g_i$$

where g_i is a gray level value of a pixel inside the circle.

(6) The energy term for the intensity over the whites of the eye is defined as

$$E_6 = E_{iw} = \frac{1}{Area_{whites}} \sum_{i \in whites} g_i$$

where g_i is a gray level value of a pixel in the whites of the eye.

(7) The new energy term E_{ie} also has contributions for attracting the correct orientation of the template using the gray level information.

$$E_7 = E_{ie} = \frac{1}{Length_{para-bound}} \sum_{i \in parabolae} g_i$$

where summations are made for the intensity of image pixels along the two parabolas.

The other three energy terms are described for the structures of the eye template that keep some parameters of the template having suitable links. The last structure energy term is newly added.

(8)

$$E_8 = E_{str1} = \bar{X}_e - \bar{X}_c$$

(9)

$$E_9 = E_{str2} = b - 2r$$

(10)

$$E_{10} = E_{str3} = b - 4c$$

In the above definition for the energy function, some new energy terms have been added to fit the template to eye features more effectively. Usually, the information for two parabolas is less than that for the circle due to the low gray level contrast around the lower eyelid. The task of attracting correct shapes of parabolas is more difficult than that for the circle. Among the parameters of the parabolas, the orientation θ plays an important role because the change of θ will also lead to the change of width, b , and heights, a and c , of the parabolas. In these energy terms, we use three new energy terms, E_c , E_p , and E_{je} , which interact with the valley, peak fields and gray level image, to help orient the template correctly. Orienting the template by multi-terms is more robust than that by single term. For example, using the corner energy term only is insensitive to the change of orientation when there are shade around two eye corners. The peak energy term has similar weakness. But the combination effect of above two energy terms plus the energy term E_{je} , which is used to describe the two parabolas in the gray level image, will force the template to orient correctly.

Several conflicting factors are included in energy terms in order to avoid some parameters to be overly changed during minimization. The intensity and valley terms over the circle have the tendency to shrink to the darkest part of the iris. This effect can be countered by the edge term that pull the circle to the contour of the iris. Similar example can be explained for the intensity of the whites and corner energy terms. Also, all the three structure energies have the counter effect on the image energy terms. The structure energy term E_{str2} , linked width of the parabola, b , with the radius of the circle, r , plays a significant role in countering the effect of shrinking the circle. The new third structure

energy adds a constraint on the width and height of the lower parabola due to the fact that the shape of the lower eyelid is usually invariant.

In Yuille's scheme, the selection of weights is determined by experiments and is hard to generalize its rule. In order to ease the selection of weights for energy terms, uniform weights are highly recommended. In our scheme, two different weights are selected for two groups of the energy terms, one for image energies and another for structures energies. Before assigning weights to energy terms, the value of each energy term has to be comparable and needs to be normalized to the range between 0 and 1. We make the ratio of the weight for image energies to the weight for structure energies be 10:1 to 6:1.

3. MINIMIZATION OF THE ENERGY FUNCTION

We defined energy terms in the last section based on the modified eye template. In this section, we concentrate on the issue of minimizing the energy function. First, we give a brief description about the preprocessing of finding a window around the eye. We then define the problem of finding an optimal match for the eye template as a nonlinear optimization problem.

3.1 Preprocessing

In order to provide an initial position of the eye template, the preprocessing was done to find the locations of eye windows [5]. Because an accurate location of the eye window is not necessary, we work on a binarized image (after thresholding). The assumption is made that the two eye regions (the eyebrow may be included) are isolated from the other parts of the face in the binarized image. To find an eye window, a search strategy is implemented to look for some transition points around the eye region in the binarized image. Making use of the symmetry

of the face geometry, the starting point of search is set near the center of two eyes. Based on the prior knowledge about the geometry of the face, we first search two points in the eye regions from the starting point. The search is then proceeded in the horizontal direction, to left and right, followed by the vertical search, up and down. This procedure is continued alternatively in the horizontal and vertical directions until two successive searches in both directions find no transitions. Because the search is implemented line by line, this algorithm is unaffected by small irregular gaps in the binary image. The eye window is determined by the four finally searched transitions (up and down, left and right) shown in Fig.2.

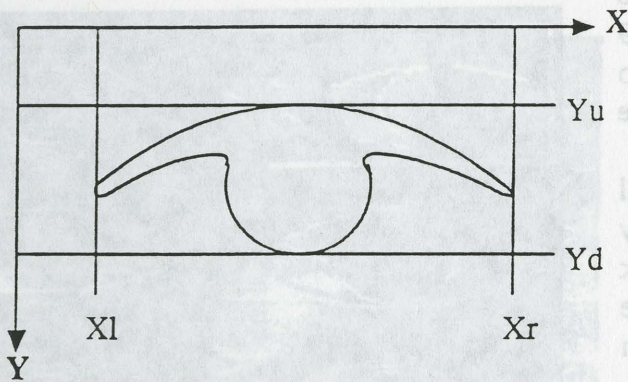


Fig. 2 Located eye window

3.2 Minimization

As mentioned before, all the weights are fixed values due to the normalization of each energy term and proper initial parameters after preprocessing. This allows each energy term to take action simultaneously during the minimization process and helps the algorithm to converge rapidly.

We rewrite the normalized energy function as a sum of weighted non-linear functions

$$F(\bar{X}) = \sum_{i=1}^{10} W_i E_i^2(\bar{X})$$

where \bar{X} is the parameter vector.

The *Levenberg-Marquardt* (*L-M*) method [6] is used to deal with the minimization problem. The basic iteration of this method is given via equations

$$X_{k+1} = X_k + d_k \quad (1)$$

$$(J_k^T W J_k + \lambda I) d_k = -J_k^T W F_k \quad (2)$$

where J_k is a Jacobian matrix for the non-linear functions $E_i(\bar{X})$, and λ is a adjustable coefficient.

A single value decomposition technique is used to solve the step size d_k from Eq (2), as after preprocessing the initial parameters are in the neighborhood of the correct values. The main advantage of using the *L-M* method is that it speeds up the procedure of the minimization and it takes only a few iterations for the template to converge to the correct position. In Eq. (2), if $\lambda \rightarrow \infty$, the *L-M* method performs like the deepest descendent method; if $\lambda \rightarrow 0$, the *L-M* method performs like the Newton's method. In our case, λ is chosen as a small value. This will make the algorithm not only have the convergence speed of the Newton's method, but also more robust than the Newton's method.

The key point of our minimization scheme, which differs from Yuille's scheme, is that the template fitting process is not deviled into several epochs and all the parameters of the template are updated in each iteration. The main advantage of the deformable template over other local processing algorithm is that it makes use of global information to improve the reliability of the processing. If energy terms for the deformable template are processed sequentially, in a few beginning epochs some information is not involved in the procedure of the minimization and the advantages of the global processing are not fully explored. In our scheme, all the global information is utilized simultaneously to avoid the template trapped in some undesired configurations. The following experimental results will demonstrate the effectiveness of this scheme.

4. EXPERIMENTAL STUDIES

The experiments were implemented on real images using a 33MHz 486 computer. The extraction of valley, peak and edge fields was done in the preprocessing stage. Also, the eye windows were found during the preprocessing, which takes about 3-4 sec.. Based on this eye window, the initial parameters for the eye template were selected.

Fig.3 shows the experimental result with initial parameters of the template based on the eye windows found in the preprocessing. The initial parameters are selected such that the center of the circle and parabolas, \bar{X}_c and \bar{X}_e , are located at the center of the window, and a , b , c and r are proportional to the width and height of the window. It is also natural to choose the orientation θ as 0° . After 9 iterations, the template was deformed itself to the final configuration in Fig.3. The processing time was about 20 sec..

In next experiment, we disturbed initial parameters of the template to some place away from the eye window. Surprisingly, it took only 6 iterations (about 12 sec.) for the template to move to the final state (shown in Fig.4).

To investigate the effect of weight change on the final results, we performed the algorithm to the same image using three different weight schedules. The results show that the effect is minor and negligible (shown in Fig.5).

To illustrate the effectiveness of processing all the energy terms simultaneously, we did an experiment for the comparison of parallel processing with sequential processing. The parameters of the template are the same for both kinds of processing schemes except the radius of the circle. A larger radius than the correct value was chosen for the sequential scheme and a smaller radius for the parallel scheme. For the sequential processing scheme,

only the valley energy term was taken action in the first step. During the fitting process, the circle in the template actuarially moved towards correct position, but the radius of the circle continuously shrunk at the same time (shown in Fig.6). We also let the edge energy term be activated together with the valley energy. When the template reached a steady state, the circle did not shrink to the darkest part of the eye, but the center of the circle did not aligned with the correct position at the same time. For the parallel processing scheme, even though the initial value of the circle radius was smaller than the correct value, it was getting larger during the minimization and finally fitted the iris very well (shown in Fig.7).

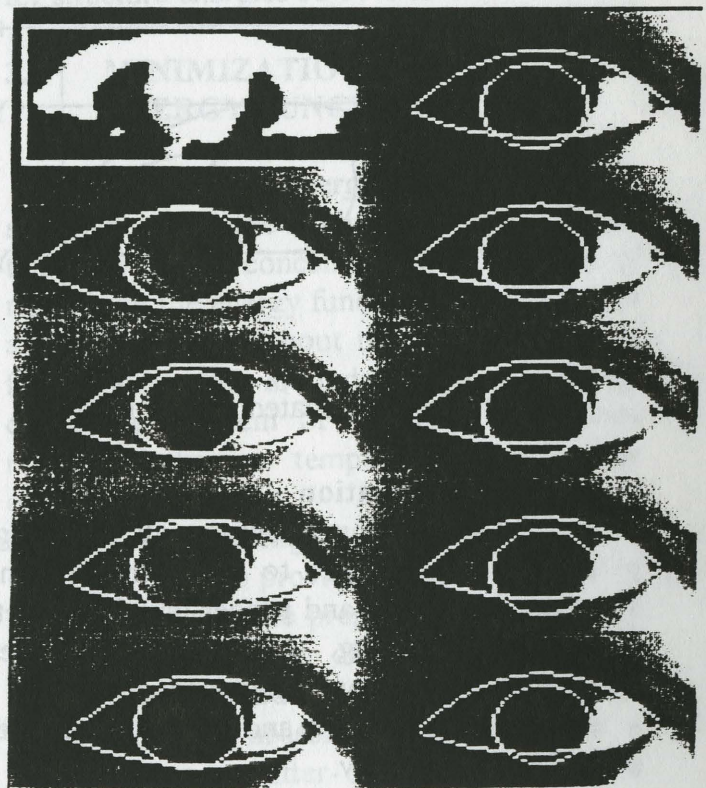


Fig.3 A sequence of eye templates being deformed from top to bottom and left to right. The first frame shows an eye window obtained in the preprocessing.

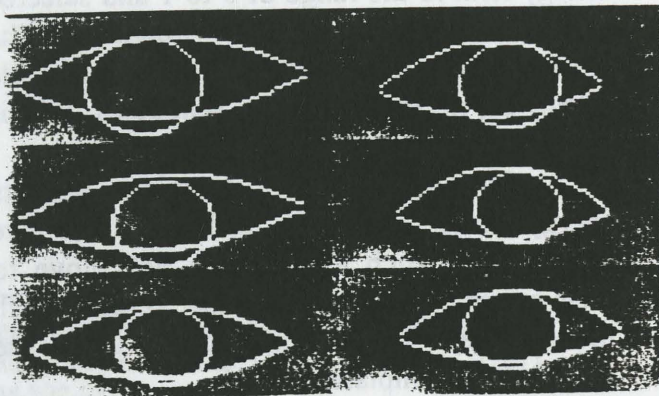


Fig 4 A dynamic sequence of eye templates from top to bottom and left to right with the initial parameters of the template being disturbed.

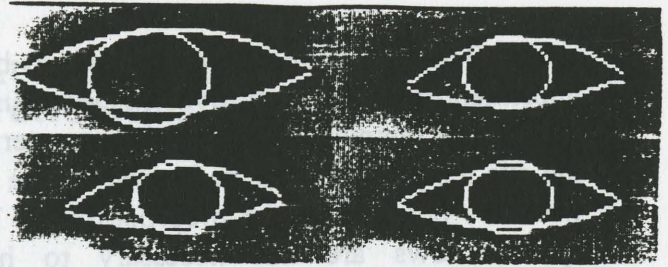


Fig 5 Top left: initial template;
Bottom left: final template with weight ratio 6:1;
Top right: final template with weight ratio 8:1;
Bottom right: final template with weight ratio 10:1.

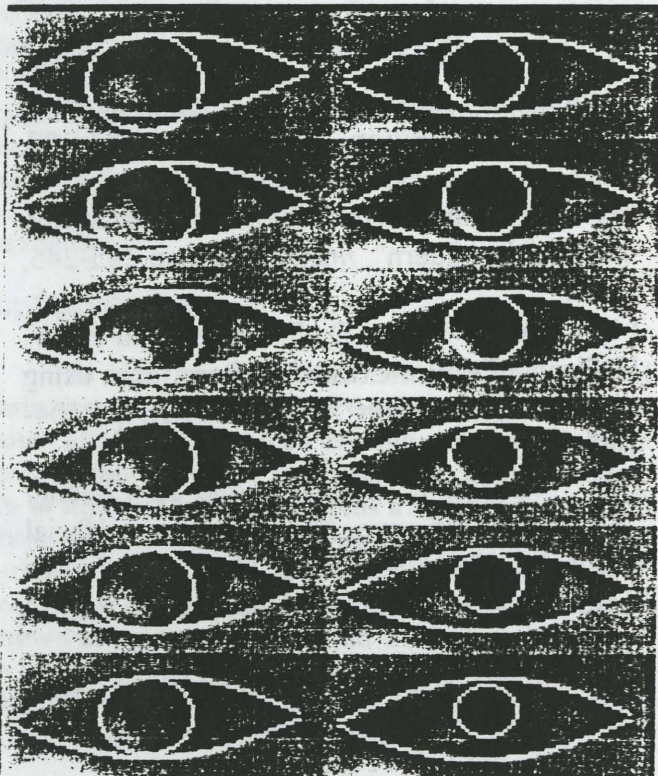


Fig.6 A sequence of eye templates being deformed with only valley energy term activated. The order is from top to bottom and left to right.

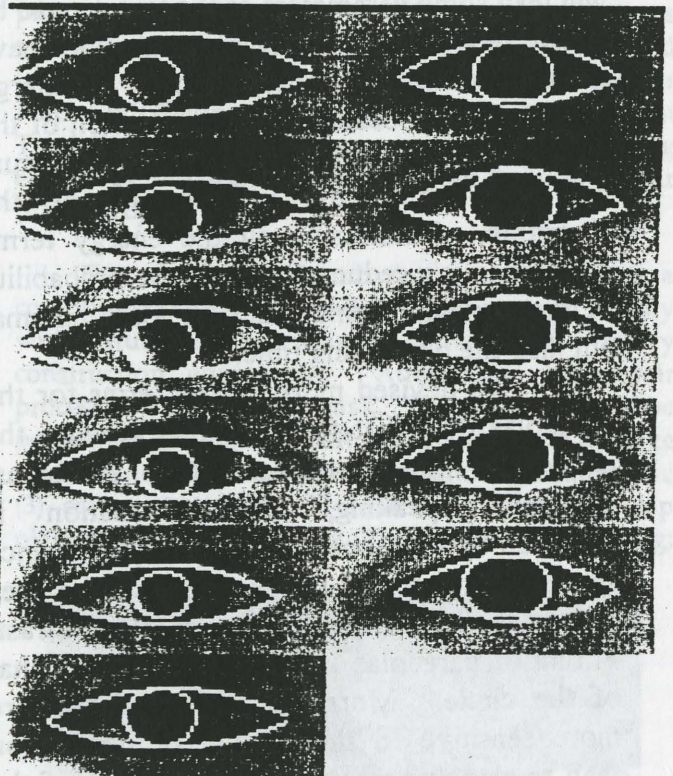


Fig.7 A sequence of eye templates with all the energy terms being minimized simultaneously. The order is from top to bottom and left to right.

5. CONCLUSION

The extraction of eye features has been implemented by an improved method using deformable template. The initial parameters of the eye template are selected based on the eye windows obtained in the preprocessing. The eye windows are not necessary to have accurate spatial positions. The template can be deformed to catch the actual eye features by suitable devised energy function.

Usually, there are many local minima in the parameter space in the image case. We include several conflicting factors in the energy terms and let them be activated in the minimization process to prevent the eye template from being stuck in local minima. Minimizing the energy function sequentially will lead some parameters of the template to be overly deformed since other constraints have not taken action. Further, the order of energy terms being proceeded and the selection of the moment when the transition of epochs occurs will also effect the performance of the algorithm. Minimizing all the energy terms simultaneously reduces not only the probability of the template being stuck in local minima, but also the processing time.

We devised more energy terms for the parabolas than that for the circle because the circle is more localized than parabolas. Generally speaking, less information is provided by the upper and lower eyelids since shades exist around the eyelids and the area around the lower eyelid has very low contrast. Fitting of parabolas is more difficult than that of the circle. Moreover, the parabolas are more sensitive to the change of orientation than to the change of other parameters of the parabolas. That is why we devised three energy terms to capture the correct orientations of the parabolas.

In order to avoid weights to be application dependable, we normalized each

energy term to the range of 0 to 1 and selected two different value of the weights. The experimental results have shown a negligible effect on different weight schedules. This makes our algorithm practical in applications.

The modified eye template and its energy terms can also deal with the eye movement in such a case eyes will move to their extreme positions. Further research will focus on eye movement tracking using the modified eye templates.

REFERENCES

- [1] Nixon, "Eye spacing measurements for facial recognition", *SPIE Proc. 575, Applications of Digital Image Processing VIII*, pp.279-285, 1985.
- [2] I. Craw, H. Ellis and J. R. Lishman, "Automatic extraction of face-features", *Pattern Recognition Lett. 5*, pp.183-187, 1987.
- [3] R. Buhr, "Analyse und klassifikation von gesichtsbildern", *ntzArchiv 8*, pp.245-245, 1986.
- [4] A. L. Yuille, D. S. Cohen and P. W. Hallinan, "Feature extraction from faces using deformable templates", *Proc. CVPR*, pp.104-109, 1989.
- [5] X. Xie, R. Sudhakar and H. Zhuang, "Estimation of eye features from facial images", *The 4th Annual Conf. on Recent Advances in Robotics*, pp.73-80, Boca Raton, 1991.
- [6] L. E. Scales, "Introduction to Non-Linear Optimization" *Springer-Verlag New York Inc.*, pp.115-118, 1985.

CONSTRUCTION AUTOMATION: NAVIGATION OF AN AUTONOMOUS VEHICLE

Carl D. Crane III
Center for Intelligent Machines and Robotics
University of Florida, Gainesville, Fla. 32611
904-392-0814

Allen Nease
U.S. Air Force Civil Engineering Support Agency
Tyndall Air Force Base, Fla. 32403
904-283-3725

ABSTRACT

This paper describes the current Air Force research program which is aimed at the development of an autonomous or remotely supervised heavy equipment excavator which is to be applied to the repair of bomb damaged runways. The technology developed under this program is applicable to a wide range of applications to include the remote handling or clean up of nuclear wastes and construction or digging tasks on the moon. This paper focuses on one aspect of the total system automation, i.e. autonomous navigation.

BACKGROUND

The U.S. Air Force has identified the need of resuming launch and recovery operations within four hours after an attack on an existing runway. Damage to the runway surface is expected to include craters up to fifty feet in diameter plus a multitude of small chips, or spalls, in the runway surface. Unexploded ordnance will likely be present. Additionally, scatterable mines will likely be air delivered and strewn along the runway amid the debris in a effort to impede repair operations. Such operations would obviously be further impaired by the presence of chemical and/or biological agents.

Recent advances in repair equipment have significantly upgraded repair capabilities. This mechanization of the repair process has resulted in fewer pieces of dedicated equipment, and consequently fewer (though usually much more skilled) human operators to affect the repair. At present, the Air Force developed Multi-Function

Excavator (MFE) (see Figure 1) can perform clearing, spreading, upheaval removal, compaction and leveling.

A logical extension of this mechanization program was the development of an autonomous system which could perform these dangerous and labor intensive tasks. Under the direction of the Air Force Civil Engineering Support Agency (AFCESA) at Tyndall Air Force Base, a research and engineering effort was initiated entitled the Rapid Runway Repair (RRR) project.

The repair of a bomb damaged runway is a complicated task which is difficult to perform rapidly with humans operating conventional heavy construction equipment. Automating the repair process is a challenging task. The repair steps to be performed during the RRR operation are (1) damage assessment; (2) unexploded ordnance neutralization; (3) crater repair; (4) fill material hauling; (5) cap placement; (6) spall repair; (7) clearing and sweeping;



Figure 1: Multi-Function Excavator

and (8) marking. The most demanding task from an automation point of view is the crater repair task. Fill material must be spread and compacted, and then leveled to within one half inch of the nominal runway surface.

The current project is focused on achieving a 1994 demonstration of the crater preparation task. The subtasks to be demonstrated include navigating to the crater to be repaired, removing uplifted concrete from around the lip of the crater, and removing large pieces of debris from the crater itself. This paper will focus on the navigation task.

PROBLEM STATEMENT

The navigation problem statement can be simply defined. It is assumed that the current position and orientation of the repair vehicle is known. Further, it is assumed that crater locations have been determined from a prior damage assessment of the runway. In this analysis, a crater is described by an n-sided polygon which surrounds the damaged area. This polygon may be concave and polygons may overlap. Lastly, a position to navigate to is specified, either by a human supervisor, or by a hierarchical control computer.

Based upon the given information, the objective is to (1) plan an efficient path to the goal, and (2) to then move along this planned path without driving into one of the craters or striking an unexpected object (either stationary or moving) which may be on the runway.

NAVIGATION TEST VEHICLE

The navigation system is being developed on a surrogate vehicle named the Navigation Test Vehicle (NTV) which is shown in Figure 2. Once the navigational algorithms are successfully tested on this platform, the sensors and software will be transferred to the excavator vehicle.

The NTV consists of a Kawasaki Mule which has been modified for computer control. Actuators have been placed on the four functions of (1) steering, (2) throttle, (3) brakes, and (4) transmission. Closed loop control is exercised on each of the four actuators via a VME based computer system which is operating under the VxWorks operating system.

OFF-LINE PATH PLANNING

Path planning has been the subject of much published research [1→5]. The current problem is a planar application of path planning and is accomplished by expanding the obstacles and then treating the vehicle as a point. Figure 3 shows the original polygonal obstacles and the expanded obstacles. The amount of expansion is equal to the radius of a circle that circumscribes the NTV.

Path planning is then accomplished by using a hierarchical A* search algorithm using the obstacle vertices which are visible from the current vehicle position as candidate move-to points. A cost is associated with each of these possible move-to points which equals the sum of the distance travelled to get to the point plus the straight line distance from this move-to point to the goal. The move-to point which has the lowest cost is chosen as the next point for the vehicle to move to. All visible obstacle vertices from this point are then considered as the next potential move-to point. The process of identifying and moving to the lowest cost point is repeated until the goal can be directly reached by a straight line segment. Figure 3 shows the resulting path of a typical application of this algorithm.

PATH EXECUTION

Two important problems must be addressed if the vehicle is to successfully navigate to the goal. First, the vehicle must have some accurate means of determining its position and orientation on the runway. Without this knowledge, it is impossible to



Figure 2: Navigation Test Vehicle

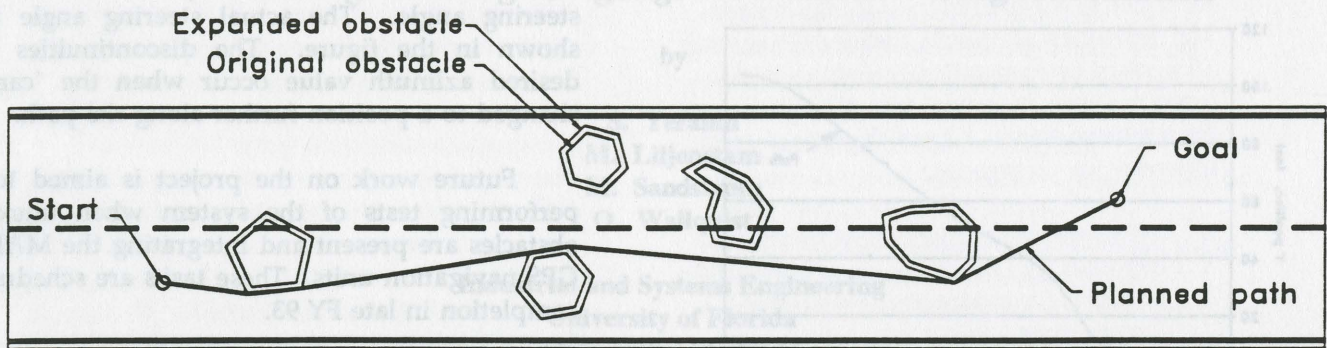


Figure 3: Off-line Path Planning

keep the vehicle on the previously planned path. Second, the vehicle must have some means of detecting objects that are obstructing its path. Once these unexpected obstacles are detected, the originally planned path must be modified to avoid this new obstacle.

The first problem of determining the vehicle's position is being addressed in two ways. An inertial navigation system named MAPS (Modular Azimuth Positioning System), manufactured by Honeywell, Inc., has been mounted on the NTV. This system communicates with the VME controlling computer via an RS-422 synchronous communications protocol. The MAPS can return the vehicle's position and orientation at a rate of approximately six hertz. The MAPS system is currently being operated with a resolution of one foot in the North/South direction and of one quarter foot in the East/West direction. Tests with the unit show that an error of approximately two feet exists after the MAPS has moved a distance of approximately two thousand feet.

The second means of determining the vehicle's position involves the use of a Global Positioning Satellite System (GPS). Wintec, Inc. of Ft. Walton Beach, Florida, is implementing a differential GPS system which will be mounted and tested on the NTV. Tests with the GPS system to date have shown that it can very accurately determine vehicle position (a resolution of inches) at a rate of approximately one hertz.

The detection of unexpected obstacles is currently being accomplished with ultrasonic transducers. An array of sixteen sensors are mounted around the NTV. These sensors have an effective

range of approximately 7.5 meters and can all be fired and updated at a rate of three hertz. The ultrasonic sensors provide a basic obstacle avoidance capability. The range is not adequate for vehicle speeds in excess of five miles per hour. Other obstacle detection sensors such as laser scanners and vision will be investigated in the future for application on the NTV.

Currently, only the MAPS is mounted on the NTV to provide position and orientation information. Effective navigation of pre-planned paths has been accomplished by using the MAPS to support a "carrot navigation" approach. The controlling computer designates a point which lies on the pre-planned path as a "carrot". The NTV steering and throttle are controlled via a standard PID control algorithm to direct the vehicle towards the "carrot". Once the NTV approaches within approximately seven feet of the "carrot", the "carrot" is moved further along the path. If the ultrasonic sensors detect an unexpected obstacle, the "carrot" is moved to the right or left in order to avoid this obstacle and to avoid the a priori known craters.

This "carrot navigation" method has proven to be very straightforward and effective. It has allowed for modular software development in that navigation along the path or navigation to avoid unexpected obstacles are very similar.

RESULTS AND CONCLUSIONS

The results of a sample test case are shown in Figures 4 and 5. For this case, the vehicle was commanded to move from a position of (0,0) and an orientation of 0 degrees measured from the X axis to a position of (100,100) ft and an orientation of 0

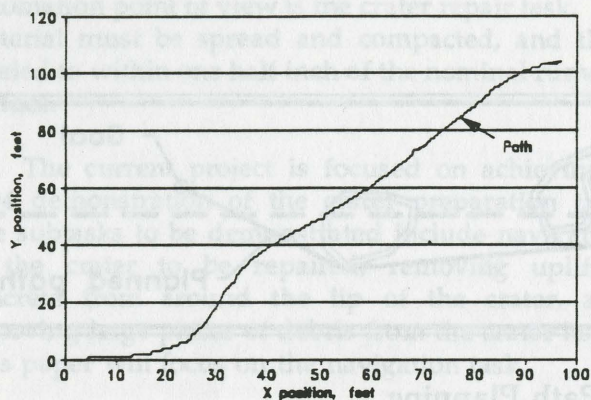


Figure 4: Results- Position Output

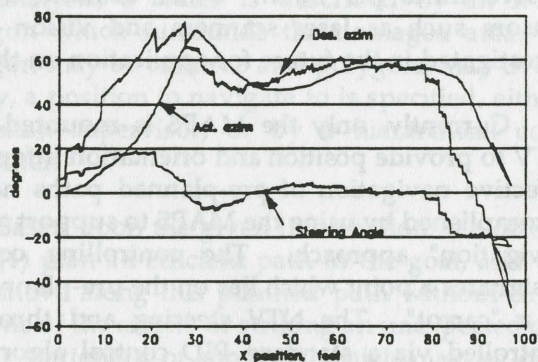


Figure 5: Results- Steering Control

degrees. There were no unexpected obstacles during this test.

An off-line plan was generated to move the vehicle to the goal position. Figure 4 shows the output from the MAPS as the vehicle traverses along the path. It is interesting to note that the resolution of the MAPS (one foot in the North/South direction (y axis for this run) and one quarter foot in the East/West direction) is clearly visible in the figure.

Figure 5 shows the results of the steering control algorithm. The desired azimuth represents the angle from the current vehicle position to the "carrot". The actual azimuth is data returned by the MAPS. The difference between the desired and actual azimuth

represents the error that is being used to influence the steering angle. The actual steering angle is also shown in the figure. The discontinuities in the desired azimuth value occur when the "carrot" is changed to a position further along the path.

Future work on the project is aimed towards performing tests of the system when unexpected obstacles are present and integrating the MAPS and GPS navigation units. These tasks are scheduled for completion in late FY 93.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the support of the U.S. Air Force Civil Engineering Support Agency, Tyndall Air Force Base, Florida. Further the authors wish to acknowledge the important contributions of David Armstrong, Rodolfo Casiple, Art Rankin, and John Webb who are basing their Master's theses on the work being done on the project. Finally the authors wish to acknowledge the contributions of Dr. Joseph Duffy, Dr. Michael Griffis, Mr. Vann Chesney, and Mr. John Duffy.

References

- [1] Lozano-Perez, T. and Wesley, M.A., 1979, "An Algorithm for Planning Collision-free Paths among Obstacles," *Commun. ACM*, vol. 22, pp. 560-570.
- [2] Takahashi, O., and Schilling, R.J., 1989, "Motion Planning in a Plane Using Generalized Voronoi Diagrams," *IEEE Trans. RA*, vol. 5, No. 2.
- [3] Noborio, H., Naniwa, T., and Arimoto, S., 1990, "A Quadtree-Based Path Planning Algorithm for a Mobile Robot," *Journal of Robotic Systems* 7(4), pp. 555-574.
- [4] Khatib, O., 1985, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *IEEE International Conference on Robotics and Automation*, St. Louis, pp. 500-505.
- [5] Borenstein, J., and Koren, Y., 1990, "Real-Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments," *IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 572-577.

A Visual Programming Language for Manufacturing Automation

by

S. Yeralan
M. Liljenstam
M. Sandstrom
O. Wallquist

Industrial and Systems Engineering
University of Florida
Gainesville, FL 32611

Acknowledgments

This research project was partially funded by a grant from Microchip Technology, Inc.

Abstract

This project explores the possibility of utilizing visual programming to simplify the programming of embedded microcontrollers to be used in manufacturing automation. A survey of existing visual programming languages is provided, and an iconic data flow language for microcontroller programming is designed and implemented. A microcontroller application is described by an acyclic recursive data flow graph, with primitives added for control flow constructs, interrupt functions, memory and hardware objects such as ports and timers. A library function is provided in the programming environment for the storage of reusable function modules. The graph is compiled into a generic microcontroller language which closely resembles assembly language. A translator can then be written to translate the generic program code into code for a specific microcontroller.

Introduction

This project is part of an ongoing effort at the department to develop an industrial control language for manufacturing automation that takes advantage of the powerful facilities offered by microcontrollers. Earlier projects include the development of a prototype for a fuzzy logic control language compiler based on the C language [1]. We wish to explore the possibility of utilizing visual programming to simplify the microcontroller programming task.

Over the last 10 years, through the appearance of inexpensive graphic displays, there has been a rapidly growing interest in using visual means to program computers. It has proven difficult to create a good general purpose visual programming environment however, so most of the more successful attempts have been directed towards limited application domains or education.

From the viewpoint of the user, the control language for manufacturing automation must fulfill the following criteria [2]:

1. Be programmable at a level higher than assembly language and be easily understood by users
2. Be easily modifiable and maintainable
3. Be capable of implementing both Boolean and analog control
4. Be robust, stable, and responsive, with a performance comparable to PID/PLC control
5. Generate code for speedy execution necessitated by real-time applications
6. Generate compact code for memory efficiency
7. Allow full access to on-chip facilities such as interrupts, ports and timers
8. Allow code development for applications with multiple microcontrollers

A Survey of Visual Programming Languages

The possibility to use visual languages in man-machine interfaces evolved with the development of inexpensive graphic screens and interactive pointing devices in the late seventies. The importance of this new area soon became

clear and the first papers on this subject were submitted in the beginning of the last decade. The term visual languages traditionally denotes all kinds of languages that deal with visual representations, even pure textual languages used for graphical output. Chang [3] recognizes the following four categories:

1. Languages that support visual interaction
2. Languages for processing visual information
3. Languages for programming with visual expressions
4. Iconic visual information processing languages

The visual programming language developed in this study is of category 3. Visual programming is, as defined above, the act of programming with visual expressions. This does not exclude the existence of textual elements in the language used. A pure Visual Programming Language (VPL) would be a language where the primitives as well as the means of combination are represented graphically, and no keyboard input is needed at all. However, keyboard input often speeds up the programming task, and most of the existing VPLs make use of textual input, to various extents.

Shu [4] proposes a three dimensional profile to characterize the different properties of a VPL. The three different axis correspond to the language level, the scope of the language and the visual extent respectively. Visual extent denotes degree of visualization. The pure VPL mentioned above being at one extreme. The scope ranges from general-purpose languages down to special-purpose languages. The level of a language describes the level of abstraction built into the language primitives.

An abundance of different VPLs has been reported in the last 10 years. Though it is not within the scope of this report to review all the VPLs reported in the literature, or even to mention them all, we intend to give a wide range of examples from the field. Languages that exemplify conceptual differences are examined in more detail. For a more complete overview of VPLs we refer to the earlier surveys of the area that have been published [9 - 11, 5, 3, 6].

As pointed out by Raeder [6], usually one of four different aspects of a program has been depicted when describing it diagrammatically: the control flow, the data flow, the data structures, or the topology.

The control flow describes the order in which different tasks in a program are performed. The classic visualization of this is the flowchart diagram. Every connection in a flowchart represents a transmission of control. This is also the traditional way to visualize a program or algorithm, mostly because it has become a standard and thereby programmers can expect other programmers to understand it. In spite of this, there does exist more structured variants of flow charts. One example of this is Nassi-Schneiderman diagrams [7], in which the code is instead structured blockwise. Here, the diagram itself corresponds to the control constructs of control flow. The transition diagram is a perfect way to describe a simple, automaton-type program, but is not as useful for large applications. Pict [12], International Visual Language [13], PIGS [4], and GDS-11 [14] are examples of VPLs based on control flow.

A serious drawback with all pure control flow descriptions, is they do not clarify a program's data dependencies. In the data flow model the way data propagates through the program is represented visually. A data flow program graph has nodes that represent operations and edges that represent the data dependencies between them [8]. A pure data flow model has no global updatable memory. Instead all data transitions are immediately visible in the diagram. An operation is enabled as soon as it has access to all of its required input, i.e., as soon as they are computed. Enabled operations produce a set of output values as a function of the input. LabVIEW [15], HI-VISUAL language [16-20], Fabrik [21], InterCONS [22], Gabriel [23], Show and Tell [4], ESTL (Enhanced Show and Tell) [24], DataVis [11], and Hyperflow [25] are visual languages based on the data flow paradigm..

In addition to data and control flow there are other paradigms. GRClass [26] and Objectcraft [27] are based on the data structures paradigm. Tinkertoy [28], VennLisp [29], IBGE (Icon-Based Graphical Editor) [30]. Dialog [4], are visual languages based on the program topology

paradigm. Visual Toolset [31], Mondrian [32], PROGRAPH [4], viz [33], and ALEX [34, 35] are visual programming languages which use a combination of the programming paradigms.

There are still many conceptual challenges in and shortcomings of visual programming languages [36]. These can be summarized as below.

- Lack of Formal Specifications ([37], [38])
- Poor Portability
- The Use of Unstructured Constructs ([9])
- High Space Demands of VPLs
- Difficulty in Visualizing Software

VLM - A Visual Language for Microcontrollers

This study developed a visual programming language for microcontrollers to facilitate programming microcontrollers to undertake industrial automation and control tasks. The VLM language is made up of functional modules and data flow connections between them. The data flow dependencies along with the control flow function modules determine the execution order between modules. The only exception to this is interrupt function modules that are triggered by the outside hardware. This means that modules belonging to unconnected sections of the data flow graph that do not belong to a control flow structure module, have no inherent order of execution. They could in principle be executed in parallel (if they are free from interacting side effects).

The execution order of modules A and B is determined by either:

- 1) A path from A, or a module enclosing A, to B, or a module enclosing B.
- 2) A sequence construct over A and B. In all other cases the execution order is undefined, sacrificing determinacy if there are interacting side effects. There are four types of primitives in the language: hardware modules, memory modules, data flow actors, and abstractions/control

We have chosen to use the recursive data flow model with the addition of control flow modules, interrupts, and side effects (to communicate with hardware and interrupts). The data flow connections are typed as: bytes or bits. The hardware modules are: port read/write, timer

read/write, set timer prescaling, read status register and interrupt enable/disable. Ports, timers, status bits and interrupts have labels describing them and hardware designators referring to the specific hardware object of the microcontroller to run the program. Memory is accessed as global individual registers, tables or stacks. All of which are referred to by their labels. Tables and stacks must be initialized to allow static allocation of memory. At initialization, their maximum size is required and tables can be set to hold initial values. The data flow actor operations include arithmetic (addition, subtraction, multiplication, division, modulo), comparisons (equal, less, greater, ...), shift/rotate/swap nibbles, logic (and, or, not, ...) and type conversions between bits and bytes. The data flow actors have side effects only in that the arithmetic and shift/rotate operations modify the status register and that the shift/rotate through carry operations are dependent on the contents of the status register. In this category is also a module for constants. The abstractions/control modules are modules for grouping together other modules. Abstraction modules simply enclose other modules to create macros and functions. Control modules are used to create sequences, selections and iterations. There is a special module for initializations, in which tables and stacks can be set up. Interrupt modules describe functions that are to be launched by hardware interrupt signals. Initialization and interrupt modules can only be placed on the top level of the program. Since only homogeneous synchronous data flow actors are used, there are no problems with consistency in the data flow model.

Interface Design

The principal features of the user interaction with the VLM application, was decided during the design of the VLM language. This was necessary because of the interactive nature of visual languages. Some additional decisions on the interaction are discussed below.

The icons in the program are all mixed modality metaphors. At first we considered using pictorial icons only, but icons that combine picture and text has been shown to be more meaningful than icons that utilize text or pictures only [44]. Moving the objects on the screen is done simply by clicking and dragging, as is becoming a

standard in graphical, interactive applications. To connect two objects the user has to click in one of an object's input or output ports, draw a line to a port on the other object, and release the button. We decided to use the same button for connecting and dragging, instead of using the right mouse button for either of those activities. This is not a limitation since connecting is only allowed when the user clicks in a port, which is rarely done when the intention is to drag the object. To help the user differentiate between these two modes of operation, the cursor changes depending on its mode. The cursor is normally an arrow, but when it enters an object it becomes a hand, showing that the object can be dragged, and when it enters a port it becomes a spoon, showing that a connection can be made. In order to keep the number of menu items to a minimum, we implemented the primitives parametrized. The user is able to alter the parameters through the use of dialog boxes, which are displayed whenever a primitive is double clicked.

Programming in VML, using the application, is mostly a process of adding objects to the screen. To support this process we wanted to have a palette displaying the different icons. It would allow the user to add a new object simply by clicking in the palette. Lack of time, however, forced us to use an ordinary menu instead. To connect a data flow from an object to several others, the user has to introduce a split point by double-clicking on a connection, and then connect the other objects to the split point. The split point can also be used to move the connections on the screen to make the program look neater, since we did not have the time to implement a line drawing algorithm. To make the programs reusable we implemented the library function. We would have preferred to implement this as a palette, much like the palette for adding new objects, and with the user able to supply his own icons. Given time, we would also have implemented the multiple select feature, common to most object manipulating applications, combined with multiple cut, copy and paste.

The application was developed in the Microsoft Windows programming environment on a 386 IBM PC personal computer. It is implemented in C++ using the OWL (Object Windows Library) by Borland International, Inc. Object Windows

uses the object-oriented features of C++ to encapsulate parts of the Windows API, insulating the user from the details of Windows programming. Specifically, OWL encapsulates window information, abstracts many Windows API functions and provides automatic message response. We used C++ programming language since the visual language itself is object oriented, with icons being manipulated on the screen. The Borland's OWL application framework provides classes for most of the Windows API library functions, which would facilitate the programming task. The most important part of the implementation was to design purposeful classes which mimic our visual objects.

Every module in the language is represented by an object that has methods for emitting code. The code is generated by recursively traversing the graph, starting at an arbitrary source, emitting internal code for the module and proceeding to all modules connected to the outputs. The process stops when: a sink is reached a module depending on multiple inputs is reached (and those have not been provided yet) When this happens, a new source is selected and code generation proceeds from there. The exceptions to this basic rule are: sequences reaching the output link of an abstraction module Sequences are compiled one time zone at a time.

Modules connected over the border between two time zones are therefore treated as a sink and a source. Abstraction modules work to synchronize the internal code. This means that in order to finish generation of all the internal code of a module before proceeding externally, graph traversing stops upon reaching the output link of an abstraction module with multiple outputs. Once all the internal code has been generated, compilation proceeds from the external outputs. Data connections are allocated a temporary storage register for transfer of data. This is stored in the graph as an attribute of the connection object. The runtime representation of the language is stack oriented to provide efficient memory allocation for functions, and at the same time permitting recursion. In the first internal register the status of arithmetic and shift/rotate operations are saved.

Next, there is a pointer to the current Activation Record (AR). "Global memory" is memory

allocated for global registers and stack pointers. When a function is called, memory is allocated for the outputs in the current activation record, and a new activation record is set up. Setting up an activation record includes setting a pointer back to the previous activation record and copying the input data to the new AR. Before returning, the called function copies the output data back to the calling AR. The external memory is used for stacks and tables. Memory allocation and various symbol tables are handled by a global Memory Manager object. Labels and lines of object code are sent to a Code Pad that writes them to a disk file.

Further Development.

In addition to what has been implemented on the prototype a number of further enhancements to the compiler have been considered:

1. Providing an extensive set of library functions and templates to the end user
2. Adding support for automatic generation of code for multiple microcontroller systems
3. Introducing primitives for fuzzy control in the language
4. Enabling the user to import and export software modules written in other languages
5. Designing and implementing a tool for visual debugging of programs
6. Extending the concept of comments to include graphs and animations
7. Various optimizations on the generated code
8. Improvements to the user interface and programming environment
9. A suggestion for adding object oriented concepts to the language to provide scope restrictions or data objects.

We believe that more work according to what will be outlined here would help towards turning this into a commercial product

Conclusions

A visual language for microcontrollers is designed and a prototype environment for the language is implemented. The combined data flow/control flow model shows promise, but additional development should be attempted to reduce or eliminate shortcomings such as side effects through globally scoped hardware and

memory objects. Another shortcoming is the relative code inefficiency compared to code developed directly in assembly language. This may be dealt with by code optimization as a post compilation process. In terms of the use of visual programming languages for microcontrollers, perhaps the most useful next step is the field testing of the concepts developed in this study. Based on further empirical experience, concepts and the architecture may be modified to best suit the needs of industrial automation and control.

References

- [1] Arrestam, R. and J. Holmlund, A Fuzzy Logic Control Language FLCL for Embedded Controllers, Research Report, Industrial Research Laboratory, University of Florida, 1991.
- [2] Yeralan, S., M. El-Hafsi, and R. Hasan, A Microcontroller Code Generator for Embedded Fuzzy-Logic Control, Research Report 92-15, Industrial & Systems Engineering, University of Florida, 1992.
- [3] Chang, S. K., Visual Languages: A Tutorial and Survey, IEEE Software vol. 4, no. 1, pp. 29-39, January 1987.
- [4] Shu, N. C., Visual Programming New York: Van Nostrand Reinhold Company, 1988.
- [5] Small, C. H., Diagram Compilers Turn Pictures Into Programs, EDN Software Engineering Special Supplement vol. 36 pp. 12-20, June 20, 1991.
- [6] Raeder, G., A Survey of Current Graphical Programming Techniques, IEEE Computer vol. 18, no. 8, pp. 11-25, (August 1985).
- [7] Nassi, I., and B. Schneiderman, Flowchart Techniques for Structured Programming, ACM Sigplan Notices, vol. 8, no. 8, Aug. 1973.
- [8] Agerwala, T. and Arvind, Data Flow Systems IEEE Computer, vol. 15, no. 2, February 1982.
- [9] Myers, B. A., Taxonomies of Visual Programming and Program Visualization, Journal of Visual Languages and Computing vol. 1, no. 1, pp. 97-123, 1990.
- [10] Shu, N. C., Visual Programming, Perspectives and Approaches, IBM Systems Journal, vol. 28, no.4, April 1989.
- [11] Hils, D. D., Visual Languages and Computing Survey: Data Flow Visual Programming Languages, Journal of Visual Languages and Computing, vol. 3, no. 1, 1992.
- [12] Glinert, E. P. and S. L. Tanimoto, Pict: An Interactive Graphical Programming Environment, IEEE Computer, vol. 17, no. 11, pp. 7-25, November 1984.
- [13] Suleiman, K. A. and W. V. Citrin, An International Visual Language, Proceedings of the 1992 IEEE Workshop on Visual Languages. IEEE Computer Society Press, 1992.
- [14] Tao Research Corporation, GDS-11, 39182 Mission Blvd, Suite 205, Fremont, CA 94539
- [15] National Instruments, LabVIEW. 12109 Technology Boulevard, Austin, Texas 78727.
- [16] Yoshimoto, I., N. Monden, M. Hirakawa, M. Tanaka, and T. Ichikawa, Interactive Iconic Programming Facility in HI-VISUAL, Proceedings of the 1986 IEEE Workshop on Visual Languages, IEEE Computer Society Press, 1986.
- [17] Hirakawa, M., M. Tanaka, and T. Ichikawa, HI-VISUAL Iconic Programming Environment, in: Visual Languages and Applications, edited by T. Ichikawa, E. Jungert, and R. Korfhage, Plenum-Press 1990. [18] Hirakawa, M., M. Tanaka and T. Ichikawa, An Iconic Programming System, HI-VISUAL, IEEE Transactions on Software Engineering, vol. 16, no. 10, October 1990.
- [19] Kado, M., M. Hirakawa, and T. Ichikawa, HI-VISUAL for Hierarchical Development of Large Programs, Proceedings of the 1992 IEEE Workshop on Visual Languages, IEEE Computer Society Press, 1992. [20] Hirakawa, M., M. Yoshimi, and T. Ichikawa, A Universal Language System for Visual Programming, Proceedings of the 1990 IEEE Workshop on Visual Languages. IEEE Computer Society Press, 1990.
- [21] Ludolph, F., Y-Y. Chow, D. Ingalls, S. Wallace, and K. Doyle, The Fabrik Programming Environment, Proceedings of the 1988 IEEE Workshop on Visual Languages, pp. 222-230. IEEE Computer Society Press, 1988.
- [22] Smith, D. N., Visual Programming in the Interface Construction Set, Proceedings of the 1988 IEEE Workshop on Visual Languages, pp. 109-120. IEEE Computer Society Press, 1988.

- [23] Lee, E. A., W-H. Ho, E. E. Goei, J. C. Bier, and S. Bhattacharyya, Gabriel: A Design Environment for DSP, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 11, November 1989.
- [24] Najork, M. A. and E. Golin, Enhancing Show-and-Tell with a polymorphic type system and higher-order functions, Proceedings of the 1990 IEEE Workshop on Visual Languages. IEEE Computer Society Press, 1990. [25] Kimura, T. D., Hyperflow: A Visual Programming Language for Pen Computers, Proceedings of the 1992 IEEE Workshop on Visual Languages. IEEE Computer Society Press, 1992.
- [26] Rogers, G., The GRClass Visual Programming System, Proceedings of the 1990 IEEE Workshop on Visual Languages. IEEE Computer Society Press, 1990.
- [27] Objectcraft, Objectcraft, 2124 Kittredge Street, Suite 118. Berkeley, CA 94704.
- [28] Edel, M., The Tinkertoy Graphical Programming Environment, Proceedings of the 1986 IEEE Workshop on Visual Languages. IEEE Computer Society Press, 1986.
- [29] Lakin, F., Spatial Parsing for Visual Languages, in: Visual Languages, edited by S. K. Chang, T. Ichikawa, and P. A. Ligomenides. Plenum Press, 1986.
- [30] Taylor, T. H. and R. P. Burton, An Icon-Based Graphical Editor, Computer Graphics World, vol.9, no.10, pp. 77-82, October 1986.
- [31] Borges, J. A. and Johnson, R. E., Multiparadigm Visual Programming Language, Proceedings of the 1990 IEEE Workshop on Visual Languages. IEEE Computer Society Press, 1990. [32] Lieberman, H., Dominoes and Storyboards: Beyond Icons on Strings, Proceedings of the 1992 IEEE Workshop on Visual Languages, IEEE Computer Society Press, 1992.
- [33] Holt, C. M., viz: A Visual Language Based on Functions, Proceedings of the 1990 IEEE Workshop on Visual Languages. IEEE Computer Society Press, 1990.
- [34] Kozen, D., T. Teitelbaum, W. Chen, J. Field, W. Pugh, and B. Vander Zanden, ALEX-An Alexical Programming Language, in Visual Languages and Applications, edited by T. Ichikawa, E. Jungert, and R. Korfhage, 1990. [35] Myers, B. A., Invisible Programming, Proceedings of the 1990 IEEE Workshop on Visual Languages, pp. 203-208, IEEE Computer Society Press, 1990.
- [36] Brooks, Jr, F. P., No Silver Bullet: Essence and Accidents of Software Engineering, IEEE Computer, vol. 20, no. 4, pp. 10-19, April 1987.
- [37] Meyer, B., Pictures Depicting Pictures. On the Specification of Visual Languages by Visual Grammars, Proceedings of the 1992 IEEE Workshop on Visual Languages. IEEE Computer Society Press, 1992. [38] Selker, T. and L. Koved, Elements of Visual Language, Proceedings of the 1988 IEEE Workshop on Visual Languages, pp. 38-44. IEEE Computer Society Press, 1988.
- [39] Davis, A. L. and R. M. Keller, Data Flow Program Graphs, IEEE Computer, vol. 15, no. 2, February 1982.
- [40] Dennis, J. B., Models of Data Flow Computation, in: Control Flow and Data Flow: Concepts of Distributed Programming, edited by M. Broy, Springer-Verlag Berlin, 1984.
- [41] Ashford Lee, E., Consistency in Dataflow Graphs. Transactions on Parallel and Distributed Systems, vol. 2, no 2. April 1991.
- [42] Matsumura, K. and S. Tayama, Visual Man-Machine Interface for Program Design and Production, Proceedings of the 1986 IEEE Workshop on Visual Languages, pp. 71-80. IEEE Computer Society Press, 1986.
- [43] Wulf, W. A., M. Shaw, P. N. Hilfinger, and L. Flon, Fundamental Structures of Computer Science, Addison-Wesley Publishing, 1981.
- [44] Guastello, S. J., M. Traut, and G. Korienek, Verbal versus pictorial representations of objects in a human-computer interface, International Journal of Man-Machine Studies, vol. 31, pp.-99-120, 1989.

Automation For Plant Tissue Culture

R. C. Harrell

Agricultural Engineering Department, University of Florida

Gainesville, FL 32611

harrell@gator.agen.ufl.edu

INTRODUCTION

Plant tissue culture is the *in vitro* growth of plant cells, organs, or entire plantlets. The advantages of propagating plants through tissue culture include rapid clonal propagation, decrease in motherstock requirements, high production densities and the production of disease free plants. Propagules can be obtained through a variety of tissue culture processes, some of which are commercial micropropagation, somatic embryogenesis, liquid based shoot tip cultures, phototropic culture systems and micro bulb/micro tuber systems. Commercial implementation of plant tissue culture utilizes a labor intensive but robust process that produces centimeter size plantlets called microcuttings. Due to the high production costs of *in vitro* plant propagation (typically \$0.2 to \$0.5 per propagule), the commercial application of plant tissue culture has been limited to high unit value plants such as ornamentals, cut flowers, and certain plantation crops. To expand the commercial applicability of plant tissue culture beyond this limited market, production systems must be developed for tissue culture processes that are more amenable with available automation technology. Somatic embryogenesis is one of these processes.

Somatic embryogenesis is a fundamentally different process than the commercial micropropagation process. The end products from this process are millimeter size plant embryos and not centimeter plantlets. These embryos have the potential to grow into clones of the plant from which an explant was taken. The advantage of somatic embryogenesis is the potential to rapidly produce unattached biological units in high densities. These embryos are more amenable to automation due to their singular nature and uniform shapes.

Somatic embryogenesis begins with a small explant of tissue excised from a mother plant. The explant is sterilized and placed in a petri dish containing a thin layer of solidified growth medium which includes nutrients and growth hormones. After a few weeks aggregates of undifferentiated cells (calli) will appear on the explant. These calli can be proliferated by manually subdividing the aggregates and transferring to fresh growth medium. Sub liter glass vessels containing a liquid growth medium are commonly used in the callus proliferation stage. Somatic embryo development can be initiated by adjusting the hormones in the growth medium. Embryo formation begins with the development of

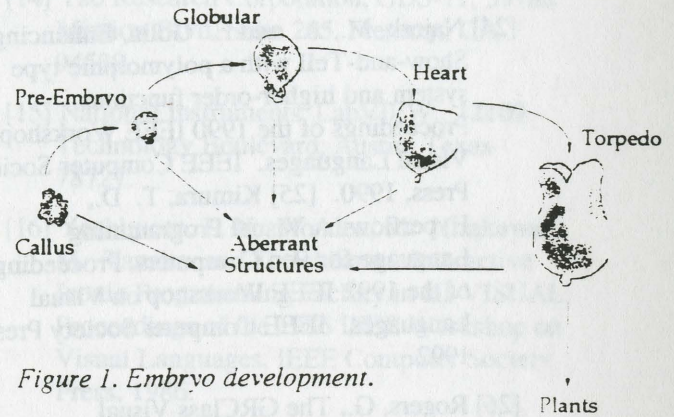


Figure 1. Embryo development.

globular structures on the calli (Figure 1). Later these globular or pro-embryos begin to elongate into heart embryos. The elongation continues and torpedo embryos are formed which are characterized by cotyledon and radical primordia. Torpedo embryos can be harvested for development into plantlets *ex vitro*. At any stage of development the normal process can be disrupted generating aberrant embryos which will not develop into mature plants.

The heterogeneity of embryo maturity and quality typically found in somatic embryogenesis cultures has impeded the commercialization of this propagation technique. This variability necessitates identifying viable embryos in culture and separating them from immature embryos and aberrant structures prior to *ex vitro* processing. For plate culture systems a machine vision guided, robot harvester was developed in Finland (Hamalainen, 1991). In this paper a fluidic, *in vitro* embryo harvester, developed for bioreactor culture systems, is described.

HARVESTER DESCRIPTION

A schematic of the embryo harvester is shown in Figure 2. The harvester consisted of three components: a machine vision system, an object tracker and a harvest chamber implemented along a vertical section of 3mm square glass tubing coupled to an airlift bioreactor with silicon tubing. When it was desired to harvest embryos a peristaltic pump was activated, drawing the suspension from the bioreactor through the 3mm conduit. Images of a 5mm section of the conduit, located approximately 100mm from the harvest chamber, were periodically acquired and analyzed by the vision system through 15x optics at a pixel resolution of approximately 57 μ m horizontal by 32 μ m vertical. An object tracker monitored the positions and velocities of particles in

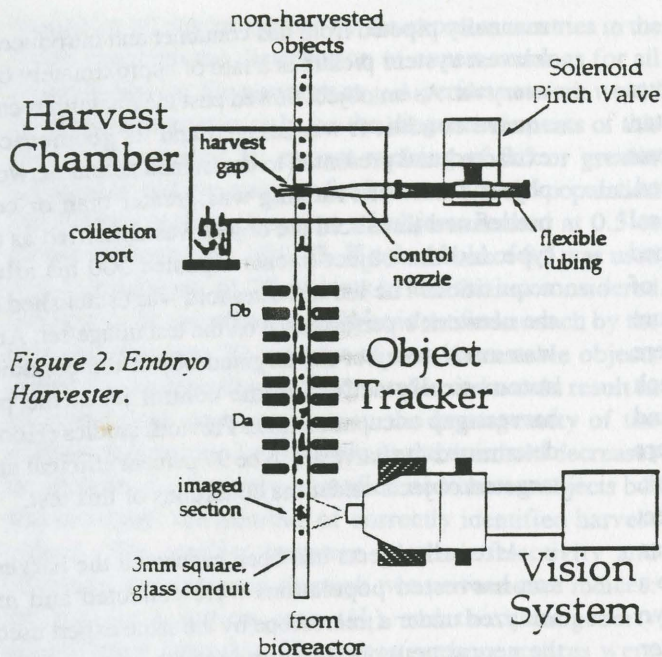


Figure 2. Embryo Harvester.

the conduit between the imaged section and the harvest chamber (Hood, 1992). Non-harvested objects passed through a 6mm gap in the conduit in the medium filled harvest chamber and entered a settlement chamber which separated objects from medium. Particulate free medium was returned to the bioreactor via the peristaltic pump. Harvested objects were deflected out of the gap with a precisely timed injection of culture medium from a control nozzle. After completion of a harvest run harvested particles were collected by opening a valve at a collection port. Non-harvested particles were returned to the airlift reactor by releasing a valve in the settlement chamber. Object sizes ranged from 200 to 1,500 μm and object velocities ranged from 30 to 50 mm/sec.

The vision system consisted of a Force CPU-30 (MC68030, 25MHZ) mounted in a VME crate. Three DataCube VVG-128 image acquisition cards were used to grab a full frame of RGB video from a Pulnix TMC 54GN color camera. A stroboscopic illuminator was used to freeze images of moving particles. Image acquisition and analysis occurred at approximately 10 HZ when no objects were present in the 5mm imaged section of conduit. When an object was detected 500 ms were allocated for image segmentation, feature extraction and object classification. A Bayesian color segmentation algorithm (Harrell and Cantliffe, 1991) was used to obtain binary images of object and background. An eight segment chain code algorithm was used to extract the object's perimeter. Seventeen size and shape related features were extracted from the perimeter data. These features were chosen for their computational simplicity and generality.

The 17 image features were processed by a neural network to rank the desirability of the object. The neural network consisted of 17 input nodes, one for each feature, 14 hidden nodes, a single output node and two bias neurodes connected to the input and hidden layers. The network output ranged from 0 to 1 with higher values corresponding to more desirable embryo morphologies. A threshold was applied to this output such that any object with a ranking greater than or equal to this value was considered harvestable with all others considered as non-harvestable. Neural network and machine vision software were developed with the C language and Microware's OS9 operating system.

An object's position and velocity in the conduit between the imaged section and the harvest chamber were monitored by the object tracker implemented with 30 infrared light-emitting diodes (LEDs) / photo-diode pairs spaced on 2.54 mm centers along the square conduit (Hood, 1992). The presence of an object at some point D_a was detected by the reduction in infrared energy illuminating the photo-diode at this location. When detected at some further point downstream (D_b) an object's velocity was estimated by knowing the elapsed time between detection and elapsed distance between detectors D_a and D_b . An object's arrival time at the conduit gap in the harvest chamber (harvest-gap occupation time) and the time it occupied the imaged section of the conduit (image-window occupation time) were calculated using its current position in the tracking region and velocity estimate. A MIZAR 7120 CPU (MC68020, 16.7 MHZ) performed these calculations and was capable of monitoring several objects in the tracking region and updating their position and velocity estimates at 40ms intervals.

Separation of targeted objects from the remainder of the population occurred in the harvest chamber. A control jet, used to eject objects from the conduit gap, was produced by energizing a solenoid pinch valve connected to flexible tubing which was attached to a control nozzle and terminated with a tubing clamp. When the valve was actuated its top platen approached the lower platen and forced a fixed volume of medium (18 mm^3) through the control nozzle into the harvest gap at a relatively high velocity (760 mm/sec). The timing of the control jet required coordination among the vision system, object tracker, and harvest chamber. When the vision system targeted an object for harvest it signaled the object tracker 500 ms after image capture. For all objects in the tracking region at time of this signal, the object tracker estimated their image-window occupation times. The object having a image-window occupation time nearest to 500 ms was targeted for harvest. The targeted object was harvested by actuating the solenoid valve after its harvest-gap occupation time had elapsed.

PROCEDURE

Neural Network Training

From 15 day old flask suspension cultures used to produce somatic embryos of *Ipomoea Batatas* CV White Star, approximately 70 mature embryos and 70 immature, aberrant and non embryo structures were manually harvested. Representative outlines of these different morphological types are shown in Figure 1. Mature embryos had developed sufficiently for *ex vitro* processing and were typical of embryos to be targeted for harvest from a bioreactor culture environment for conversion to plantlets. All objects were fixed with a distilled water/0.5% bleach solution. Each object was manually introduced into the harvest system and imaged as it entered the imaging section of the square conduit (Figure 2). Object images were archived on a disk. This process was repeated from two to five times per object resulting in 581 images. All object images were ranked by an expert on a scale from 0 to 0.9 with 0.9 corresponding to a prototypical mature embryo and 0 to an obvious non embryo structure. Any object receiving a ranking of 0.5 or greater was considered by the expert as sufficiently developed for *ex vitro* and therefore harvestable. The 581 images were randomly presented to the expert during the rating session to minimize any bias resulting from the sampling process.

A neural network training set was constructed from a random subset of approximately 43% of the images. The remaining 57% of the images were used to construct an independent set to analyze the neural network's performance. 17 geometric features were extracted from each image. Training set feature vectors were randomly presented to the network 11,000 times. The choice of 11,000 training iterations was based on experience with these types of data sets. The mean squared error between the network output node and expert ranking was minimized by adjusting network weights using conventional back propagation training techniques (Rumelhart and McClelland, 1986). Independent assessment of network performance was evaluated with the test set images. Feature vectors were extracted for each object in the test set and presented to the trained network. The output from the neural network for each test object was compared to the expert rating for that object. These results were tabulated and analyzed to select a harvest threshold value for the network which yielded an acceptable compromise between rejection of non-harvest objects and acceptance of harvest objects.

Harvester Testing

Approximately 160 mature embryos and 140 immature, aberrant and non embryo structures were manually harvested from 15 day old flask suspension cultures used to produce somatic embryos of *Ipomoea Batatas* CV White Star. These objects were fixed with a distilled water /0.5% bleach solution and placed in a 100ml glass vial. Each object was

manually pipeted from this container and introduced into the harvest system pickup at a rate of approximately one object every 4s. As an object flowed past the imaging section of the square conduit it was imaged and its geometric features extracted and presented to the trained neural network. If the object's network ranking was greater than or equal to a predefined threshold the object was classified as a harvest type and the object tracker signaled 500 ms after image acquisition. The harvest threshold was established based on the network's performance on the test image set. An attempt was made to eject the targeted object the conduit gap by automatically activating the control jet at the predicted harvest-gap occupation time. Previous studies (Hood, 1992) determined the harvester to be 97 percent efficient at ejecting targeted objects under the conditions of this test.

After all objects had been processed the harvested and non-harvested populations were collected and manually analyzed under a microscope by the same expert used to train the neural network. Analysis results were reported as the number of harvestable objects (expert ranking of 0.5 or greater) and non-harvestable objects (expert ranking less than 0.5) in the harvested and non-harvested populations. The process was repeated for a total of three harvest runs. No appreciable changes in the appearance of the test objects were observed throughout the three replications of the harvester tests. There was, however, a small reduction in the number of particles from one run to the next resulting from the manual manipulations.

RESULTS

Neural Network Training

The performance of the neural network on the test set is summarized in the results matrix shown in Figure 3. Elements in the i^{th} column represent the distribution of the network's

Figure 3. Neural network performance.

		Expert Rankings									
		0	1	2	3	4	5	6	7	8	9
Network Rankings	0	0	1	3	4	0	0	0	0	0	0
	1	0	1	2	3	2	0	0	0	0	0
	2	1	0	0	7	0	0	0	0	0	0
	3	0	4	5	12	2	0	0	0	0	0
	4	0	3	4	20	4	0	0	0	0	0
	5	0	1	4	15	7	0	0	2	0	0
	6	0	2	3	25	20	2	0	3	1	0
	7	0	5	5	19	31	6	1	10	7	0
	8	0	1	0	7	29	6	3	9	3	3
	9	0	0	0	0	3	1	2	0	1	0
		Network Harvest Group									

ranking for all objects ranked by the expert as i ; entries in the j^{th} row represent the distribution of expert rankings for all objects ranked by the network as j . A perfect network would have non-zero entries only on the diagonal elements of this matrix. Recall that an expert ranking of 0.5 or greater indicated an embryo suitable for harvest. The test population had a homogeneity (fraction of objects ranked at 0.5 or greater by the expert) of 0.23. If a threshold of 0.5 was used with this network all 76 objects in the test set considered harvestable by the expert would be identified as such by the network; however 70 percent of non-harvestable objects would also be identified as harvestable and would result in no appreciable improvement in the homogeneity of the selected population. An increase in the threshold decreases the number of incorrectly identified non-harvest objects but also decreases the number of correctly identified harvest objects. The conflict between network selectivity and efficiency was analyzed through two performance indices: the harvest selection rate (H_r) and the homogeneity improvement ratio (α). These performance indices were defined as follows:

$$H_r = \frac{\# \text{ harvestable objects correctly identified}}{\text{Total \# harvestable objects in test set}}$$

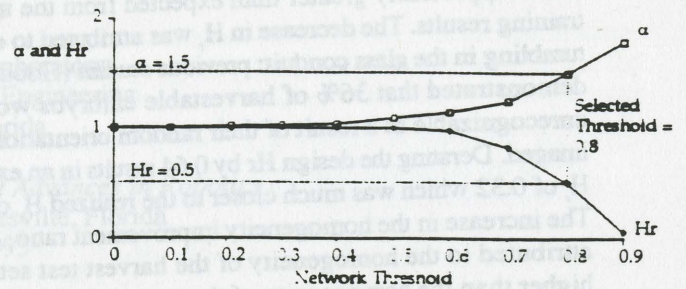
$$\text{and } \alpha = \frac{(1 - H_i)}{(1 - H_r)} \text{ where}$$

$$H_i = \frac{\# \text{ harvestable objects in test set}}{\text{Total objects in test set}}$$

$$H_r = \frac{\# \text{ harvestable objects correctly identified}}{\text{Total objects selected by the network as harvestable.}}$$

H_r is the efficiency of the network at identifying harvestable objects from a given population and has a maximum value of 1. α represents the improvement in homogeneity of the selected population over the homogeneity of the initial population and approaches infinity if the selected population is pure (contains only harvestable objects) and is 1 if the selected population has the same harvest/non-harvest makeup as the initial population.

Figure 4. Performance indices vs. network threshold.



A plot of H_r and α as a function of harvest threshold for this network is shown in Figure 4. For threshold values of 0.4 or less the network selected almost all objects (Figure 4) and both H_r and α were approximately 1. As the threshold was increased H_r geometrically decreased and α geometrically increased. A threshold of 0.8 (H_r of 0.51 and α of 1.51) was selected as a reasonable compromise between network selectivity and efficiency. The effect of this threshold on network performance is shown in the shaded blocks of Figure 3. These blocks represent the portion of the test population correctly classified as non-harvestable (upper left) and harvestable (lower right). The blocks outside of the shaded regions represent the subpopulation incorrectly classified by the network. At a harvest threshold of 0.8 approximately one half of harvestable objects in the network test set were recognized as such by the network. The homogeneity of the selected population was 0.49 compared to the 0.23 homogeneity of the network test population.

Harvest Test

The results from the harvest tests are summarized in Table 1. The homogeneity of the harvest test population, H_i , was 0.56 (56% of the objects in this population were considered harvestable by the expert). After the three harvest runs the homogeneity of the harvested population, H_r , (averaged over the three replications) was 0.88. The harvest selection rate, H_r , was 0.28 (28% of all harvestable objects were harvested) and the homogeneity improvement ratio

TABLE 1. Harvest Test Results.

Run #	Initial Population	H_i	Harvest Objects Harvested	Non-Harvest Objects Harvested	H_r	H_f	α
1	302	0.55	43	7	0.26	0.86	3.2
2	286	0.56	39	5	0.24	0.89	3.8
3	275	0.57	54	7	0.34	0.89	3.7
Summary	863	0.56	136	19	0.28	0.88	3.6

was 3.6. The experimental H_r was appreciably less than the 0.5 expected from network training and the actual harvest α of 3.6 appreciably greater than expected from the network training results. The decrease in H_r was attributed to embryo tumbling in the glass conduit; previous studies (Hood, 1992) demonstrated that 36% of harvestable embryos would be unrecognizable as a result of their random orientation when imaged. Derating the design H_r by 0.64 results in an expected H_r of 0.32 which was much closer to the realized H_r of 0.28. The increase in the homogeneity improvement ratio, α , was attributed to the homogeneity of the harvest test set being higher than the homogeneity of the neural network test set (0.56 compared to 0.23). Using the harvest test set homogeneity to solve for a results in an anticipated homogeneity improvement ratio of 3.7 which is more in line with the harvest results.

DISCUSSION

This work has demonstrated the successful *in vitro* harvest of somatic embryos under non aseptic conditions when embryos are manually introduced into the harvester. Twenty eight percent of harvestable embryos were segregated by the system which resulted in an improvement in population homogeneity of 3.6. To improve upon this performance the accuracy of the machine vision classifier must be improved. Moreover, for this system to be of practical interest to the somatic embryogenesis community on-line, aseptic harvest must be demonstrated.

REFERENCES

Hamalainen, J.. 1991. personal correspondence. VTT, Technical Research Centre of Finland, Laboratory of Electrical and Automaton Engineering, Otakaari 7 B SF-02150 Espoo, Finland (jari.hamalainen@sah.vtt.fi)

Harrell, R.C. and D. J. Cantliffe. 1991. Automated evaluation of somatic embryogenesis in sweet potato by machine vision. in: Levine, R., Vasil, I.K. (editors), Cell Culture and Somatic Cell Genetics of Plants, vol. 8, Automation in Plant Tissue Culture, pp179-195. Academic Press, San Diego

Hood, C.F.. 1992. Automated harvest of somatic embryos. unpublished Ph.D. dissertation, Agricultural Engineering Department, University of FL, Gainesville, FL 32611, 195 pages.

Rumelhart, D.E. and McClelland, J.L. (eds). 1986. Parallel Distributed Processing: Exploration in the Microstructures of Cognition. Vol 1. MIT Press.

CALCULATION OF SINGULARITIES IN NON-REDUNDANT SERIAL KINEMATIC CHAINS

Keith L. Doty

Machine Intelligence Laboratory
Department of Electrical Engineering
University of Florida

Sixth Annual Conference on Recent Advances in Robotics
University of Florida, Gainesville, Florida
April 19-20, 1993

ABSTRACT

Non-redundant manipulators possess Jacobians J with full-column rank in at least one configuration. The unique manipulator singularity surface can be computed from $\det[J^T M_v J] = 0$ for any metric M_v . If one selects $M_v = S^2$, S a diagonal matrix with positive elements on the diagonal, the determinant will indicate all the distinct task spaces whose dimensions equal $\max\{\text{Rank}[J]\}$ and will yield expressions for each task space related surface of singularity. The intersection of all the task space related singularity surfaces equals the unique singularity surface of the manipulator. The latter surface does not depend on the point and frame of reference in which the calculations are made or on M_v while the task spaces and their related singularity surfaces do. Several examples illustrate the ideas presented.

1. INTRODUCTION

Singularities play an important role in inverse velocity and position kinematics ([7],[10],[13]), statics, control, and manipulator workspace analysis. In the robotics literature authors have focused primarily on the analysis of singularities of n -DOF serial kinematic chains for $n \geq 6$. Waldron, for example, [15],[16] examines 6-DOF manipulator Jacobians, determines the analytical conditions for singularity by taking the Jacobian determinant at the midframe and interprets the results geometrically using screw theory. Flueckiger and Bedrossian [5] classify singular configurations of redundant manipulators and determine the situations where it becomes possible to move the robot into a non-singular configuration from a singular one through self-motions alone. Workspace analysis in [6], [9], [14] does not approach the problem by finding the surfaces of singularity of a robot and then determining the Cartesian images, but rather deals directly with the forward kinematics relationship for the position of the end-frame origin of the manipulator. Analysis of singular conditions for manipulators with less than 6-DOF appears not to have received as much attention, even though most commercial manipulators in use possess less than six

degrees-of-freedom. Pai's work [11] provides a notable exception to this observation where he considers a general kinematic map from configuration space to a task space with the same or smaller dimensions. As a practical matter these subspaces tend to possess three dimensions or less. Pai classifies manipulators by the smoothness of their singularity manifolds and addresses design issues arising from singularity analysis.

The work here takes an algebraic approach to the analysis of workspace singularities by determining when the rank of the Jacobian of non-redundant manipulators decreases. Such manipulators necessarily have less than 7-DOF, but the converse is not true. For example, an RRRR planar manipulator is redundant. A methodology in this paper allows one to systematically find those configurations for which the manipulator can no longer control arbitrary motions in a specific velocity task space.

The Robot Velocity Equation

In robotics, the frame-velocity or twist equation

$$V = J \dot{q} \quad (1-1)$$

indicates the linear relationship between the joint-rates \dot{q} and the end-frame twist motion V defined with respect to the robot base frame and computed at the manipulator end-frame origin. The six dimensional twist $V := [v^T \ \omega^T]^T$ consists of the angular velocity ω of the end-frame rigid body and the translational velocity v of the end-frame origin. The matrix J equals the manipulator Jacobian and is a function of configuration q in joint space Q , $J(q)$. The *inverse velocity problem* requires finding a joint-rate vector solution \dot{q}_s which produces a specified frame velocity V . In general, J may be singular or not square. In such cases, given the symmetric, positive-definite matrices M_q and M_v , defined as metrics on the space Q' of joint rates \dot{q} and the space \mathcal{V} of twists, respectively, one might seek the minimum- M_q -norm, M_v -least-squares solution offered by the weighted generalized-inverse [1], [4],

$$\dot{q}_s = J^\# V, \quad (1-2)$$

where $J^\#$ equals the weighted generalized-inverse of J .

Calculation of the Weighted Generalized-Inverse

A full-rank decomposition of the matrix J equals

$$J = F C, \quad (1-3)$$

where $\text{Rank}[J_r] = \text{Column_rank}[F] = \text{Row_rank}[C]$.

While the decomposition (1-3) is not unique, the weighted generalized-inverse [1],[4]

$$J^\# := M_q^{-1} C^T [C M_q^{-1} C^T]^{-1} [F^T M_v F]^{-1} F^T M_v \quad (1-4)$$

is unique. While a weighted generalized-inverse solution to the frame-velocity equation always exists, it is not continuous through a singularity because the weighted generalized-inverse changes if J changes rank: $\text{Rank}[J] = \text{Rank}[J^\#]$, independent of the choice of metrics. Thus, singularity analysis becomes important even with the application of weighted generalized-inverses and that analysis does not depend upon the metrics employed.

Non-Redundant and Redundant Serial Kinematic Chains

In this paper, n -DOF serial kinematic chain robot manipulators will be called *non-redundant manipulators* when its spatial order (SO) equals the number of actuated joints [3], $SO = \max \{\text{Rank}[J]\} = n$. An n -DOF manipulator is *redundant* when

$$SO = \max \{\text{Rank}[J]\} < n.$$

Definition 1 Non-Redundant and Redundant Manipulators

An n -DOF serial kinematic chain is a *non-redundant manipulator* when $SO = \max \{\text{Rank}[J]\} = n$ and *redundant* when $SO = \max \{\text{Rank}[J]\} < n$. The *redundancy* r_m of a manipulator equals

$$r_m := n - SO. \quad (1-5)$$

The Jacobian $J = J(q)$ equals a function of the manipulator joint variable vector q and so it is possible for both $\max \{\text{Rank}[J]\} = n$ and $\text{Rank}[J(q_s)] < n$ for some configuration q_s . When $\text{Rank}[J(q_s)] < n$, the manipulator assumes a *singular configuration* q_s . The spatial degree-of-freedom ($SDOF$) possessed by a manipulator in configuration q is defined as $SDOF(q) := \text{Rank}[J(q)]$. In singular configurations q_s of an n -DOF manipulator, $SDOF(q_s) = \text{Rank}[J(q_s)] < SO \leq n$. Thus, in singular configurations a manipulator's joint DOF always exceeds the dimensions of the twist space describing the motion capabilities of the robot. In other

words, the robot instantaneously possesses redundancy in a singular configuration. The *joint-redundancy* $r(q)$ of a manipulator in configuration q equals

$$r(q) := n - SDOF(q). \quad (1-6)$$

For example, the Jacobian J of the 4-DOF SCARA manipulator (RRRP) has maximum rank four [3], and so the manipulator is non-redundant. But, when the arm is fully stretched out, the rank drops to three. In effect, the second revolute joint cannot contribute to the instantaneous motion and the manipulator is, therefore, redundant in that configuration.

The joint redundancy for a redundant manipulator equals r_m for all non-singular configurations. At singularities q_s , redundant manipulators assume additional joint redundancy because $SDOF(q_s) < SO$. To illustrate, consider a 4-DOF planar manipulator with four parallel revolute axes (RRRR). This manipulator must always be in redundant configurations since $\max \{\text{Rank}[J]\} = 3 < 4$. Every configuration of a redundant manipulator, therefore, must be a joint-redundant one. In contrast, a non-redundant manipulator must have at least one configuration which is not joint-redundant.

Task Space Jacobian

In many cases a roboticist seeks to control a restricted set of the velocity parameters in V . The most common situations discussed in the literature restrict consideration to translational velocity control, $v = J_v \dot{q}$, where J_v equals the first three rows of J , or to angular velocity control, $\omega = J_\omega \dot{q}$, where J_ω equals the last three rows of J . No fundamental objection arises should one choose to consider other combination of the velocity variables. In particular, this paper will focus on the selection of any desired subset of the velocity task variables and designate the resultant vector by V_r ,

$$V_r := \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_k \end{pmatrix} \quad \xi_i \neq \xi_j, i \neq j, \quad (1-7)$$

$$\xi_i \in \{v_x, v_y, v_z, \omega_x, \omega_y, \omega_z\}.$$

By assumption, the side effects on the motion produced by values assumed by the excluded velocity variables will not matter to the problem being solved. Of course, if this is not the case, either one must redefine the task space or solve the problem in a different fashion.

Construct the velocity equation for the restricted velocity vector V_r by deleting the appropriate elements in V and the corresponding rows in the Jacobian J in (1-1) to obtain,

$$V_r = J_r \dot{q} \quad (1-8)$$

and its inverse solution,

$$\dot{q}_s = J_r^\# V_r \quad (1-9)$$

where J_r , the *task space Jacobian*, denotes the modified J . We use (1-8) and (1-9) to represent the various velocity equations, including the extreme case where $V_r = V$ and $J_r = J$.

The *task space* $\mathcal{V}_r := \{V_r\}$ has dimension $k \leq 6$. When $\dim[\mathcal{V}_r] < \text{Rank}[J_r(q)]$, the robot provides redundant control of the task space variables at q . When $\dim[\mathcal{V}_r] = \text{Rank}[J_r(q)]$ the robot provides non-redundant control of the task space variables at q . In the remaining case, $\dim[\mathcal{V}_r] > \text{Rank}[J_r(q)]$, the robot cannot control all the task space variables at q . Configuration q_0 , where $\dim[\mathcal{V}_r] > \text{Rank}[J_r(q_0)]$, is called a *task space related singularity* or *task space related singular point* and the manipulator assumes a *task space related singular configuration*. The manifold of all task space related singular points constitutes the *task space related surface of singularity*. Task space related singularities do not necessarily equal singularities of the manipulator and, in general, are not invariant to translation. This will be proven later.

In this paper we consider only task spaces whose dimensions equal the number of actuated joints of a non-redundant manipulator. The joint-rate space $Q' = \{\dot{q}\}$ and task space \mathcal{V}_r have dimension n and $J_r : Q' \rightarrow \mathcal{V}_r$ equals an $n \times n$ matrix. If $\text{Rank}[J_r(q)] = n$, J_r is invertible at q and any task space requirements can be met by the robot in configuration q . At task space related singularities q_0 , $\text{Rank}[J_r(q_0)] < n$, the task space specifications cannot be realized precisely and a weighted generalized-inverse can be used to determine a minimum-norm, least-squares joint-rate solution to the manipulator task space velocity equation.

The principal objectives of this paper are to develop a theory to

- 1) Find the singular configurations q_s in which a non-redundant manipulator loses motion capability, $\text{Rank}[J(q_s)] < SO = \max\{\text{Rank}[J]\} = n \leq 6$, n the number of degrees-of-freedom of the manipulator,
- 2) Determine all the task spaces with dimension n for an n -DOF non-redundant manipulator and
- 3) Derive the task space related singularity surfaces in configurations space associated with those task spaces.

2. SINGULAR CONFIGURATIONS OF NON-REDUNDANT MANIPULATORS

Consider the case where the Jacobian J of the manipulator equals a $6 \times n$ matrix, $n \leq 6$, and $\text{Rank}[J] = n$, i.e., the manipulator is non-redundant. In a singular configuration q_0 , the null space of J possesses a non-zero vector \dot{q}_1 such that $J(q_0)\dot{q}_1 = 0$. Singular configurations exist if, and only if, the determinant of each $n \times n$ matrix calculated from any n rows of J equals zero. There are $N := \binom{6}{6-n}$ such determinants. These are well known facts from linear algebra [8], however, an informal demonstration will motivate later observations. To prove necessity, assume one such determinant does not equal zero. Its corresponding $n \times n$ matrix can be inverted forcing $\dot{q}_1 = 0$, contradicting the assumption that q_0 a singular configuration. To prove sufficiency, assume all such determinants equal zero. Any subset of row vectors containing n rows must be a linearly dependent set. Consequently, the rank of the Jacobian must be less than n and, hence, the manipulator must be in a singular configuration.

Surfaces of Singularity

The *manipulator surface of singularity* equals the set of all points in configuration space where J does not possess full column rank. The surface of singularity for a non-redundant manipulator with $n \leq 6$ degrees-of-freedom, therefore, equals the intersection of all the surfaces generated by setting the determinant of each $n \times n$ submatrix of J to zero. In other words, configurations which force all non-trivial $n \times n$ determinants of J to zero defines the surface of singularity for J .

The $n \times n$ submatrices of a non-redundant manipulator with n columns can only be constructed by deleting $6-n$ rows. The resultant matrix corresponds exactly to a task space Jacobian J_r . The resulting task space \mathcal{V}_r excludes from V_r those velocity variables corresponding to the crossed-out rows of J . The task space related singularities and surface of singularity, therefore, derive from $\det[J_r]$.

Calculating the Surface of Singularity

One could directly calculate the determinant of each $n \times n$ submatrix J_{rk} of the manipulator Jacobian J in order to compute the task space related surface of singularity and the surface of singularity for the robot. As a practical matter, manual computation of the $k = 1, \dots, N = \binom{6}{6-n}$

such determinants can be tedious and error prone, even when most of the determinants equal zero in robotics applications. The following procedure, however, permits writing a simple program for a symbolic mathematics application package which will develop all the non-zero determinants simultaneously.

For any symmetric and positive-definite matrix M_v ,

$$\text{Rank}[J] = \text{Rank}[J^\#] = \text{Rank}[J^T M_v J]$$

for a non-redundant manipulator. Hence, any configuration q for which J drops in rank must also cause the rank of $J^T M_v J$ to drop, and vice-versa, independent of M_v .

Thus, $\det[J^T M_v J] = 0$ for exactly those configurations in which J is singular, independent of M_v . This proves

Theorem 1

The manipulator surface of singularity may be calculated by $\det[J^T M_v J] = 0$ for any symmetric positive definite matrix M_v . The resultant singularity surface does not depend upon M_v .

Theorem 1 implies that the surface of singularity for a manipulator depends only on the structural properties of the manipulator and joint positions and not the metric. Since the simplest symbolic expression for the manipulator Jacobian of a serial kinematic chain occurs at the midframe [12], symbolic calculation of the surface of singularity using (2-1) will be most tractable at the midframe.

Because $\det[J^T M_v J] = 0$ cannot be influenced by the choice of M_v , select the scaling metric $M_v = S^T S = S^2$, where $S = \text{diag}(\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6)$. Apply the Binet-Cauchy [8] theorem to compute

$$\det[J^T S^2 J] = \det[(SJ)^T (SJ)],$$

$$= \sum_{k=1}^N \det[J_{rk}]^2 f_{rk}(x_1, x_2, x_3, x_4, x_5, x_6), \quad (2-1)$$

where $x_i = \xi_i^2$ and $f_{rk}(x_1, x_2, x_3, x_4, x_5, x_6)$, called an x -factor, equals the product of all the x_i 's, except those with the same indices as the rows deleted from J to produce J_{rk} . Because $f_{rk}(x_1, x_2, x_3, x_4, x_5, x_6) > 0$, $\det[J^T S^2 J] = 0$ if, and only if, $\det[J_{rk}]^2 = 0$ for all k . The latter implies $\det[J_{rk}] = 0$ for all k , the same condition established for singular configurations of J given earlier.

Expression (2-1) is easily computed by a symbolic mathematics software application package and easily identifies the task subspaces associated with J_{rk} , namely, the components of V whose position indices match the

indices in the term $f_{rk}(x_1, x_2, x_3, x_4, x_5, x_6)$. For example, in a 3-DOF non-redundant manipulator, the factor $f_{rk}(x_1, x_2, x_3, x_4, x_5, x_6) = x_2 x_3 x_5$ corresponds to the velocity task space with $V_r = [v_y \ v_z \ \omega_y]^T$.

Invariance of the Manipulator Surface of Singularity

Consider the twist coordinate transformation $J' = GJ$, where

$$G = \begin{pmatrix} R & R B \\ \Phi & R \end{pmatrix} \quad (2-2)$$

includes rotating the frame of reference by R and translating the origin by the vector b which uniquely determines the skew-symmetric matrix B .

Considering that $\det[G] = 1$ and that the metric M_v transforms according to $M_v' = G^{-T} M_v G^{-1}$ [4], we have

$$\det[J'^T M_v' J'] = \det[J^T M_v J].$$

Therefore, the manipulator singularities and the task spaces related singularities are invariant to the point and frame of reference selected for the Jacobian J , provided one employs the transformed metric in the primed frame. In other words,

Theorem 2

The manipulator and task related singularity surfaces are invariant to twist coordinate transformations G , where $J' = GJ$, provided $M_v' = G^{-T} M_v G^{-1}$.

Task space related singular surfaces do depend upon the metric selected (refer to Example 3), but the manipulator surface of singularity does not. Suppose in the primed frame the scaling metric S^2 is used instead of M_v' . The task spaces in the primed frame, in general, no longer equal the task spaces in the unprimed frame, but, by Theorem 1, the configurations q for which

$$\det[J'^T S^2 J'] = \det[J^T (SG)^T SG J] = 0$$

will also force $\det[J^T (S^T S) J] = 0$, and vice-versa. In effect, one can always use a scaling metric to find the manipulator's surface of singularity at any point and frame of reference in which J is computed. This proves

Theorem 3

The manipulator singularity surface consists of all configurations q that satisfy $\det[J^T (S^T S) J] = 0$, regardless of the point and frame of reference used in the calculation of J .

The examples below will illustrate the computational process for finding the surface of singularity for a manipulator and computing the various task related singularity surfaces.

Example 1 RPRR Manipulator

Table 1 presents the DH-parameters for an RPRR manipulator.

Table 1 : Kinematic Parameters for a 4-DOF Robot

Joint	d	θ	a	α
1 r	0	θ_1	0	90°
2 p	d_2	90°	0	90°
3 r	0	θ_3	0	90°
4 r	0	θ_4	0	0°

The midframe Jacobian 2J_4 of this manipulator equals

$${}^2J_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ d_2 & 0 & 0 & 0 \\ 1 & 0 & 0 & s_3 \\ 0 & 0 & 0 & -c_3 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (2-3)$$

This and other Jacobians in this paper can be obtain by applying the tables in [2]. The singularities of 2J_4 occur when

$$\det[{}^2J_4^T S^2 {}^2J_4] = d_2^2 s_3^2 x_2 x_3 x_4 x_6$$

$$+ d_2^2 c_3^2 d_2^2 x_2 x_3 x_5 x_6 + c_3^2 x_2 x_4 x_5 x_6 = 0. \quad (2-4)$$

The factors $d_2^2 s_3^2$, $c_3^2 d_2^2$ and c_3^2 in (2-4) equal the

square of the three non-zero 4×4 determinants of 2J_4 and must simultaneously be zero in order for 2J_4 to loose rank. Since $d_2 \neq 0$, these conditions imply $s_3^2 + c_3^2$

$= 0$, which cannot happen. Hence, 2J_4 never loses rank and there is no surface of singularity for the manipulator.

This manipulator supports three distinct, four-dimensional task spaces in the midframe. The task space velocity equations $V_{ri} = J_{ri} \dot{q}$, $i=1,2,3$, are indicated by the x -factors in (2-4). Namely,

$$V_{r1} = [v_y \ v_z \ \omega_x \ \omega_z]^T, \quad V_{r2} = [v_y \ v_z \ \omega_y \ \omega_z]^T,$$

$$V_{r3} = [v_y \ \omega_x \ \omega_y \ \omega_z]^T, \text{ respectively. Further,}$$

$${}^2J_{4r1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ d_2 & 0 & 0 & 0 \\ 1 & 0 & 0 & s_3 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad {}^2J_{4r2} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ d_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -c_3 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$${}^2J_{4r3} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & s_3 \\ 0 & 0 & 0 & -c_3 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (2-5)$$

Because the coefficient of the x -factor $f_{ri}(x_1, \dots, x_6)$ equals the square of the determinant of the corresponding task space Jacobian ${}^2J_{4ri}$,

$$\det[{}^2J_{4r1}]^2 = d_2^2 s_3^2, \quad \det[{}^2J_{4r2}]^2 = d_2^2 c_3^2,$$

$$\det[{}^2J_{4r3}]^2 = c_3^2. \quad (2-6)$$

the task related singularities for each of the task space velocity equations equals those configurations which force the task space's corresponding x -factor in (2-4) to zero.

The task space $\nu_{ri} = \text{Range}[{}^2J_{4ri}] = 4$, as long as $\det[{}^2J_{4ri}]$ in (2-6) does not equal zero, $i=1,2,3$. The task related singularity surfaces S_{ri} equal $S_{r1} = s_3 = 0$, $S_{r2} = S_{r3} = c_3 = 0$.

Example 2 Surface of Singularity of an RRRP Manipulator

Table 2 presents the DH-parameters for this manipulator.

Table 2 : Kinematic Parameters for a 4-DOF Robot

Joint	d	θ	a	α
1 r	0	θ_1	0	90°
2 r	0	θ_2	a_2	90°
3 r	0	θ_3	a_3	90°
4 p	d_4	0	0	0°

The midframe Jacobian 2J_4 of this manipulator equals

$${}^2J_4 = \begin{pmatrix} 0 & 0 & 0 & s_3 \\ -a_2 c_2 & 0 & 0 & -c_3 \\ 0 & -a_2 & 0 & 0 \\ s_2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -c_2 & 0 & 1 & 0 \end{pmatrix} \quad (2-7)$$

and

$$\begin{aligned} \det[{}^2J_4^T S^2 {}^2J_4] &= a_2^4 c_2^2 s_3^2 x_1 x_2 x_3 x_6 \\ &+ a_2^2 s_2^2 s_3^2 x_1 x_3 x_4 x_6 + a_2^2 c_3^2 s_2^2 x_2 x_3 x_4 x_6 \\ &+ a_2^2 c_2^2 s_3^2 x_1 x_2 x_5 x_6 + s_2^2 s_3^2 x_1 x_4 x_5 x_6 \\ &+ c_3^2 s_2^2 x_2 x_4 x_5 x_6 = 0. \end{aligned} \quad (2-8)$$

The above equals zero if, and only if,

$$c_2^2 s_3^2 + s_2^2 s_3^2 + c_3^2 s_2^2 = s_3^2 + s_2^2 = 0.$$

Therefore, the surface of singularity S_m of this manipulator equals

$$S_m(\theta_2, \theta_3) = s_2^2 + s_3^2 = 0. \quad (2-9)$$

According to (2-8) this manipulator supports six, four-dimensional task spaces in the midframe. Only the task space associated with x -factor $x_1 x_2 x_3 x_6$ allows complete control over the linear velocity of the origin of frame F_2 . The surface of singularity for this task space equals

$$S_{r1}(\theta_2, \theta_3) = c_2 s_3 = 0. \quad (2-10)$$

Observe that the first and fourth terms in (2-8) specify the same singularity conditions for their task space related singularities as do the second and fifth terms and the third and sixth terms.

The task spaces and their surfaces of singularity change if different metrics are used, but the manipulator surface of singularity is invariant to the choice of point and reference frame, as proven earlier. The next example illustrates both results for the manipulator in *Example 2*.

Example 3 RRRP Manipulator Revisited

At the origin of frame three, the manipulator Jacobian, expressed in frame three, equals

$${}^3J_4 = \begin{pmatrix} -a_2 c_2 s_3 & 0 & 0 & 0 \\ a_3 s_2 s_3 & -a_2 - a_3 c_3 & 0 & 0 \\ a_3 c_2 + a_2 c_2 c_3 & 0 & -a_3 & 1 \\ c_3 s_2 & s_3 & 0 & 0 \\ -c_2 & 0 & 1 & 0 \\ s_2 s_3 & -c_3 & 0 & 0 \end{pmatrix} \quad (2-11)$$

The computation of $\det[{}^3J_4^T S^2 {}^3J_4] = 0$ is much more involved than the computation of $\det[{}^2J_4^T S^2 {}^2J_4]$, but it eventually leads to the requirement that

$$\begin{aligned} \det[{}^3J_4^T S^2 {}^3J_4] &= [a_2 c_2 s_3 (a_2 + a_3 c_3)]^2 x_1 x_2 x_3 x_5 \\ &+ [a_2 c_2 s_3]^2 x_1 x_3 x_4 x_5 + s_2^2 [a_2 c_3 + a_3]^2 x_2 x_3 x_4 \\ &x_5 + [a_2 c_2 c_3 s_3]^2 x_1 x_3 x_5 x_6 + [a_2 s_2 s_3]^2 x_2 x_3 x_5 \\ &x_6 + s_2^2 x_3 x_4 x_5 x_6 = 0. \end{aligned} \quad (2-12)$$

The possible four-dimensional task spaces for this manipulator at the origin of frame F_3 , indicated by the indices of the x -factors in (2-12), differ from those task spaces at the origin of frame F_2 specified by the x -factors in (2-8) and so do their surfaces of singularity.

In order for a configuration to be a point of singularity, the last term in (2-12) forces $s_2 = 0$, which, together with the second term forces $s_3 = 0$. These two conditions, however, force all the others terms to zero, hence, the manipulator singularity surface equals

$$S_m(\theta_2, \theta_3) = s_2^2 + s_3^2 = 0, \quad (2-13)$$

which is the same as (2-9). This results illustrates *Theorem 3*.

3. CONCLUSION

The author has attempted to clarify the distinction between redundant and non-redundant manipulators. Essentially, non-redundant manipulators possess a Jacobian with full-column rank in at least one configuration of joint positions. For $n < 6$, one can choose to control different subset of n of the six velocity variables, provided the minor of the Jacobian corresponding to the n selected velocity variables has full rank. A methodology (2-1) for computing the surface of singularity for a non-redundant, n -DOF manipulator, $n \leq 6$, simultaneously finds the determinants of all the $n \times n$ minors of the manipulator Jacobian. The determinants of these minors define the task spaces related surfaces of singularity and characterizes the possible n -dimensional task spaces supported by the robot in the point and frame of reference selected for computing the Jacobian. The manipulator surface of singularity, the task spaces, and the task space related surfaces of singularity are invariant to twist transformations G when the metric M_v transforms according to $M_v' = G^{-T} M_v G^{-1}$ under G and is employed in the prime frame calculations. The task spaces and their associated singularity surfaces, however, depend upon the metric selected. In contrast, the

manipulator surface of singularity is independent of any metric used to calculate it and so one can use the scaling metric at any point and frame of reference and obtain the same results.

REFERENCES

[1] A. Ben-Israel and T.N.E. Greville, *Generalized Inverses: Theory and Applications*. New York: Wiley, 1974.

[2] Keith L. Doty, "Tabulation of the Symbolic Midframe Jacobian of a Robot Manipulator", *Int. J. Robotics Res.*, Vol 6, No.4, pp.85-97, 1987

[3] K. L. Doty, "An Essay on the Application of Weighted Generalized-Inverses in Robotics", *Fifth Conf. on Recent Advances in Robotics*, Florida Atlantic University, Boca Raton, Florida, June 11-12, 1992.

[4] Keith L. Doty, C. Melchiorri and C. Bonivento, "A Theory of Generalized Inverses Applied to Robotics", *Int. J. Robotics Res.*, Vol 12, No.1, pp.1-19, Feb 1993.

[5] K. Flueckiger and N.S. Bedrossian, "Unit Invariant Characterization of Spatial Redundant Manipulator Singularities", *IEEE Proceedings Int. Conf. on Robotics and Automation*, Nice France, pp. 409-414, May 1992.

[6] K.C. Gupta and B. Roth, "Design Considerations for Manipulator Workspace", *ASME J. Mechan. Design*, Vol 104, No. 4, pp.704-712, Oct. 1982.

[7] D. Kholi and M.S. Hsu, "The Jacobian Analysis of Workspaces of Mechanical Manipulators", *Mech. & Mach. Theory*, Vol 23, No. 3, pp.265-275, 1988

[8] Peter Lancaster, *Theory of Matrices*, Academic Press, New York, 1969.

[9] T.W. Lee and D.C.H. Yang, "On the Evaluation of Manipulator Workspace", *ASME J. Mechanisms, Transmissions, Automat. Design*, Vol 105, pp. 70-77, March 1983.

[10] F.L. Litvin, Zhang Yi, V. Parenti Castelli and C. Innocenti, "Singularities, Configurations, and Displacement Functions for Manipulators", *Int. J. Robot. Res.*, Vol. 5, No. 2, pp. 52-65, 1986.

[11] D.K. Pai and M.C. Leu, "Genericity and Singularities of Robot Manipulators", *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 5, pp. 545-559, Oct. 1992.

[12] M. Renaud, *Contribution à la modelisation et à la commande dynamique des robots manipulateurs*, Ph.D. Thesis, Université Paul Sabatier de Toulouse, 1980.

[13] K. Sugimoto, J. Duffy and K.H. Hunt, "Special Configurations of Spatial Mechanisms and Robot Arms", *Mechanism Machine Theory*, Vol 17, No. 2, pp. 119-132, 1982.

[14] D.C.H. Yang and T.W. Lee, "On the Workspace of Mechanical Manipulators", *ASME J.*

Mechanisms, Transmissions, Automat. Design, Vol 105, pp. 62-69, March 1983.

[15] K.J. Waldron, S.L. Wang and S.J. Bolin, "A Study of the Jacobian Matrix of Serial Manipulators", *ASME J. Mechanisms, Transmissions, Automat. Design*, Vol 107, pp. 230-238, June 1985.

[16] S.L. Wang and K.J. Waldron, "A Study of the Singular Configurations of Serial Manipulators", *ASME J. Mechanisms, Transmissions, Automat. Design*, Vol.109, pp.14-20, March 1987.

STUDY OF THE JACOBIAN OF SINGLE-BEAM LASER TRACKING SYSTEMS

Hanqi Zhuang and Zvi S. Roth
Robotics Center and Department of Electrical Engineering
Florida Atlantic University, Boca Raton, FL 33431

Zhijia Qu
Department of Electrical Engineering
University of Central Florida, Orlando, FL 32816

Abstract

A laser tracking system has the potential of being one of most accurate target tracking and coordinate measuring devices. An important issue related to the control of the laser tracking system is its tracking ability. A Jacobian matrix relating target velocities to gimbal angle velocities based on an ideal gimbal-mirror model is derived in this article. A measure is defined to investigate the system's tracking robustness. The trackable space of the system is determined by a symbolic approach and verified by numerical simulation.

1. Introduction

A laser tracking system consisting of laser interferometers, optics and electromechanical mirror-positioning devices has the potential of being one of most accurate tracking and coordinate measuring devices [1-5]. To achieve this objective, important issues in the control of laser tracking systems have to be addressed.

It has been the authors' and other researchers' experience that if a target moves too fast in the direction perpendicular or near-perpendicular to the reflecting beam, the laser system may lose tracking of the target. Let us assume for the sake of discussion that the laser beam is guided by a motorized mirror-gimbal mount. One leading cause of a loss of tracking is that the gimbal angle velocities required to move the mirror-gimbal mechanism to follow the target cannot be achieved by the control system. If an infinite gimbal angle velocity is required at a particular target position, then it is said

that the system is singular at this point. This problem may be investigated by studying the structure of the Jacobian that relates the target velocities to the gimbal angle velocities.

In this article, the Jacobian matrix is derived based on an ideal gimbal-mirror model. Both symbolic and numerical approaches are employed to investigate the singularity of the Jacobian and the tracking ability of the laser system. In the symbolic approach, the determinant of the Jacobian is derived and the roots of the determinant are obtained explicitly with the help of a symbolic reduction software package. These roots represent the singularity points of the system. In the numerical approach, a measure for the tracking ability, namely the condition number of the Jacobian matrix, is computed at sampled points of the gimbal angle space. If a very dense map of the condition number is computed, which is possible for the gimbal-mirror mechanism since it has only two joints, a graphical illustration of the tracking ability of the system can then be obtained.

This article is organized in the following manner. In Section 2, the basic operation principles of a laser tracking system are outlined. Section 3 presents the ideal gimbal-mirror model. The Jacobian matrix for the laser system is derived in Section 4. Singularity and tracking ability of the system are discussed in Sections 5 and 6.

2. Operation of a Laser Tracking System

A schematic diagram of a particular laser tracking mirror positioning servo mechanism

currently in use at the Florida Atlantic University Robotics laboratory is shown in Figure 1.

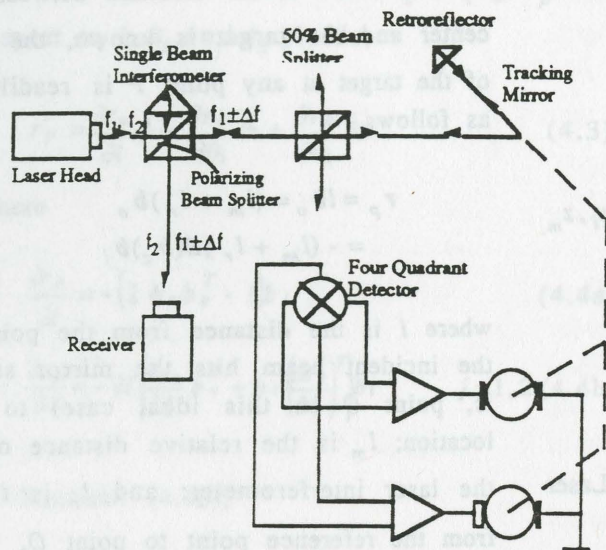


Figure 1 Schematic of a Laser Tracker

Two orthogonally polarized beams at narrowly spaced frequencies, referred to as the reference beam having a frequency of f_2 and measuring beam having a frequency of f_1 , are emitted by the laser head. The reference beam is diverted to the receiver by a polarizing beamsplitter located on the single beam interferometer. The measuring beam proceeds through the interferometer and the 50% beamsplitter and is directed by the tracking mirror to a retroreflector located on the moving target. The returning beam from the retroreflector goes parallel to the incoming beam and enters the 50% beamsplitter from the opposite direction. The beam which passes through the 50% beamsplitter is reflected by the polarizing beamsplitter and emerges into the receiver. If the movable retroreflector changes position, a Doppler frequency shift of Δf occurs, which is then translated through interferometry to a relative displacement reading having a resolution in the order of magnitude of a fraction of a wavelength. The beam reflected by the 50% beamsplitter is

transmitted to a four quadrant detector, which detects the deviations of the returning beam from the center. Error signals drive the servo systems which adjust the angles of the two degrees-of-freedom mirror gimbal.

If rotation angles of the two gimbal axes are measured with sufficient accuracy along with the distance measurement, a single-beam laser tracker will be capable of performing accurate 3D measurements, provided that the system is properly modeled and carefully calibrated.

3. Model of a Single-Beam Tracker

A tracker unit consists of a two-degrees-of-freedom gimbal mount and a mirror. It is assumed that the two gimbal axes are intersecting and perpendicular, that the second gimbal axis lies on the mirror surface, and that the incident beam hits the mirror at its center. This is a highly idealized model. For a more practical model that includes gimbal imperfections, readers are referred to [7]. A simplified geometry of a sequentially-moved mirror is shown in Figure 2. Three Cartesian coordinate frames, the base frame $\{x_b, y_b, z_b\}$, the first link frame $\{x_l, y_l, z_l\}$, and the mirror frame $\{x_m, y_m, z_m\}$, are assigned as shown in Figure 2. The origins of the three frames are placed at the mirror center O . z_b is the rotation axis of the first joint, z_l is the rotation axis of the second joint, and z_m is normal to the mirror surface. When the gimbal is at its home position ($\theta_1 = \theta_2 = 0$), the x axes of the first two frames are coincident with the mirror surface normal. $\{x_b, y_b, z_b\}$ is brought to $\{x_m, y_m, z_m\}$ by the following 4x4 homogeneous transformation,

$${}^bT_m = Rot(z, \theta_1)Rot(x, -90^\circ) \\ Rot(z, \theta_2)Rot(z, 90^\circ)Rot(x, 90^\circ) \quad (3.1)$$

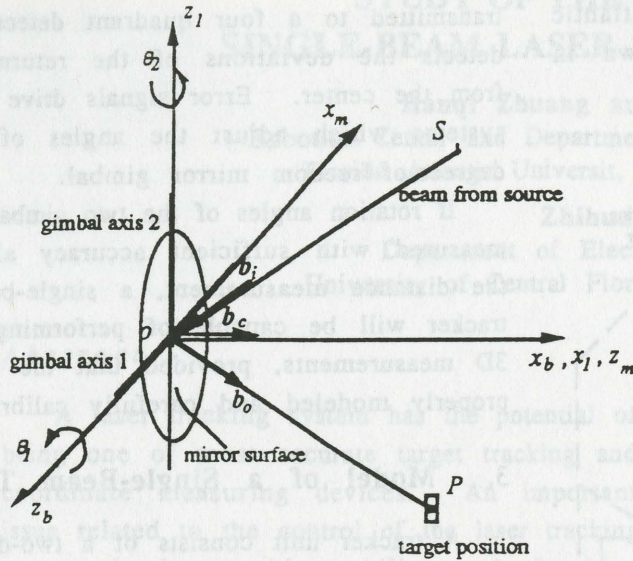


Figure 2 Ideal Geometry of a Single-Beam Laser Tracker

Thus the unit surface normal of the mirror represented in $\{x_b, y_b, z_b\}$, denoted by $b_c \equiv [b_{c,x}, b_{c,y}, b_{c,z}]^T$, is obtained from the first three elements of the third column of bT_m ,

$$b_{c,x} = \cos\theta_1 \cos\theta_2 \quad (3.2a)$$

$$b_{c,y} = \sin\theta_1 \cos\theta_2 \quad (3.2b)$$

$$b_{c,z} = -\sin\theta_2 \quad (3.2c)$$

Denote by b_i and b_o , respectively, the unit direction vectors of the incident and reflected beams with respect to $\{x_b, y_b, z_b\}$. The direction of the incident beam is fixed, but that of the reflected beam varies with the location of the target. If the directions of the incident beam and the surface normal of the mirror are known, the direction of the reflected beam can be computed using the following relationship,

$$b_o = -B(b_c)b_i \quad (3.3)$$

where

$$B(b_c) = \begin{bmatrix} 2b_{c,x}^2 - 1 & 2b_{c,x}b_{c,y} & 2b_{c,x}b_{c,z} \\ 2b_{c,x}b_{c,y} & 2b_{c,y}^2 - 1 & 2b_{c,y}b_{c,z} \\ 2b_{c,x}b_{c,z} & 2b_{c,y}b_{c,z} & 2b_{c,z}^2 - 1 \end{bmatrix} \quad (3.4)$$

is the *mirror image reflection matrix* [6,7].

Let R be a reference target point at which $\theta_1 = \theta_2 = 0$. If the distance between the mirror center and the target is known, the location r_p of the target at any point P is readily computed as follows

$$\begin{aligned} r_p &= lb_o = (l_m + l_r)b_o \\ &= -(l_m + l_r)B(b_c)b_i \end{aligned} \quad (3.5)$$

where l is the distance from the point at which the incident beam hits the mirror surface (that is, point O in this ideal case) to the target location; l_m is the relative distance measured by the laser interferometer; and l_r is the distance from the reference point to point O . To compute the target location r_p , the parameters l_r and b_i need to be obtained by calibration. l_m , θ_1 , and θ_2 are provided by distance and angular measurements of the laser tracking system, and b_c is computed by Equation (3.2) using the measured θ_1 and θ_2 . Equation (3.5) is the *mirror-gimbal model* that can be used for target tracking.

4. The Jacobian of the Single Beam Laser Tracking System

Let $q = [l, \theta_1, \theta_2]^T$ be the augmented joint variable vector of the gimbal-mirror mechanism. A Jacobian matrix J satisfies the following equation,

$$\dot{r}_p = J\dot{q} \quad (4.1)$$

where \dot{r}_p is the Cartesian velocity of the target, and \dot{q} is the augmented joint velocity of the gimbal.

Equation (3.5) can be rewritten as

$$r_p = -l(2b_c(\theta)b_c(\theta)^T - I)b_i \quad (4.2)$$

where $\theta = [\theta_1, \theta_2]^T$. The structure of $b_c(\theta)$ is given in Equation (3.2). Differentiating r_p with respect to l , θ_1 and θ_2 yields,

$$\dot{r}_p = \frac{\partial r_p}{\partial l} \dot{l} + \frac{\partial r_p}{\partial \theta_1} \dot{\theta}_1 + \frac{\partial r_p}{\partial \theta_2} \dot{\theta}_2 \quad (4.3)$$

where

$$\frac{\partial r_p}{\partial l} = -(2b_c b_c^T - I)b_i \quad (4.4a)$$

$$\frac{\partial r_p}{\partial \theta_j} = -2l \left[\frac{\partial b_c}{\partial \theta_j} b_c^T + b_c \left(\frac{\partial b_c}{\partial \theta_j} \right)^T \right] b_i \quad j=1,2 \quad (4.4b)$$

In Equation (4.4b),

$$\frac{\partial b_c}{\partial \theta_1} = \begin{bmatrix} -\sin(\theta_1)\cos(\theta_2) \\ \cos(\theta_1)\cos(\theta_2) \\ 0 \end{bmatrix} \quad (4.5a)$$

$$\frac{\partial b_c}{\partial \theta_2} = \begin{bmatrix} -\cos(\theta_1)\sin(\theta_2) \\ -\sin(\theta_1)\sin(\theta_2) \\ -\cos(\theta_2) \end{bmatrix} \quad (4.5b)$$

Thus, the Jacobian matrix

$$J \equiv \begin{bmatrix} \frac{\partial r_p}{\partial l} & \frac{\partial r_p}{\partial \theta_1} & \frac{\partial r_p}{\partial \theta_2} \end{bmatrix} \quad (4.6)$$

can readily be computed from Equations (4.4) and (4.5).

5. Singularity and Tracking Ability Analysis

A *tracking ability measure* in this paper is defined as the condition number of the Jacobian matrix, which is the ratio of its largest singular value over its smallest singular value. Whenever the condition number is small, the tracking ability of the laser system is good. On

the other hand, whenever the condition number is large, the tracking ability of the system is significantly degraded. A singularity point is the point at which the inverse of the Jacobian matrix does not exist. At such a point, an infinite gimbal angle velocity is required to track the moving object at the direction that is perpendicular to the reflection beam.

One can basically devise two approaches to investigate the singularity issue of the laser tracking system: symbolical and numerical. In the symbolical approach, the determinant of the Jacobian matrix is solved using a symbolical reduction package. Singular points of the laser tracking system are defined as the roots of the determinant. In the numerical approach, the tracking ability measure is used to represent the degree of singularity of the system at a particular target (or equivalently the corresponding joint) position.

5.1 Symbolical Approach to Singularity Analysis

For this purpose, a symbolic reduction software package MACSYMA is used. Rather than deriving the determinant of the Jacobian matrix, it is first triangularized. The roots of the three equations obtained from equating the three diagonal terms to zero are the roots of the Jacobian determinant.

A 3-D *trackable target space* is defined as the target space consisting of all trackable target points. A 2-D *trackable gimbal angle space* is defined as the gimbal angle space in which each point produces a trackable target point. Since at a fixed distance, a target point r_p can be uniquely determined from a given set of θ_1 and θ_2 , one only needs to investigate the behavior of the trackable angle space. The domain of the trackable angle space depends on the direction of the incident beam. However the *volume* of the trackable angle space is independent of the incident beam direction. For example, if the incident beam vector $b_i =$

$[0.7071, 0, 0.7071]^T$, then clearly the trackable angle space is $-90^\circ < \theta_1 < 90^\circ$ and $-45^\circ < \theta_2 < 45^\circ$, assuming that the reflecting beam cannot move across the plane defined by the incident beam and the 2nd gimbal axis (refer to Figure 2). Now if the incident beam direction is $b_i = [0, 0, 1]^T$, the range of θ_1 does not change and that of θ_2 changes to $-90^\circ < \theta_2 < 0^\circ$. Clearly, the volume of the trackable angle space does not change.

By a similar argument, one can also conclude that the relative positions of the singularity points from the incident beam are independent of the representation of the incident beam in the base coordinate frame. Therefore without loss of generality, one can assume a particular value for the incident beam direction. Listed in Equation (5.1) is the symbolic result obtained with $b_i = [0.707, 0, 0.707]^T$. The three equations correspond to zero values of each of the diagonal elements of the triangularized Jacobian.

$$-\sqrt{2}\cos(\theta_1)\cos(\theta_2)\sin(\theta_2) + \sqrt{2}\cos(\theta_1)^2\cos(\theta_2)^2 - \frac{\sqrt{2}}{2} = 0 \quad (5.1a)$$

$$2\cos(\theta_1)\cos^3(\theta_2) - 0.5\cos(\theta_1)\cos(\theta_2)\sin(\theta_2) - \sin^2(\theta_1)\cos^4(\theta_2) + (\sin^2(\theta_1) + 0.5\cos^2(\theta_2)) = 0 \quad (5.1b)$$

$$\cos(\theta_2)(\sin(\theta_2) + \cos(\theta_1)\cos(\theta_2)) = 0 \quad (5.1c)$$

Among Equations (5.1a)-(5.1c), only Equation (5.1c) can provide simple analytical results. The solution derived from Equation (5.1c) is

$$\tan(\theta_2) + \cos(\theta_1) = 0 \quad (5.2)$$

Equation (5.2) defines a singularity curve that runs along the boundary of the trackable gimbal angle space. Figure 3 illustrates the curve defined by Equation (5.2).

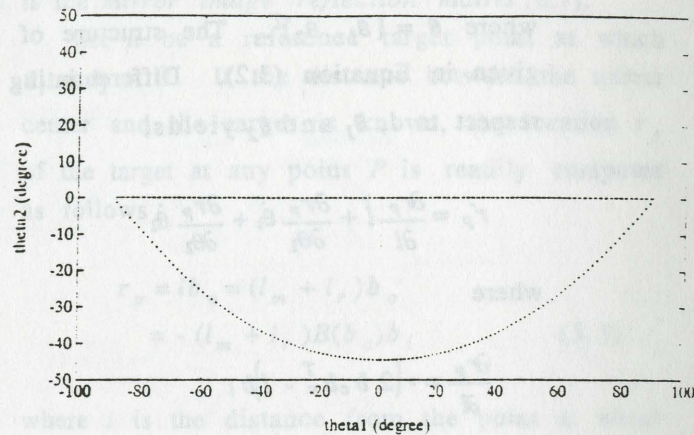


Figure 3 The Singularity Curve of the Laser Tracking System

5.2 Numerical Approach to Tracking Ability Analysis

If the target moves along the direction of the reflected beam, the angles of the gimbal do not change. Least effort is needed for the laser system to track this type of target motion. On the other hand, if the target moves at the direction that is perpendicular to the reflect beam, the system is most susceptible to loss of tracking. To explore the tracking ability of the system, one only needs to investigate the case that the target moves along a spherical surface centered at the mirror center. Mathematically, this is equivalent to fixing the length l of the laser beam. Figure 4 illustrates the singularity curve of the laser tracking system when $l = 1$ and $b_i = [0.707, 0, 0.707]^T$. The curve is basically a blurred version of the curve shown in Figure 3, which is computed from Equation (5.2). Figure 5 shows the condition number of the Jacobian on the θ_1 - θ_2 plane for the case of $l = 1$ and $b_i = [0.707, 0, 0.707]^T$. To improve the resolution of the map, condition numbers that exceed 100 are truncated to 100.

By studying the results of the numerical simulation together with those of the symbolic reduction, the following observations can be made:

1. Equation (5.2) represents part of the boundary of the trackable gimbal angle space. Assuming $b_i = [0.7071, 0, 0.7071]^T$, the trackable space is bounded by the following equations:

$$-90^\circ < \theta_1 < 90^\circ \quad \text{if } 0^\circ \leq \theta_2 < 45^\circ \quad (5.3a)$$

$$\tan(\theta_2) + \cos(\theta_1) > 0 \quad \text{if } -45^\circ < \theta_2 < 0^\circ \quad (5.3b)$$

2. There are no internal singularities inside the trackable gimbal angle space. That is, Equations (5.1a) and (5.1b) either have no solution at all or no solution inside of the trackable domain. The trackability of the gimbal-mirror mechanism is generally good at the internal region of the trackable gimbal angle space. As the target moves closer to the boundary of the trackable space, the trackability of the system is greatly reduced.

3. Although the singularity curve of the system is independent of l , the shape of the tracking ability map depends on the value of the laser beam length l since the condition number of the Jacobian depends on l . Thus, to improve the tracking ability of the laser system, l shall be scaled properly so that the Jacobian matrix has a better condition number at every target point.

4. If the ranges of practical gimbal angles are limited, one can use Equation (5.3) to set up the system's incident beam direction so that the maximum ranges of the gimbal angles can be effectively utilized.

6. Conclusions

A Jacobian matrix has been derived for a laser tracking system based on an ideal mirror-gimbal model. A tracking ability measure,

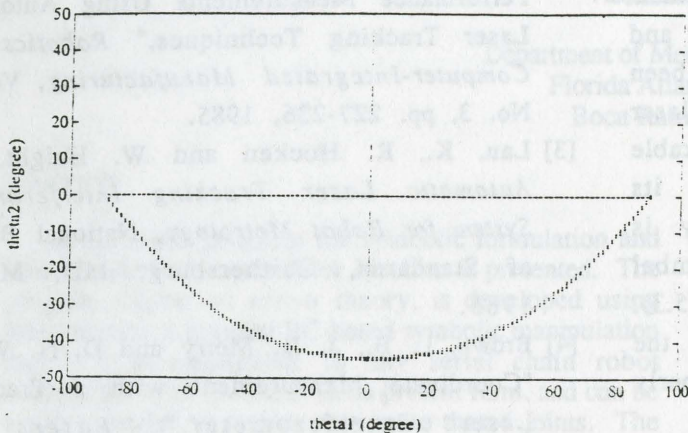


Figure 4 The Singularity Curve of the Laser Tracking System Obtained by the Numerical Approach

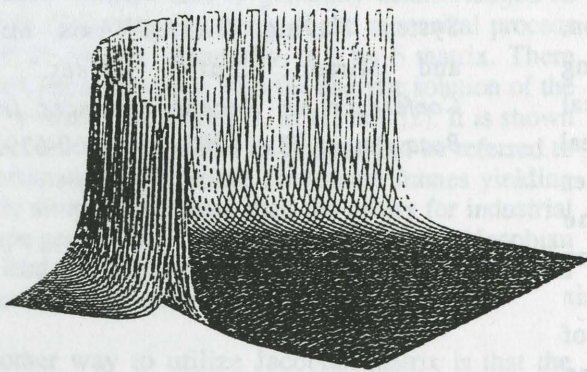


Figure 5 The Tracking Ability Map of the Laser Tracking System

defined as the condition number of the Jacobian matrix, has been used to investigate the tracking ability of the laser tracking system. The boundary of the trackable gimbal angle space has been determined in terms of the determinant of the Jacobian by a symbolic approach and verified by numerical simulation. It has been shown that the tracking ability of the laser system is generally good inside the trackable space. As the target moves towards its boundary, the system's tracking ability is drastically reduced. The ranges of gimbal angles can be maximized using Equation (5.3). Further, to improve the numerical stability, the length l of the laser beam shall be properly scaled.

The Jacobian matrix presented in this article can also be used for the control of the laser tracking system. If a target deviation can be estimated from error signals of the quadrant detector and the geometry of the system, proper gimbal angle corrections can then be computed using the Jacobian matrix.

We have also constructed a Jacobian from a more practical kinematic model given in [7], following the same procedure outlined in Section 4. This Jacobian accounts for the effect of mirror center offset. By numerical simulation, we found that the singularity curve and tracking ability map of the Jacobian are almost identical to those of the Jacobian computed from the ideal mirror-gimbal model. Because mirror center offsets are normally very small, the corresponding variations of the Jacobian elements are also small, compared to their nominal values. Consequently, the variation of the condition number computed from these elements is small. In conclusion, the remarks made in this paper are applicable to more general cases that include mirror-gimbal imperfections.

References

- [1] Lau, K., N. Dagalakis and D. Myers, "Testing," *International Encyclopedia of*

Robotics, Application and Automation, R. C. Dorf and S. Y. Nof, John Wiley & Sons, Inc., 1988, pp. 1753-1769.

- [2] Lau, K., R. Hocken and L. Haynes, "Robot Performance Measurements Using Automatic Laser Tracking Techniques," *Robotics and Computer-Integrated Manufacturing*, Vol. 2, No. 3, pp. 227-236, 1985.
- [3] Lau, K., R. Hocken and W. Haight, *An Automatic Laser Tracking Interferometer System for Robot Metrology*, National Bureau of Standards, Gaithersburg, MD, March, 1985.
- [4] Brown, L. B., J. B. Merry and D. N. Wells, "Coordinate Measurement with a Tracking Laser Interferometer," *Lasers & Applications*, Oct., 1986, pp. 69-71.
- [5] Zhuang, H., B. Li, Z. S. Roth, and X. Xie, "Self-Calibration and Mirror Center Offset Elimination of a Multi-Beam Laser Tracking System," *Journal of Robotic and Autonomous Systems*, 9(1992), pp. 255-269.
- [6] Levi, H., "Applied Optics - A Guide to Optical System Design," Vol. 1, John Wiley & Sons, 1968.
- [7] Zhuang, H. and Z.S. Roth, "Modeling and Self-calibration of a Single-Beam Laser Tracking System Having Gimbal Axis Misalignment and Mirror Center Offset," *The 5th Conference on Recent Advances in Robotics*, Boca Raton, FL, 1992, pp. 660-679.

A Symbolic Program to Formulate and Simplify Jacobians of Robot Manipulators

Shou-Hung W. Ling
Research Assistant

Ming Z. Huang
Assistant Professor

Department of Mechanical Engineering
Florida Atlantic University
Boca Raton, FL 33431

Abstract

A computer program for symbolic formulation and simplification of manipulator Jacobian is presented. The program, based on screw theory, is developed using Mathematica, a popular PC-based symbolic manipulation tool. It is applicable to any serial chain robot manipulator with six d.o.f. in its present form, and can be easily extended to robots with more than 6 joints. The program requires the robot D-H parameters, reference frame in which Jacobian is to be formulated, and joint type as inputs, and automatically generates the Jacobian, its inverse as well as its determinant as the output symbolically. The program can serve as an effective time-saving tool for both teaching and research.

1. Introduction

The Jacobian matrix is the coefficient matrix which relates the joint rates to the velocity of the end-effector. The inverse of the Jacobian matrix is used often in real-time control, and is generally acknowledged as presenting difficulties because of the numerical process involved to inverse, generally, a 6 by 6 matrix. There have been papers dealing with the analytic solution of the inverse problem, for example, in [1] and [2]. It is shown in [2] that the Jacobian and its inverse can be referred to any coordinate frame, some coordinate frames yielding relatively simple Jacobians than the others for industrial robot-type geometries. The simplification of the Jacobian matrix leads to enhanced computational efficiency and makes analytical inversion feasible.

Another way to utilize Jacobian matrix is that the determinant of the Jacobian becomes zero at geometrically singular points. It is possible to find singularity positions by equating an analytical formulation of the determinant to zero.

Because of the widespread use of the Jacobian matrix, it will be a good ideal to have a tool to formulate Jacobian matrix and handle all the transformation calculations. In [6], a PASCAL program was written to generate the symbolic Jacobian matrix. Tables were also

made available for the complete symbolic expressions for the midframe Jacobian of an arbitrary 6 d.o.f. manipulator. The consideration at that time was that computer programs to handle symbolic equations were not easily accessible to most of the people. Now, the symbolic manipulation program is popular and reasonably inexpensive. Mathematica, with its ability to deal with symbolic formulae, is one of the most sophisticated symbolic manipulation programs available. The main objective of this paper is to present a software tool, developed using Mathematica, to symbolically do these transformation calculations.

2. Formulation of the Jacobian

The velocity state of end-effector of a serial manipulator can be expressed by means of the angular velocity of the end-effector and the velocity of reference point fixed in the hand reference frame. The Jacobian matrix can be formulated by differentiating displacement relationship between joint angles and end-effector displacement. Yet the best way to formulate Jacobian matrix is by using screw theory. Screw system theory is discussed in detail in [3], and the formulation of the Jacobian matrix is explained in detail in [1].

If vector μ is defined as the velocity of the point in the end-effector which is instantaneously coincident with the origin of the fixed frame, and ω is the angular velocity of the end-effector, then a Jacobian matrix can be written in the form:

$$\begin{bmatrix} \omega \\ \mu \end{bmatrix} = J\dot{\theta} \quad (1)$$

where

$$J = \begin{bmatrix} u_i \\ \lambda_i \end{bmatrix}$$

$$\lambda_i = \rho_i \times u_i$$

and

J : 6x6 Jacobian matrix

$\dot{\theta}$: 6x1 vector of joint rates.

- u_i : 3x1 unit vector in the direction of the axis of joint i .
- p_i : 3x1 position vector of axis i relative to the fixed frame.

Note that the Jacobian above only applies to robots with all revolute joints. For a robot containing prismatic joints, the following modification is necessary. That is, if joint k is a prismatic joint, then column k assumes the form $[0, u_k]$ and $\dot{\theta}_i$ is replaced by \dot{d}_i in the $\dot{\theta}$ vector.

It is explained in detail in [2] that each column of the Jacobian matrix in Eq (1) is, in fact, a screw. Although the formulation of Eq (1) is quite simple, yet we have to translate all the columns (i.e., the screws) of the Jacobian to the same coordinate frame before we do any calculation. In addition, as shown in [1], by choosing suitable coordinate frame to formulate Jacobian matrix, it can be simplified dramatically.

The coordinate transformation relationship can be simply expressed by using the rotation matrix. If the transformation is intended to be performed from frame i to frame j , then (refer to [6]) :

$${}^j u = {}^j R_i {}^i u \quad (2)$$

$${}^j \lambda = {}^j R_i {}^i \lambda + r_{ij} \times {}^j R_i {}^i u \quad (3)$$

where

r_{ij} : the 3x1 vector from the origin of frame i to the origin of frame j .

${}^j R_i$: the 3x3 rotation matrix, from frame i to frame j .

3. Description of the Program

The effectiveness of this transformation can be best illustrated by the following example. A program using Mathematica has been written for this purpose. The Mathematica program written here is a general program dealing with serial type robot with 6 (or less) screw joints. The user has to do is to provide :

1. the robot D-H parameters
2. the reference frame desired to formulate the Jacobian.
3. the unit screws specifying the joint type of all the axis of the manipulator; namely, $\{\{0,0,1\},\{0,0,0\}\}$ for a revolute joint, and $\{\{0,0,0\},\{0,0,1\}\}$ for a prismatic joint.

Then the program will output the following in symbolic form:

1. the Jacobian matrix based on the chosen coordinate frame.
2. the determinant of the Jacobian.
3. the inverse of the Jacobian.

The flow chart of the program is shown in Appendix A, and the program is listed in Appendix B. It is remarked that the attached program can be directly ported to any machine running Mathematica without modification.

4. Example

The following example demonstrates the use of the symbolic program with the Stanford Arm. The D-H parameters of the Stanford robot arm are listed below (adopted from [5], pp.38):

Stanford robot arm link coordinate parameters				
Joint i	θ_i	α_i	a_i	d_i
1	$\theta_1 = -90$	-90	0	d_1
2	$\theta_2 = -90$	90	0	d_2
3	-90	0	0	d_3
4	$\theta_4 = 0$	-90	0	0
5	$\theta_5 = 0$	90	0	0
6	$\theta_6 = 0$	0	0	d_6

Table 1 : D-H Parameters of Stanford Arm

Since axes 4, 5, and 6 of the arm all pass through the same point, i.e., the origin of frame 3 (refer to [5] pp.38), frame 3 is chosen as the reference frame to formulate the Jacobian matrix. Using the program we developed, upon providing the necessary input described above, the program automatically outputs the robot Jacobian as follows.

Jacobian:

$$\{ \{ 0, -1, 0, 0, -\text{Sin}[\text{th4}], \text{Cos}[\text{th4}] \text{Sin}[\text{th5}] \}, \\ \{ -\text{Sin}[\text{th2}], 0, 0, 0, \text{Cos}[\text{th4}], \text{Sin}[\text{th4}] \text{Sin}[\text{th5}] \}, \\ \{ \text{Cos}[\text{th2}], 0, 0, 1, 0, \text{Cos}[\text{th5}] \}, \\ \{ -(d3 \text{Sin}[\text{th2}]), 0, 0, 0, 0, 0 \}, \\ \{ -(d2 \text{Cos}[\text{th2}]), d3, 0, 0, 0, 0 \}, \\ \{ -(d2 \text{Sin}[\text{th2}]), 0, 1, 0, 0, 0 \} \}$$

The above expression, listed in sequential order of row, is one of the matrix expression used in Mathematica. There are two levels of parentheses. The outer parenthesis represents the whole 6x6 matrix, and each corresponding pair of the inner parentheses represents the row vector (1x6) of the Jacobian matrix.

The determinant and inverse of the Jacobian (based on frame 3) generated by the program are also listed below:

Determinant of Jacobian:

$$-(d3^2 \sin[\theta_2] \sin[\theta_5])$$

Inverse of Jacobian :

```
{
  {0, 0, 0, -(1/(d3 Sin[th2])), 0, 0},
  {0, 0, 0, -(d2 Cos[th2])/(d3^2 Sin[th2]), (1/d3), 0},
  {0, 0, 0, -(d2/d3), 0, 1},
  {-((Cos[th4] Cos[th5])/Sin[th5]), -((Cos[th5] Sin[th4])/Sin[th5]), 1, -((-d2 Cos[th2] Cos[th4] Cos[th5]) - d3 Cos[th5] Sin[th2] Sin[th4] - d3 Cos[th2] Sin[th5])/(d3^2 Sin[th2] Sin[th5]), -((Cos[th4] Cos[th5])/(d3 Sin[th5])), 0},
  {-Sin[th4], Cos[th4], 0, -((d3 Cos[th4] Sin[th2] Sin[th5] - d2 Cos[th2] Sin[th4] Sin[th5])/(d3^2 Sin[th2] Sin[th5]), -Sin[th4]/d3, 0}
  { Cos[th4]/Sin[th5], Sin[th4]/Sin[th5], 0, (d2Cos[th2]Cos[th4]+d3Sin[th2]Sin[th4])/(d3^2 Sin[th2] Sin[th5]), Cos[th4]/(d3 Sin[th5]),0}
}
```

Note that if, instead of frame 3, frame 0 is chosen to formulate the Jacobian, the result is much more complicated. We have included such a result in Appendix C for comparison. It can be seen that an apparent advantage of the program is that it enables the user to identify the reference frame which results in the simplest form of Jacobian simply by 'trying' various frames - a task typically requiring geometric insight if such a program is not available.

5. Discussion

The Jacobian formulation part has been discussed in many papers. So the discussion here will concentrate on the problems we encountered when using Mathematica. There are two constraints which should be carefully managed when using Mathematica (or, may we say, any other symbolic manipulation program): memory and time (refer to [4] and [8]).

Memory space, when using Mathematica, can represent an insuperable barrier in a calculation. If the expression that would generate in a particular calculation is larger than the memory we have available, then we will simply never be able to do that computation on the computer system we are using. There are various ways

that can be used to save memory in a Mathematica session. The first is to remove values that are no longer in use by using *Remove[]* function (refer to [4] for details). The second way to save memory in storing expression is to use the function *Share[]* (refer to [4]). What *Share[]* does is to make sure all expressions that are identical are actually stored in the same place of computer memory. The third way, we can simplify expression by defining some transformation patterns ourselves. In the above example, some basic trigonometric relations are good enough, it can simplify the expression dramatically.

Time is usually not a very critical limitation in the computations. But if we did not define the transformation patterns carefully enough (e.g., a regressive rule exists in the program), Mathematica may never come up with a result. Carefully selecting the transformation patterns and simplifying expressions in appropriate places are two crucial keys in saving time.

6. Conclusion

We have presented a general program for symbolic formulation and simplification of manipulator Jacobian using a popular PC-based symbolic manipulation tool, Mathematica. The program presented here, which is very easy to use, can serve as an effective time-saving tool for both teaching and research. It has been demonstrated that for a typical industrial robot with either parallel or intersecting normal axes, the inversion of Jacobian matrix can be obtained symbolically within ten minutes (running on a Macintosh II). While the idea that Jacobian can be simplified dramatically by choosing an appropriate coordinate frame is well known, it is our understanding that there has been only a few symbolic (as opposed to numeric) implementation of such an algorithm.

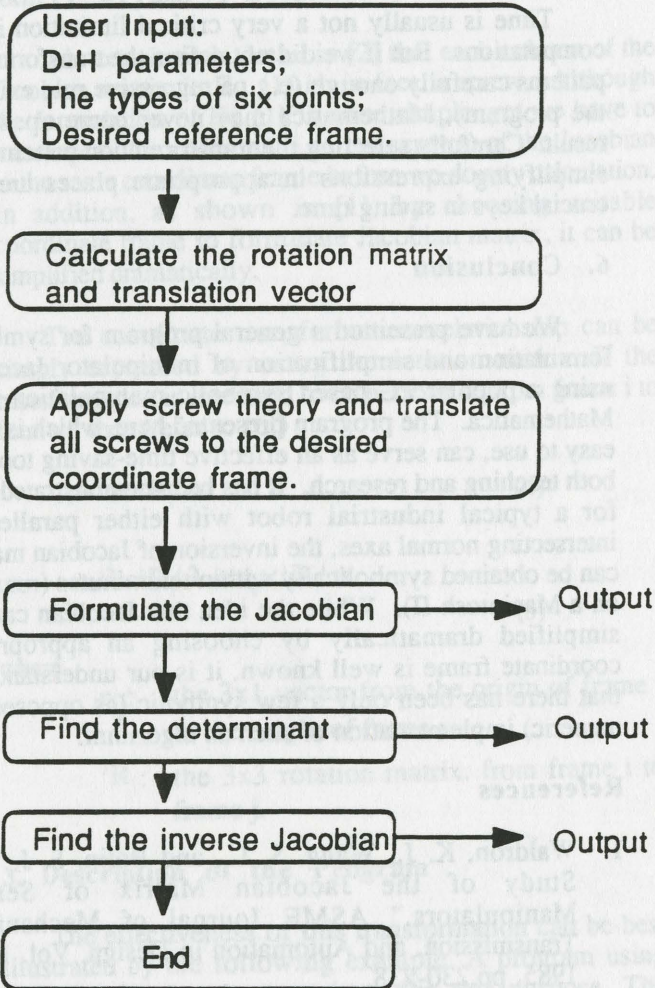
References

1. Waldron, K. J., Wang, S. L., and Bolin, S. J., "A Study of the Jacobian Matrix of Serial Manipulators," ASME Journal of Mechanics, Transmission, and Automation in Design, Vol. 107, 1985, pp.230-238.
2. Hunt, K. H., "Robot Kinematics-A Compact Analytic Inverse Solution for Velocities", Transactions of the ASME, Vol.109, March 1987, pp.42-49.
3. Hunt, K. H., *Kinematic Geometry of Mechanisms*, Oxford, 1978.
4. Wolfram, S., *Mathematica*, Addison-Wesley, 1988.
5. Fu, K. S., Gonzalez, R. C., and Lee, C. S. G., *Robotics*, McGraw-Hill, 1987.
6. Doty, K.L., "Tabulation of the Symbolic Midframe Jacobian of a Robot Manipulator", The International Journal of Robotics Research, Vol.6, No. 4, 1987, pp. 85-97.

7. Huang, M. Z., Class Notes on "Advanced Topics on Robotics", Mechanical Engineering Department, Florida Atlantic University, 1992.
8. Wong, T. L., Class Notes on "Computer Aided Design", Mechanical Engineering Department, Florida Atlantic University, 1992.

Appendix A:

Program Flow Chart



Appendix B:

```

(***** This is a Mathematica Program *****)
(*      A Symbolic Approach to Formulate      *)
(*      and Simplify Jacobian Matrix          *)
(*      by Using                               *)
(*      Mathematica and Screw Theory          *)

```

```

(*----- User Input -----*)
(*----- D-H parameters -----*)
theta={ th1,th2,-Pi/2,th4,th5,th6}

```

```

afa={-Pi/2,Pi/2,0,-Pi/2,Pi/2,0}
arm={0,0,0,0,0,0}
dist={d1,d2,d3,0,0,d6}
(*----- Desired reference frame -----*)
reframe=3
(*----- Types of joint -----*)
$O1={{0,0,1},{0,0,0}}
$I2={{0,0,1},{0,0,0}}
$S23={{0,0,0},{0,0,1}}
$S34={{0,0,1},{0,0,0}}
$A5={{0,0,1},{0,0,0}}
$S6={{0,0,1},{0,0,0}}

```

```

(*----- Transformation Pattern -----*)
TrigRules = {
  Sin[x_]^2+Cos[x_]^2 -> 1,
  Sin[x_] Cos[y_] + Sin[y_] Cos[x_] -> Sin[x+y],
  Sin[x_] Cos[y_] - Sin[y_] Cos[x_] -> Sin[x-y],
  Cos[x_] Cos[y_] + Sin[y_] Sin[x_] -> Cos[x-y],
  Cos[x_] Cos[y_] - Sin[x_] Sin[y_] -> Cos[x+y],
  z_ Sin[x_] Cos[y_] + z_ Sin[y_] Cos[x_] -> z Sin[x+y],
  z_ Sin[x_] Cos[y_] - z_ Sin[y_] Cos[x_] -> z Sin[x-
y],
  z_ Cos[x_] Cos[y_] + z_ Sin[y_] Sin[x_] -> z Cos[x-
y],
  z_ Cos[x_] Cos[y_] - z_ Sin[x_] Sin[y_] -> z
Cos[x+y],
  z_ Sin[x_]^2 + z_ Cos[x_]^2 -> z
}
Share[]

```

```

(*---- Form rotation matrix & translation vectors ----*)
am=Table[0,{6},{4},{4}]
rm=Table[0,{6},{3},{3}]
rotmtx=Table[0,{7},{3},{3}]
r=Table[0,{7},{3},{3}]

n$1=Table[0,{2},{3}]
n$2=Table[0,{2},{3}]
n$3=Table[0,{2},{3}]
n$4=Table[0,{2},{3}]
n$5=Table[0,{2},{3}]
n$6=Table[0,{2},{3}]

```

```

For [i=1,i<=6,i++,
  am[[i,1,1]]=Cos[theta[[i]]];
  am[[i,1,2]]=-Sin[theta[[i]]]*Cos[afa[[i]]];
  am[[i,1,3]]=Sin[theta[[i]]]*Sin[afa[[i]]];
  am[[i,1,4]]=arm[[i]]*Cos[theta[[i]]];
  am[[i,2,1]]=Sin[theta[[i]]];
  am[[i,2,2]]=Cos[theta[[i]]]*Cos[afa[[i]]];
  am[[i,2,3]]=-Cos[theta[[i]]]*Sin[afa[[i]]];
  am[[i,2,4]]=arm[[i]]*Sin[theta[[i]]];
  am[[i,3,1]]=0;
  am[[i,3,2]]=Sin[afa[[i]]];
  am[[i,3,3]]=Cos[afa[[i]]];
  am[[i,3,4]]=dist[[i]];
  am[[i,4,1]]=0;

```

```

am[[i,4,2]]=0;
am[[i,4,3]]=0;
am[[i,4,4]]=1;
rm[[i,1,1]]=Cos[theta[[i]]];
rm[[i,1,2]]=-Sin[theta[[i]]]*Cos[afa[[i]]];
rm[[i,1,3]]=Sin[theta[[i]]]*Sin[afa[[i]]];
rm[[i,2,1]]=Sin[theta[[i]]];
rm[[i,2,2]]=Cos[theta[[i]]]*Cos[afa[[i]]];
rm[[i,2,3]]=-Cos[theta[[i]]]*Sin[afa[[i]]];
rm[[i,3,1]]=0;
rm[[i,3,2]]=Sin[afa[[i]]];
rm[[i,3,3]]=Cos[afa[[i]]]

(*----- Begin to calculate rotmtx01.....*)
(*----- Rotation matrix on new frame -----*)
temp3=IdentityMatrix[3]
For[i=1,i<=reframe,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[1]]=Transpose[temp3] ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=2,i<=reframe,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[2]]=Transpose[temp3] ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=3,i<=reframe,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[3]]=Transpose[temp3] ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=4,i<=reframe,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[4]]=Transpose[temp3] ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=5,i<=reframe,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[5]]=Transpose[temp3] ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=6,i<=reframe,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[6]]=Transpose[temp3] ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=reframe+1,i<=1,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[2]]=temp3 ]
Remove[i]

temp3=IdentityMatrix[3]

```

```

For[i=reframe+1,i<=2,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[3]]=temp3 ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=reframe+1,i<=3,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[4]]=temp3 ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=reframe+1,i<=4,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[5]]=temp3 ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=reframe+1,i<=5,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[6]]=temp3 ]
Remove[i]

temp3=IdentityMatrix[3]
For[i=reframe+1,i<=6,i++,
temp3=temp3 . rm[[i]] //.TrigRules;
rotmtx[[7]]=temp3 ]
Remove[i]

rotmtx[[reframe+1]]=IdentityMatrix[3]
Remove[temp3]

(* ----- Begin to calculate r01.....*)
(* ----- Translation vectors on new frame ----*)
temp4=IdentityMatrix[4]
For[i=1,i<=reframe,i++,
temp4=temp4.am[[i]] ]
Remove[i]
If[1<=reframe,
temp=Inverse[temp4] //.TrigRules;
r[[1,1,1]]=0;
r[[1,1,2]]=-temp[[3,4]];
r[[1,1,3]]=temp[[2,4]];
r[[1,2,1]]=temp[[3,4]];
r[[1,2,2]]=0;
r[[1,2,3]]=-temp[[1,4]];
r[[1,3,1]=-temp[[2,4]];
r[[1,3,2]]=temp[[1,4]];
r[[1,3,3]]=0,
bypass1]

temp4=IdentityMatrix[4]
For[i=2,i<=reframe,i++,
temp4=temp4.am[[i]] ]
Remove[i]
If[2<=reframe,
temp=Inverse[temp4] //.TrigRules;
r[[2,1,1]]=0;

```



```

r[[2,1,2]]=-temp[[3,4]];
r[[2,1,3]]=temp[[2,4]];
r[[2,2,1]]=temp[[3,4]];
r[[2,2,2]]=0;
r[[2,2,3]]=-temp[[1,4]];
r[[2,3,1]]=-temp[[2,4]];
r[[2,3,2]]=temp[[1,4]];
r[[2,3,3]]=0,
bypass2]

```

```

temp4=IdentityMatrix[4]
For[i=3,i<=reframe,i++,
temp4=temp4.am[[i]] ]
Remove[i]
If[3<=reframe,
temp=Inverse[temp4] //.TrigRules;
r[[3,1,1]]=0;
r[[3,1,2]]=-temp[[3,4]];
r[[3,1,3]]=temp[[2,4]];
r[[3,2,1]]=temp[[3,4]];
r[[3,2,2]]=0;
r[[3,2,3]]=-temp[[1,4]];
r[[3,3,1]]=-temp[[2,4]];
r[[3,3,2]]=temp[[1,4]];
r[[3,3,3]]=0,
bypass3]

```

```

temp4=IdentityMatrix[4]
For[i=4,i<=reframe,i++,
temp4=temp4.am[[i]] ]
Remove[i]
If[4<=reframe,
temp=Inverse[temp4] //.TrigRules;
r[[4,1,1]]=0;
r[[4,1,2]]=-temp[[3,4]];
r[[4,1,3]]=temp[[2,4]];
r[[4,2,1]]=temp[[3,4]];
r[[4,2,2]]=0;
r[[4,2,3]]=-temp[[1,4]];
r[[4,3,1]]=-temp[[2,4]];
r[[4,3,2]]=temp[[1,4]];
r[[4,3,3]]=0,
bypass4]

```

```

temp4=IdentityMatrix[4]
For[i=5,i<=reframe,i++,
temp4=temp4.am[[i]] ]
Remove[i]
If[5<=reframe,
temp=Inverse[temp4] //.TrigRules;
r[[5,1,1]]=0;
r[[5,1,2]]=-temp[[3,4]];
r[[5,1,3]]=temp[[2,4]];
r[[5,2,1]]=temp[[3,4]];
r[[5,2,2]]=0;
r[[5,2,3]]=-temp[[1,4]];
r[[5,3,1]]=-temp[[2,4]];
r[[5,3,2]]=temp[[1,4]];

```

```

r[[5,3,3]]=0,
bypass5]

```

```

temp4=IdentityMatrix[4]
For[i=6,i<=reframe,i++,
temp4=temp4.am[[i]] ]
Remove[i]
If[6<=reframe,
temp=Inverse[temp4] //.TrigRules;
r[[6,1,1]]=0;
r[[6,1,2]]=-temp[[3,4]];
r[[6,1,3]]=temp[[2,4]];
r[[6,2,1]]=temp[[3,4]];
r[[6,2,2]]=0;
r[[6,2,3]]=-temp[[1,4]];
r[[6,3,1]]=-temp[[2,4]];
r[[6,3,2]]=temp[[1,4]];
r[[6,3,3]]=0,
bypass6]

```

```

temp4=IdentityMatrix[4]
For[i=reframe+1,i<=1,i++,
temp4=temp4.am[[i]] ]
Remove[i]
If[(reframe+1)<=1,
temp=temp4 //.TrigRules;
r[[2,1,1]]=0;
r[[2,1,2]]=-temp[[3,4]];
r[[2,1,3]]=temp[[2,4]];
r[[2,2,1]]=temp[[3,4]];
r[[2,2,2]]=0;
r[[2,2,3]]=-temp[[1,4]];
r[[2,3,1]]=-temp[[2,4]];
r[[2,3,2]]=temp[[1,4]];
r[[2,3,3]]=0,
bypass7]

```

```

temp4=IdentityMatrix[4]
For[i=reframe+1,i<=2,i++,
temp4=temp4.am[[i]] ]
Remove[i]
If[(reframe+1)<=2,
temp=temp4 //.TrigRules;
r[[3,1,1]]=0;
r[[3,1,2]]=-temp[[3,4]];
r[[3,1,3]]=temp[[2,4]];
r[[3,2,1]]=temp[[3,4]];
r[[3,2,2]]=0;
r[[3,2,3]]=-temp[[1,4]];
r[[3,3,1]]=-temp[[2,4]];
r[[3,3,2]]=temp[[1,4]];
r[[3,3,3]]=0,
bypass8]

```

```

temp4=IdentityMatrix[4]
For[i=reframe+1,i<=3,i++,
temp4=temp4.am[[i]] ]
Remove[i]

```

```

If[(reframe+1)<=3,
  temp=temp4 //TrigRules;
  r[[4,1,1]]=0;
  r[[4,1,2]]=-temp[[3,4]];
  r[[4,1,3]]=temp[[2,4]];
  r[[4,2,1]]=temp[[3,4]];
  r[[4,2,2]]=0;
  r[[4,2,3]]=-temp[[1,4]];
  r[[4,3,1]]=-temp[[2,4]];
  r[[4,3,2]]=temp[[1,4]];
  r[[4,3,3]]=0,
  bypass9]

```

```

temp4=IdentityMatrix[4]
For[i=reframe+1,i<=4,i++,
  temp4=temp4.am[[i]] ]
  Remove[i]
If[(reframe+1)<=4,
  temp=temp4 //TrigRules;
  r[[5,1,1]]=0;
  r[[5,1,2]]=-temp[[3,4]];
  r[[5,1,3]]=temp[[2,4]];
  r[[5,2,1]]=temp[[3,4]];
  r[[5,2,2]]=0;
  r[[5,2,3]]=-temp[[1,4]];
  r[[5,3,1]]=-temp[[2,4]];
  r[[5,3,2]]=temp[[1,4]];
  r[[5,3,3]]=0,
  bypass10]

```

```

temp4=IdentityMatrix[4]
For[i=reframe+1,i<=5,i++,
  temp4=temp4.am[[i]] ]
  Remove[i]
If[(reframe+1)<=5,
  temp=temp4 //TrigRules;
  r[[6,1,1]]=0;
  r[[6,1,2]]=-temp[[3,4]];
  r[[6,1,3]]=temp[[2,4]];
  r[[6,2,1]]=temp[[3,4]];
  r[[6,2,2]]=0;
  r[[6,2,3]]=-temp[[1,4]];
  r[[6,3,1]]=-temp[[2,4]];
  r[[6,3,2]]=temp[[1,4]];
  r[[6,3,3]]=0,
  bypass11]

```

```

temp4=IdentityMatrix[4]
For[i=reframe+1,i<=6,i++,
  temp4=temp4.am[[i]] ]
  Remove[i]
If[(reframe+1)<=6,
  temp=temp4 //TrigRules;
  r[[7,1,1]]=0;
  r[[7,1,2]]=-temp[[3,4]];
  r[[7,1,3]]=temp[[2,4]];
  r[[7,2,1]]=temp[[3,4]];
  r[[7,2,2]]=0;

```

```

r[[7,2,3]]=-temp[[1,4]];
r[[7,3,1]]=-temp[[2,4]];
r[[7,3,2]]=temp[[1,4]];
r[[7,3,3]]=0,
bypass12]

```

```

r[[reframe+1]]=Table[0,{3},{3}]
Remove[temp,temp4]

```

```

(*----- Begin to calculate new screws -----*)
n$1[[1]]=rotmtx[[1]].S01[[1]] //TrigRules
n$1[[2]]=rotmtx[[1]].S01[[2]]+r[[1]].rotmtx[[1]].S01[[1]]
//TrigRules

```

```

n$2[[1]]=rotmtx[[2]].S12[[1]] //TrigRules
n$2[[2]]=rotmtx[[2]].S12[[2]]+r[[2]].rotmtx[[2]].S12[[1]]
//TrigRules

```

```

n$3[[1]]=rotmtx[[3]].S23[[1]] //TrigRules
n$3[[2]]=rotmtx[[3]].S23[[2]]+r[[3]].rotmtx[[3]].S23[[1]]
//TrigRules

```

```

n$4[[1]]=rotmtx[[4]].S34[[1]] //TrigRules
n$4[[2]]=rotmtx[[4]].S34[[2]]+r[[4]].rotmtx[[4]].S34[[1]]
//TrigRules

```

```

n$5[[1]]=rotmtx[[5]].S45[[1]] //TrigRules
n$5[[2]]=rotmtx[[5]].S45[[2]]+r[[5]].rotmtx[[5]].S45[[1]]
//TrigRules

```

```

n$6[[1]]=rotmtx[[6]].S56[[1]] //TrigRules
n$6[[2]]=rotmtx[[6]].S56[[2]]+r[[6]].rotmtx[[6]].S56[[1]]
//TrigRules

```

```

n$1=Simplify[n$1] //TrigRules

```

```

n$1=Flatten[n$1]
n$2=Flatten[n$2]
n$3=Flatten[n$3]
n$4=Flatten[n$4]
n$5=Flatten[n$5]
n$6=Flatten[n$6]

```

```

(*----- Output Jacobian matrix -----*)
(*----- On new frame -----*)
jacobian=Transpose[{n$1,n$2,n$3,n$4,n$5,n$6}]
Print["Jacobian:"]
Print[jacobian]

```

```

(*----- Calculate determinant -----*)
Print["Determinant of Jacobian:"]
Print[Det[jacobian] //TrigRules]

```

```

(*----- Calculate inverse Jacobian -----*)
Print["Inverse of Jacobian:"]
Print[Inverse[jacobian] //TrigRules]

```

Appendix C

Jacobian (if frame 0 is chosen in example):

$$\begin{aligned} & \{0, -\sin[\theta_1], 0, \cos[\theta_1] \sin[\theta_2], \\ & \quad \cos[\theta_1] \cos[\theta_2] \cos[\theta_4] - \sin[\theta_1] \sin[\theta_4], \\ & \quad \cos[\theta_1] \cos[\theta_5] \sin[\theta_2] + \\ & \quad (\cos[\theta_4] \sin[\theta_1] + \cos[\theta_1] \cos[\theta_2] \sin[\theta_4]) \\ & \quad \sin[\theta_5]\}, \\ & \{0, \cos[\theta_1], 0, \sin[\theta_1] \sin[\theta_2], \\ & \quad \cos[\theta_2] \cos[\theta_4] \sin[\theta_1] + \cos[\theta_1] \sin[\theta_4], \\ & \quad \cos[\theta_5] \sin[\theta_1] \sin[\theta_2] + \\ & \quad (-\cos[\theta_1] \cos[\theta_4]) + \cos[\theta_2] \sin[\theta_1] \sin[\theta_4]) \\ & \quad \sin[\theta_5]\}, \\ & \{1, 0, 0, \cos[\theta_2], -(\cos[\theta_4] \sin[\theta_2]), \\ & \quad \cos[\theta_2] \cos[\theta_5] - \sin[\theta_2] \sin[\theta_4] \sin[\theta_5]\}, \\ & \{0, -(d_1 \cos[\theta_1]), \cos[\theta_1] \sin[\theta_2], \\ & \quad \cos[\theta_2] (d_2 \cos[\theta_1] + d_3 \sin[\theta_1] \sin[\theta_2]) - \\ & \quad (d_1 + d_3 \cos[\theta_2]) \sin[\theta_1] \sin[\theta_2], \\ & \quad -((d_1 + d_3 \cos[\theta_2]) \\ & \quad (\cos[\theta_2] \cos[\theta_4] \sin[\theta_1] + \cos[\theta_1] \sin[\theta_4])) - \\ & \quad \cos[\theta_4] (d_2 \cos[\theta_1] + d_3 \sin[\theta_1] \sin[\theta_2]) \sin[\theta_2], \\ & \quad (d_2 \cos[\theta_1] + d_3 \sin[\theta_1] \sin[\theta_2]) \\ & \quad (\cos[\theta_2] \cos[\theta_5] - \sin[\theta_2] \sin[\theta_4] \sin[\theta_5]) - \\ & \quad (d_1 + d_3 \cos[\theta_2]) (\cos[\theta_5] \sin[\theta_1] \sin[\theta_2] + \\ & \quad (-\cos[\theta_1] \cos[\theta_4]) + \cos[\theta_2] \sin[\theta_1] \sin[\theta_4]) \\ & \quad \sin[\theta_5]\}, \\ & \{0, -(d_1 \sin[\theta_1]), \sin[\theta_1] \sin[\theta_2], \\ & \quad -(\cos[\theta_2] (-(d_2 \sin[\theta_1]) + d_3 \cos[\theta_1] \sin[\theta_2])) + \\ & \quad \cos[\theta_1] (d_1 + d_3 \cos[\theta_2]) \sin[\theta_2], \\ & \quad (d_1 + d_3 \cos[\theta_2]) (\cos[\theta_1] \cos[\theta_2] \cos[\theta_4] - \\ & \quad \sin[\theta_1] \sin[\theta_4]) + \end{aligned}$$

$$\begin{aligned} & \cos[\theta_4] (-(d_2 \sin[\theta_1]) + d_3 \cos[\theta_1] \sin[\theta_2]) \\ & \sin[\theta_2], \\ & \quad -((\cos[\theta_2] \cos[\theta_5] - \sin[\theta_2] \sin[\theta_4] \sin[\theta_5]) \\ & \quad (-(d_2 \sin[\theta_1]) + d_3 \cos[\theta_1] \sin[\theta_2])) + \\ & \quad (d_1 + d_3 \cos[\theta_2]) (\cos[\theta_1] \cos[\theta_5] \sin[\theta_2] + \\ & \quad (\cos[\theta_4] \sin[\theta_1] + \cos[\theta_1] \cos[\theta_2] \sin[\theta_4]) \\ & \quad \sin[\theta_5])\}, \\ & \{0, 0, \cos[\theta_2], -(\cos[\theta_1] (d_2 \cos[\theta_1] + d_3 \sin[\theta_1] \\ & \quad \sin[\theta_2]) \\ & \quad \sin[\theta_2]) + (-(d_2 \sin[\theta_1]) + d_3 \cos[\theta_1] \sin[\theta_2]) \\ & \quad \sin[\theta_1] \sin[\theta_2], -((d_2 \cos[\theta_1] + d_3 \sin[\theta_1] \\ & \quad \sin[\theta_2]) \\ & \quad (\cos[\theta_1] \cos[\theta_2] \cos[\theta_4] - \sin[\theta_1] \sin[\theta_4])) + \\ & \quad (-(d_2 \sin[\theta_1]) + d_3 \cos[\theta_1] \sin[\theta_2]) \\ & \quad (\cos[\theta_2] \cos[\theta_4] \sin[\theta_1] + \cos[\theta_1] \sin[\theta_4]), \\ & \quad -((d_2 \cos[\theta_1] + d_3 \sin[\theta_1] \sin[\theta_2]) \\ & \quad (\cos[\theta_1] \cos[\theta_5] \sin[\theta_2] + \\ & \quad (\cos[\theta_4] \sin[\theta_1] + \cos[\theta_1] \cos[\theta_2] \sin[\theta_4]) \\ & \quad \sin[\theta_5])) + \\ & \quad (-(d_2 \sin[\theta_1]) + d_3 \cos[\theta_1] \sin[\theta_2]) \\ & \quad (\cos[\theta_5] \sin[\theta_1] \sin[\theta_2] + \\ & \quad (-\cos[\theta_1] \cos[\theta_4]) + \cos[\theta_2] \sin[\theta_1] \sin[\theta_4]) \\ & \quad \sin[\theta_5])\} \end{aligned}$$

Spatial Coordinate Measurement Using One Theodolite

Yan Jiahua and Oren Masory

Robotic Center
Florida Atlantic University
Boca Raton, FL 33431

Abstract

This paper describes two procedures by which the position of point in three dimensional space can be measured using only one theodolite. The procedures are based on two set of measurements taken from two arbitrary locations. The required hardware needed for the implementation of these methods are two standard measuring tapes installed vertically at two arbitrary locations or a standard bar mounted horizontally. Experimental results indicate that the position a point can be measured with accuracy better than 0.005 inch.

1. INTRODUCTION

Current industrial robots exhibit high positioning repeatability but poor positioning accuracy. While poor accuracy does not present a problem in applications where task geometry is being taught on-line by an operator, it is critical for applications programmed off-line. To overcome this problem, a procedure called Robot Calibration has been proposed as a solution [1]. Essential step in this procedure is to obtain accurate measurements of the manipulator pose.

Many different measurement schemes have been proposed, which include: vision systems [2], Coordinate Measuring Machines [3], Laser interferometer and laser tracking [4], and accurate fixtures [1].

The use of theodolite is widely used for spatial position measurement [5,6]. This paper describes the principle of two measurement

procedures, by which the spatial position of a point can be determined, using a single theodolite. The procedures are simple, and requires low cost supplementary hardware of one standard bar or two commercial measuring tapes.

2. USING VERTICAL SCALES METHOD

2.1 The Measurement Principle

Figure 1 illustrates the measurement setup: two standard scales S_1 and S_2 have been hung vertically in arbitrary locations within the field of view of the theodolite. The theodolite is positioned at T_1 and is leveled using its internal spirit level sensor defining horizontal plan π_1 . The scales S_1 and S_2 intersect π_1 plan at points A and B respectively and thus T_1A is perpendicular to S_1 and T_1B to S_2 . Then the theodolite is moved to a second location, T_2 , and is leveled again defining horizontal plan π_2 . The plan π_2 is parallel to π_1 and the distance between them, δh , can be read directly from the scales. The scales intersect π_2 plan at the points A' and B', and thus T_2A' is perpendicular to S_1 and T_2B' to S_2 . The points T_1 and T_2 are the projections of T_1 and T_2 on π_2 and π_1 respectively.

The distance between the two location of the theodolite projected on π_1 , T_1T_2 , is required for the computation procedure described later. With the theodolite in location T_1 , an arbitrary point, A'', on scale S_1 is selected. The distance T_1A can be obtained by:

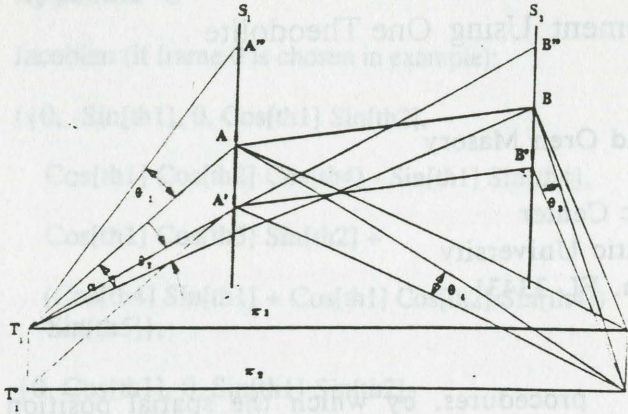


Figure 1: Measurement setup.

$$\overline{T_1A} = \frac{\overline{AA''}}{\tan \theta_1} \quad (1)$$

where $\overline{AA''}$ is measured from the scale S_1 and θ_1 is read from the theodolite. Similarly the distance $\overline{T_1B}$ can be found by selecting an arbitrary point, B'' , on scale S_2 :

$$\overline{T_1B} = \frac{\overline{BB''}}{\tan \theta_2} \quad (2)$$

This procedure is repeated at location T_2 yielding:

$$\overline{T_2A'} = \frac{\overline{AA'}}{\tan \theta_3} \quad (3)$$

$$\overline{T_2B'} = \frac{\overline{BB'}}{\tan \theta_4} \quad (4)$$

Note that the distances $\overline{AA''}$, $\overline{AA'}$, $\overline{BB''}$ and $\overline{BB'}$ are measured from the scales and the angles θ_1 , θ_2 , θ_3 and θ_4 are measured by the theodolite. From the triangle T_1AB one can obtain:

$$\overline{AB} = (\overline{T_1A}^2 + \overline{T_1B}^2 - 2\overline{T_1A} \cdot \overline{T_1B} \cos \alpha)^{1/2} \quad (5)$$

where the angle α is measured by the theodolite. From the triangle T_1AB one also can obtain:

$$\angle T_1AB = \cos^{-1} \left(\frac{\overline{T_1A}^2 + \overline{AB}^2 - \overline{T_1B}^2}{2\overline{T_1A} \cdot \overline{AB}} \right) \quad (6)$$

Since the plans π_2 and π_1 are parallel to each other and the scales S_1 and S_2 are perpendicular to these plans, the following holds: $T_2A = T_2A'$, $T_2B = T_2B'$ and $A'B' = AB$. The angles of the triangle T_2AB are given by:

$$\angle T_2AB = \cos^{-1} \left(\frac{\overline{T_2A'}^2 + \overline{AB}^2 - \overline{T_2B'}^2}{2\overline{T_2A'} \cdot \overline{AB}} \right) \quad (7)$$

$$\angle T_1AT_2 = \angle T_1AB - \angle T_2AB \quad (8)$$

In the triangle T_1AT_2 the length of T_1A , T_2A and the angle $\angle T_1AT_2$ are known, so the length of T_1T_2 can be found by:

$$\overline{T_1T_2} = (\overline{T_1A}^2 + \overline{T_2A}^2 - 2\overline{T_1A} \cdot \overline{T_2A} \cos \angle T_1AT_2)^{1/2}$$

2.2 Measurement Procedure

In the Figure 2 the spatial position of point P is to be measured. The projection of point P on π_1 is P' . With the theodolite at the position T_1 , the angles ψ_1 and ϕ_1 can be measured by the theodolite. Similarly the angles ψ_2 and ϕ_2 are measured with the theodolite in position T_2 . The angles ϕ_1 and ϕ_2 are given by:

$$\phi_1 = \angle AT_1T_2 - \phi_1 \quad (10)$$

$$\phi_2 = \angle BT_2T_1 - \phi_2 \quad (11)$$

where

$$\angle AT_1T_2 = \cos^{-1} \left(\frac{\overline{T_1A}^2 + \overline{T_1T_2}^2 - \overline{T_2A}^2}{2\overline{T_1A} \cdot \overline{T_1T_2}} \right) \quad (12)$$

and

$$\angle BT_2T_1 = \cos^{-1} \left(\frac{\overline{T_2B}^2 + \overline{T_1T_2}^2 - \overline{T_1B}^2}{2\overline{T_2B} \cdot \overline{T_1T_2}} \right) \quad (13)$$

In the triangle T_1T_2P' the length T_1T_2 and the angles φ_1 and φ_2 are known, so the following relations can be obtained:

$$\overline{T_1P'} = \overline{T_1T_2} \frac{\sin \varphi_2}{\sin(\varphi_1 + \varphi_2)} \quad (14)$$

From Figure 2 one also can get :

$$\overline{T_2P'} = \overline{T_1T_2} \frac{\sin \varphi_1}{\sin(\varphi_1 + \varphi_2)} \quad (15)$$

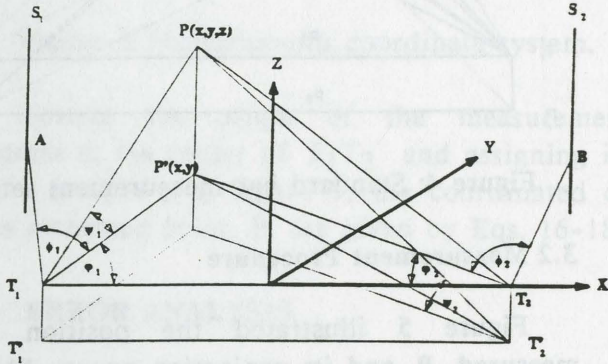


Figure 2: Coordinate system definition.

Setting the origin of the measurement coordinate system at the middle of the line T_1T_2 so that the X axis is along T_1T_2 the Y axis lay on plan π_1 and the Z axis is perpendicular to this plan. The coordinated of point P can be calculated by:

$$y = \overline{T_1P'} \sin \varphi_1 \quad (16)$$

$$x = \overline{T_1P'} \cos \varphi_1 - \frac{1}{2} \overline{T_1T_2} \quad (17)$$

$$z = \overline{T_1P'} \tan \psi_1 \quad (18)$$

3. HORIZONTAL BAR METHOD

3.1 The Measurement Principle

In this setup, a standard bar on which three positions, A B and C, along a straight line are accurately marked, is being used. The bar is installed horizontally (not necessarily with high accuracy) in the field of view of the theodolite. Figure 3 illustrates the top view of the set up where A, B and C are the marked points on the bar and T represents the location of the theodolite.

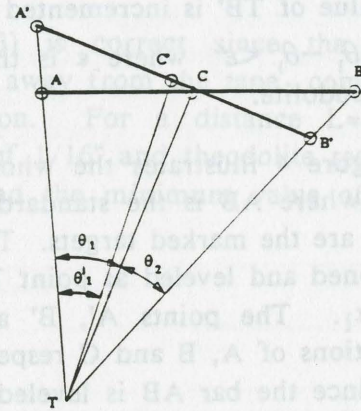


Figure 3: Definition for standard bar procedure.

The lengths AB and AC are known while the angles θ_1 and θ_2 are measured by the theodolite. The distance TB is estimated by $TB' < TB$, in this case, the point B moves to B'. Drawing a circle of radius AB, with its center at B', the line TA will be intersected at A'. Note that $A'B' = AB$ and $A'C' = AC$. From the triangle TA'B' we get:

$$\sin \alpha' = \frac{\overline{TB'} \sin(\theta_1 + \theta_2)}{\overline{AB}} \quad (19)$$

$$\angle \beta' = 180^\circ - (\theta_1 + \theta_2 + \alpha') \quad (20)$$

$$\overline{TA'} = \frac{\overline{AB} \sin \beta'}{\sin(\theta_1 + \theta_2)} \quad (21)$$

The angle θ_1' can be determined by:

$$\theta'_1 = \sin^{-1} \frac{\overline{A'C'} \sin \alpha'}{\overline{TC'}} \quad (22)$$

where TC' is given by:

$$\overline{TC'} = (\overline{TA'}^2 + \overline{A'C'}^2 - 2\overline{TA'} \cdot \overline{A'C'} \cos \alpha')^{1/2}$$

The angle θ'_1 is smaller than the angle θ_1 since we have assumed $TB' < TB$. At this point the value of TB' is incremented by a small step until $\theta_1 - \theta'_1 < \varepsilon$ where ε is the resolution of the theodolite.

Figure 4 illustrates the whole measurement setup where AB is the standard bar and A, B and C are the marked targets. The theodolite is positioned and leveled at point T_1 defining the plan π_1 . The points A', B' and C' are the projections of A, B and C respectively on plan π_1 . Since the bar AB is leveled, $A'B' = AB$ and $A'C' = AC$. The angles $A'T_1C'$ and $A'T_1B'$ are measured by the theodolite and therefore the distances T_1A' and T_1B' can be determined by the procedure above. Then, the theodolite is moved to a second location, T_2 , and the procedure is repeated finding T_2A'' and T_2B'' . Since the plans π_1 and π_2 are parallel, $T_2A' = T_2A''$ and $T_2B' = T_2B''$. From the triangles $T_1A'B'$, $T_2A'B'$ and T_1AT_2 one can determine the following:

$$\angle T_1A'B' = \cos^{-1} \frac{\overline{T_1A'}^2 + \overline{A'B'}^2 - \overline{T_1B'}^2}{2\overline{T_1A'} \cdot \overline{A'B'}} \quad (24)$$

$$\angle T_2A'B' = \cos^{-1} \frac{\overline{T_2A'}^2 + \overline{A'B'}^2 - \overline{T_2B'}^2}{2\overline{T_2A'} \cdot \overline{A'B'}} \quad (25)$$

$$\angle T_1AT_2 = \angle T_1A'B' - \angle T_2A'B' \quad (26)$$

$$\overline{T_1T_2} = \left[\overline{T_1A'}^2 + \overline{T_2A'}^2 - 2\overline{T_1A'} \cdot \overline{T_2A'} \cos(\angle T_1A'T_2) \right]^{1/2} \quad (27)$$

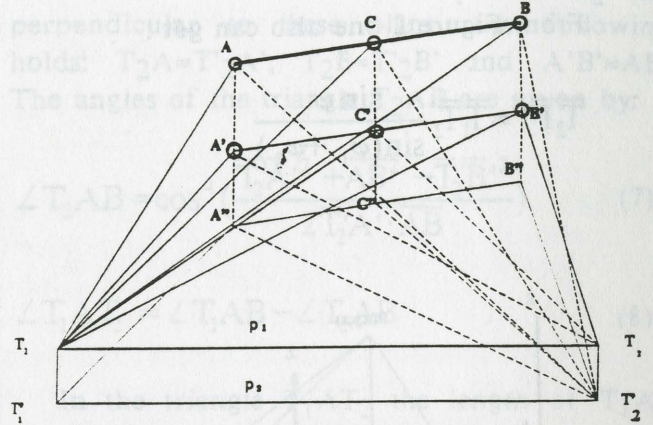


Figure 4: Standard bar measurement setup.

3.2 Measurement Procedure

Figure 5 illustrated the position being measured, P , and its projection on π_1 P' . The angles ϕ_1 and ψ_1 are measured by the theodolite from location T_1 while ϕ_2 and ψ_2 from location T_2 . The angles ϕ_1 and ϕ_2 are given by:

$$\phi_1 = \angle A'T_1T_2 - \phi \quad (27)$$

$$\phi_2 = \angle B'T_2T_1 - \phi_2 \quad (28)$$

where

$$\angle A'T_1T_2 = \cos^{-1} \frac{\overline{T_1A'}^2 + \overline{T_1T_2}^2 - \overline{A'T_2}^2}{2\overline{T_1A'} \cdot \overline{T_1T_2}} \quad (29)$$

$$\angle B'T_2T_1 = \cos^{-1} \frac{\overline{T_2B'}^2 + \overline{T_1T_2}^2 - \overline{T_1B'}^2}{2\overline{T_2B'} \cdot \overline{T_1T_2}} \quad (30)$$

The distance T_1P' can be found:

$$\overline{T_1 P'} = \overline{T_1 T_2} \frac{\sin \varphi_2}{\sin(\varphi_1 + \varphi_2)} \quad (31)$$

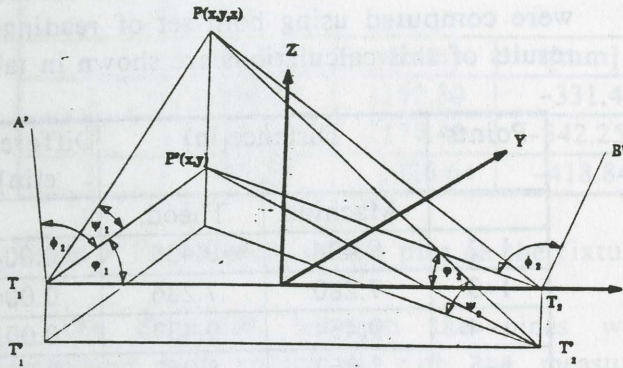


Figure 5: Measurements coordinate system.

Setting the origin of the measurement system at the center of $T_1 T_2$ and assigning its axes as shown in figure 5, the coordinates of the measured point, P, are given by Eqs. 16-18.

4. ERROR ANALYSIS

The analysis of errors was performed in a straight forward manner for the first procedure. Using a symbolic manipulation program (REDUCE) explicit expressions for the target coordinates x,y and z (given by Eqs. 16-18) as function of the measured variables and their corresponding errors, were obtained by successive substitution of Eqs. (1) through (15). Then the error along each axis was estimated by (demonstrated for the x coordinate):

$$\Delta X = \sum_{i=1}^n \left| \frac{\partial X}{\partial v_i} \Delta v_i \right| \quad (32)$$

where v_i is a measured variable and Δv_i is its corresponding measurement error.

The results of this analysis indicate that the accuracy of the measurements is highly sensitive to the resolution in which the data from the measuring tapes can be read. This result was expected since only the tapes provide length

measurements and these are in very low resolution.

To increase the accuracy of these readings, above the resolution of the measuring tapes, the theodolite was used to measure the angles α and β , shown in figure 3, and the distance was determined by:

$$\delta = (\text{Scale Resolution}) \frac{\alpha}{\alpha + \beta} \quad (33)$$

Eq. (33) is correct since the theodolite located far away from the tape compares with its resolution. For a distance $L=15\text{ft}$, tape resolution of $1/16''$ and theodolite resolution of 2 arc second the minimum value of δ is less than $0.002''$.

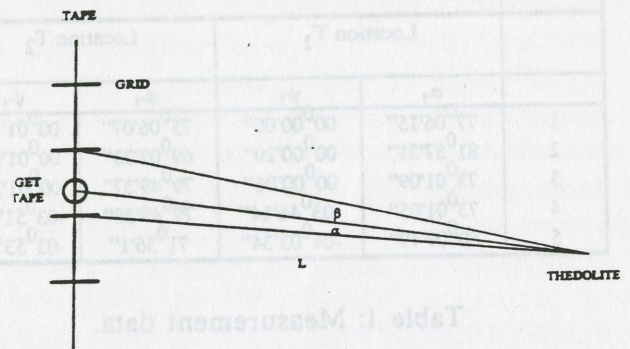


Figure 6: Measurement accuracy enhancement.

Using this technique, it is estimated by Eq. (32). that the errors will be in the order of few thousands of an inch which is accurate enough for measurement of relatively large structures. Similar results are expected for the second procedure as well.

5. EXAMPLES

In order to demonstrate the feasibility of the above methods two experiments were performed.

5.1 Experiment using two vertical tapes

A target was mounted on a CNC machine tool and moved to several known locations within the machine workspace. The position of the target at each location was measured by an electronic theodolite model Kern-E1, which has a resolution of 02". Two standard commercial measuring tapes with resolution of 1/16inch, were used in this experiment. Using the position information from the CNC controller, the distance between the locations were calculated. The same calculation was performed using the results obtained by the measurement procedure. Table 1 provides the measurement data taken during the process. The other measured angles are given by: $\theta_1=10^052'32''$, $\theta_2=10^058'34''$, $\theta_3=09^023'06''$, $\theta_4=12^034'08''$ and $\alpha=98^049'46''$.

Point	Measurements			
	Location T ₁		Location T ₂	
	ϕ_1	ψ_1	ϕ_2	ψ_2
1	77 ⁰ 06'15"	00 ⁰ 00'00"	75 ⁰ 06'07"	00 ⁰ 01'18"
2	81 ⁰ 57'31"	00 ⁰ 00'20"	69 ⁰ 02'33"	00 ⁰ 01'02"
3	73 ⁰ 01'09"	00 ⁰ 00'06"	79 ⁰ 49'37"	00 ⁰ 01'26"
4	73 ⁰ 01'05"	-03 ⁰ 46'14"	79 ⁰ 49'39"	-03 ⁰ 51'18"
5	80 ⁰ 09'49"	-04 ⁰ 03'34"	71 ⁰ 36'1"	-03 ⁰ 53'02"

Table 1: Measurement data.

Table 2 presents the coordinates of the target locations with respect to the machine tool coordinate system which readily available in the CNC controller with very high resolution (0.0001inch). The coordinates of the target with respect to the measurement coordinate systems were calculated by the above procedure.

Point	Machine			Theodolite		
	x[in]	y[in]	z[in]	x[in]	y[in]	z[in]
1	0	0	0	85.255	-1.581	0
2	8	5	0	80.492	-9.729	-0.008
3	-7	-2	0	87.047	5.481	0.003
4	-7	-2	-6	87.045	5.482	-5.998
5	5	2	-6	83.405	-6.640	-6.007

Table 2: Coordinates of the targets .

Since the relative position between the two coordinate system were not determined, in order to evaluate the quality of the measurements the distance between the locations were computed using both set of readings. The results of this calculations are shown in table 3.

Points	Distance[in]		Differenc e[in]
	Machine	Theod.	
1-2	9.434	9.438	0.004
1-3	7.280	7.286	0.006
1-4	9.434	9.437	0.003
1-5	8.062	8.062	0.000
2-3	16.553	16.561	0.008
2-4	17.607	17.612	0.005
2-5	7.348	7.350	0.002
3-4	6.000	6.001	0.001
3-5	14.000	14.011	0.011
4-5	12.649	12.657	0.008

Table 3: Distance between target locations.

5.2 Experiment using horizontal standard bar

A standard bar with three marks, made on CNC machine tool, was used in this experiment. The positions of three pins, mounted on a fixture, were accurately measured by a Coordinate Measuring Machine (CMM), and their relative distance was used to evaluate the measurement accuracy. The dimensions between the marks on the bar are: AB=630mm and AC=300mm. The theodolite measurement data are given in Table 4.

Point	Measurements			
	Location T ₁		Location T ₂	
	ϕ_1	ψ_1	ϕ_2	ψ_2
1	1 ⁰ 00'56"	-8 ⁰ 28'36"	6 ⁰ 41'04"	-8 ⁰ 00'02"
2	5 ⁰ 18'24"	-8 ⁰ 41'11"	2 ⁰ 55'26"	-8 ⁰ 30'51"
3	5 ⁰ 43'05"	-11 ⁰ 19'23"	4 ⁰ 38'00"	-11 ⁰ 01'30"

Table 4: Measurement data.

The coordinates of the pins on the fixture were calculated according to the above procedure and are given in Table 5.

Point	Target's coordinates		
	x[mm]	y[mm]	z[mm]
1	-234.04	2192.80	-331.47
2	-67.14	2174.43	-342.25
3	-88.29	2026.64	-418.84

Table 5: Coordinates of the pins on the fixture.

The distances between the pines were calculated and compared with the measured ones, as shown in Table 6.

Points	Distance[mm]		Differ. [mm]
	CMM	Theodolite	
1-2	168.28	168.25	-0.03
1-3	237.84	237.67	-0.17
2-3	168.00	167.80	-0.20

Table 6: Distance between the pins.

6. CONCLUSIONS

Two procedures for the measurement of spatial position of a point, using only one theodolite, have been presented. Both are based on two sets of measurement taken from two locations. A simple devices are used to provide distance reference. It has been shown that measurement accuracy of few thousands of an inch can be achieved.

7. REFERENCES

1. Mooring B.W, Roth Z.S., Driels M.R, Fundamentals of Manipulator Calibration, John Wiley & Sons, 1991.
2. Jilain L., "The 3D space Reconstruction of Intewrsecting Cameras", J. Beijing Inst. of Tech., Vol. 10, No. 1, 1990.
3. Mooring B.W., Padavala S.S., "The Effect of Model Complexity on Robot Accuracy,"

Proc. IEEE Conf. on Robotics and Automation, 1989.

4. Lau K., Hocken R., Haynes L., "Robot Performance Measurements Using Automatic Laser Tracking Techniques", Robotics and Computer-Integrated Manufacturing, Vol. 2, No. 3, 1985.
5. Whitney D.E., Lozinski C.A., Rourke J.M., "Industrial Robot Forward Calibration Method and Results", ASME J. of Dynamic Systems, Measurement, and Control, Vol. 108, 1986.
6. Wild/Leitz RMS 2000 Remote Measuring System, Wild Heerburg Ltd. & Ernst Leitz Wetzlar Ltd.

KINEMATIC CALIBRATION AND COMPENSATION OF A STEWART PLATFORM

Oren Masory, Jian Wang and Hanqi Zhuang

Robotics Center
Florida Atlantic University
Boca Raton, FL 33431

ABSTRACT

Manufacturing tolerances, installation errors and link offsets contaminate the nominal values of the kinematic parameters used in the kinematic model of the platform. Since the platform controller determines the length of the actuators according to the nominal model, the resulting pose of the platform is inaccurate. To enhance the positioning accuracy of the platform, there is a need to determine the actual values of these parameters and how to incorporate them into the inverse kinematic procedure performed by the controller. This paper presents and demonstrates by simulation a method by which these parameters can be identified using pose measurements.

1. INTRODUCTION

The Stewart platform, proposed by D. Stewart as an aircraft simulator [1], is a six degree-of-freedom parallel manipulator where the end-effector is attached to a moveable plate supported in-parallel by six linear actuated links. Recently, this kind of parallel manipulator has attracted considerable interest due to their inherent characteristics, compared with serial manipulators, which include: 1) High force/torque capacity since the load is distributed to several in-parallel actuators; 2) High structural rigidity; and 3) It is claimed that their accuracy is better due to the noncumulative joint errors. This interest resulted in a large number of reports dealing with topics such as: kinematics analysis [1-5]; workspace analysis [6-8]; practical

design/construction considerations [2]; dynamics [9-10]; and numerous applications [11-15].

In a previous paper [16] it was shown that the accuracy of a Stewart platform is of the same order as of a serial manipulator with relatively the same nominal dimensions. The degradation in accuracy is mainly due to manufacturing tolerances, used to construct the platform, manifested as deviations between the nominal kinematic parameters of the platform model and the actual parameters. Since the platform's controller determines the length of the actuators according to the nominal model, the resulted pose of the platform is inaccurate. One way to enhance platform accuracy is by kinematic calibration, a process by which the actual kinematic parameters are identified and then used to modify the kinematic model used by the controller. Thus, the controller will use a more accurate model and as a result the accuracy of the platform will be improved.

While there are numerous reports dealing with the calibration issues of serial robot manipulators, little attention has been given to parallel manipulators. In [17] a calibration method based on the following idea was proposed: by fixing one link at a time and moving the other five links during measurement process, one is able to compute kinematic parameters one link at a time. The approach requires relatively little computation. However, the identified parameters may not be optimal because: 1) The measurable workspace is small because of the restricted motion pattern of the robot; 2) Kinematic parameters of individual links are computed separately,

therefore, the coupling effects among the legs are not fully explored; and 3) The model used did not include all the kinematic parameters of the platform (it was assumed that the U-joints and the ball-joints used to connect the links between the base and the platform were perfect).

This paper presents an error-model based approach for kinematic calibration of a Stewart platform which is directly adopted from serial manipulator calibration methodology. As a result, the procedure and the algorithm are easily understood and implemented. Similar to the serial manipulator calibration process, the calibration process of the Stewart platform consists of four steps: 1) Construction of the platform kinematic error model; 2) Measurement of the platform pose with respect to a reference frame as well as its joint variables (link lengths for Stewart platform); 3) Identification of the unknown kinematic parameters in the model; and 4) Compensation of the pose error based on the identified model.

This paper focuses on the error-model construction which is the central issue in adopting this method. Due to the inherent difficulty of analytical solution of the forward kinematics of the Stewart platform, it is inevitable that numerical methods be used to compute the identification Jacobian matrix that relates the robot pose errors to the kinematic parameter errors. The computation algorithm of the identification Jacobian matrix, as well as the a compensation scheme are discussed in detail. Simulated measurements, which include realistic noise, are used to evaluate the effectiveness of the proposed calibration approach.

2. KINEMATIC MODELING OF A STEWART PLATFORM

The Stewart platform, illustrated in figure 1, is composed of six variable length links, a fixed base and a movable plate to which the tool is attached. A base coordinate {B} is

placed at the base center O_b with its Z axis perpendicular to the base plane. Similarly, the coordinate frame {P} is located at the center of the moving plate. The links are connected to the base by U-joints and to the plate by ball joints. These links can be extended or contracted by actuators.

Two different models exist: 1) The nominal model which neglects manufacturing and installation errors and therefore joints are treated as points and the links are treated as lines; and 2) The accurate model which take into consideration the above errors. For both cases the forward and inverse kinematic problems, defined below, must be solved:

1. Forward kinematics: Given the values of the six link lengths and the kinematic parameters, determine the platform pose.
2. Inverse kinematics: Given the platform pose and the kinematic parameters, determine the link lengths.

For the nominal model, the coordinates of B_i , with respect to {B} (denoted as b_i) represent the ball joint centers and the coordinates of P_i with respect to {P}, denoted as p_i , represent the centers of the U-joints. The length of each link, given by the distance between B_i to P_i (denoted as d_i) is the joint variable. There are forty two kinematic parameters in this model: twelve position vectors (b_i and p_i) and six link lengths. The solution for the inverse kinematic problem, in this case, is simple:

$$d_i = |R p_i + q - b_i| \quad i=1,2,\dots,6 \quad (1)$$

where R and q define the orientation and translation of {P} with respect to {B}.

However, even for this simple model there is no closed form solution for the forward kinematic problem and only a few numerical approaches have been proposed [4,5].

The accurate model which accommodates all manufacturing and installation errors has been presented in previous paper [16]. The model consists of six joint-link trains, each treated as a serial manipulator of 2R-P-3R

type modeled by eight consecutive transformations. Therefore, twenty two independent kinematic parameters are needed to incorporate manufacturing and installation errors in each joint-link train kinematic model. Thus, all together one hundred and thirty two kinematic parameters are needed to accurately model the kinematics of a Stewart platform.

3. PARAMETER IDENTIFICATION

3.1 Problem Statement

The kinematic parameter identification problem can be stated as follows: Given the measured link lengths and measured platform poses (for j different platform poses), estimate the kinematic parameters which define the transformation between the two. The minimum number of measured poses, j_{min} , depends on the number of parameters needed to be identified. Since measurement noise exists, the parameters are identified using least square techniques and therefore j should be larger than j_{min} .

3.2 Error-model Based Approach

The error-model based identification method is widely applied for serial robot manipulator calibration [18-21]. The calibration procedure, shown in Figure 2, follows these steps:

1. The robot pose vector is expressed as a function of the kinematic parameter vector, \mathbf{u} , and the given joint variable vector \mathbf{v} :

$$\mathbf{x} = \mathbf{f}(\mathbf{u}, \mathbf{v}) \quad (2)$$

where the function \mathbf{f} represents the forward kinematics of the robots.

Differentiating Eq. (2) with respect to kinematic parameters yields the identification Jacobian \mathbf{J} :

$$\mathbf{J} = d\mathbf{x}/d\mathbf{u} = d\mathbf{f}(\mathbf{u}, \mathbf{v})/d\mathbf{u} \quad (3)$$

For serial manipulators, the forward kinematics is easily obtained and can be expressed in closed form and the identification Jacobian, \mathbf{J} , can be expressed analytically [19].

2. The pose, \mathbf{X}_k is calculated by $\mathbf{f}(\mathbf{u}_k, \mathbf{v}_j^m)$, where \mathbf{f} represents the forward kinematics of the serial manipulator, \mathbf{v}_j^m is the measured joint variables at pose j , and \mathbf{u}_k is the kinematic parameters vector (k is iteration index).
3. A pose error, $d\mathbf{x}$, defined as the difference between the measured pose, \mathbf{x}^m , and \mathbf{X}_k .
4. The Jacobian $\mathbf{J}(\mathbf{u}_k, \mathbf{v}_j^m)$ is calculated.
5. The kinematic parameters vector is updated by: $\mathbf{u}_{k+1} = \mathbf{u}_k + \Delta\mathbf{u}$ where $\Delta\mathbf{u} = \mathbf{J}^{-1}(\mathbf{u}_k, \mathbf{v}_j^m) d\mathbf{x}$ where \mathbf{J}^{-1} is the pseudo inverse of \mathbf{J} .
6. The procedure defined by steps 2 through 5 is repeated until $d\mathbf{x}$ is small enough then the last value of \mathbf{u} is considered to represent the actual values of the kinematic parameters.

Similar procedure, as shown in figure 3, is used to solve the parameter identification problem of the Stewart platform. The identification procedure is given as follows:

1. Obtain measurements of the platform poses \mathbf{X}_j^m (j is pose index, $j=1,2,\dots,s$) and the corresponding link lengths d_j^m .
2. Let the initial parameter vector $\mathbf{u}_k|_{k=0} = \mathbf{u}^n$, where \mathbf{u}^n is the known nominal kinematic parameters vector and k is an iteration index.
3. Compute the platform pose, \mathbf{X}_{jk} , using d_j and \mathbf{u}_k by solving the forward kinematic problem of the platform.
4. Calculate the pose error vectors $\delta\mathbf{x}_k$ (a 6×1 vector) from \mathbf{X}_j^m and \mathbf{X}_{jk} .
5. Use the norm of $\delta\mathbf{x}_k$ as a termination condition: If $|\delta\mathbf{x}_k| < \epsilon$, then the identified parameter $\mathbf{u}^a = \mathbf{u}_k$ and stop the program. Otherwise continue to step 6.
6. Compute the identification Jacobian \mathbf{J}_k .
7. Determine a correction vector for \mathbf{u}_k , $\delta\mathbf{u}_k = [(\mathbf{J}_k^T \mathbf{J}_k)^{-1} \mathbf{J}_k^T] \delta\mathbf{x}_k$.
8. Update \mathbf{u}_k by $\mathbf{u}_{k+1} = \mathbf{u}_k + \delta\mathbf{u}_k$ and go back to step 3.

The algorithm and procedure for the forward kinematic solution (step 3) is presented in [16]. The computation of δx_k (step 4) and the formulation of the identification Jacobian matrix (step 6) are discussed in the following section where the iteration index k is omitted for clarity.

3.3 Jacobian Formulation

The major issue in parameter identification by the error-model based approach is the formulation of the identification Jacobian, J , which relates a differential pose vector, δx , to a differential change of the kinematic parameters vector, δu :

$$\delta x = J \delta u \quad (4)$$

where δx is a 6×1 vector which can be expressed by:

$$\delta x = [\delta x_1^T, \delta x_2^T, \dots, \delta x_s^T]^T \quad (5)$$

In Eq (5), δx_j is a 6×1 error vector given by:

$$\delta x_j = (\delta r_j^T, \delta p_j^T)^T \quad j=1,2,\dots,s \quad (6)$$

where: δr_j is a 3×1 orientation error vector.

δp_j is a 3×1 position error vector.

The pose error vector δx_j can be obtained by the error matrix ΔX_j which is given by

$$\Delta X_j = X_j^{-1} (X_j^m - X_j) \quad (7)$$

where X_j is an 4×4 homogeneous pose transformation matrix and X_j^m is the measured platform pose transformation matrix.

Let $\Delta X_j = \{\Delta R_j, \Delta p_j\}$ where ΔR_j is an anti-skew 3×3 orientation error matrix and Δp_j is a 3×1 translation error vector. Then $\delta p_j = \Delta p_j$, and δr_j can be computed from ΔR_j

$$\delta r_j S = \Delta R_j \quad (8)$$

where S is a cross operator defined by [19]

$$S = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \quad (9)$$

The Jacobian is composed of s sub-Jacobian matrices J_1, \dots, J_s :

$$J = [J_1, J_2, \dots, J_s]^T \quad (10)$$

For each measurement j , J_j is defined as

$$\delta x_j = J_j \delta u \quad (j=1,2,\dots,s) \quad (11)$$

where J_j is a 6 by m submatrix where m is the number of parameters needed to be identified and is obtained by:

$$J_j = [j_{1j}, j_{2j}, \dots, j_{mj}] \quad (12)$$

where j_{ij} , $i=1,2,\dots,m$, is a 6×1 vector represents the partial derivative of the pose vector x_j with respect to the i th parameter u_i . It can be obtained numerically by perturbation method as follows:

$$j_{ij} = \delta x_{ij} / \delta u_i \quad (13)$$

where δu_i is an appropriate small change in the parameter u_i . The 6×1 error vector δx_{ij} is obtained from the transformation matrices $X_j(u)$ and $X_j(u+\delta u_i)$, just as the δx_j is obtained from $X_j(u)$ and X_j^m . Noting that δu_i is a $m \times 1$ vector whose elements are all zeros except the i th element which has the value of δu_i ($\delta u_i = [0, \dots, 0, \delta u_i, 0, \dots, 0]^T$). In other words, the perturbation matrix $X_j(u+\delta u_i)$ is the forward kinematics solution using the parameter vector with its i th element perturbed. In summary, the Jacobian can be expressed as following $6 \times m$ matrix:

$$J = \begin{bmatrix} \frac{\delta x_{11}}{\delta u_1} & \frac{\delta x_{21}}{\delta u_2} & \dots & \frac{\delta x_{m1}}{\delta u_m} \\ \frac{\delta x_{12}}{\delta u_1} & \frac{\delta x_{22}}{\delta u_2} & \dots & \frac{\delta x_{m2}}{\delta u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta x_{1s}}{\delta u_1} & \frac{\delta x_{2s}}{\delta u_2} & \dots & \frac{\delta x_{ms}}{\delta u_m} \end{bmatrix} \quad (14)$$

It can be seen from Eqs. (10) to (14) that the computation of the identification Jacobian involves a great number of platform forward kinematics solutions for the nominal and the accurate models. Both are solved numerically, and therefore the parameter identification procedure is computationally expensive.

4. COMPENSATION

The compensation problem of the Stewart platform can be stated as follows: Given the required platform pose, x^a , and the identified kinematic parameters, u^a , find the corresponding links length vector d .

The solution is obtained by solving the inverse kinematic problem of each joint-link train using Newton-Raphson method [16]. This problem is similar to the problem of solving the inverse kinematics problem of a six degree freedom, 2R-P-3R, serial manipulator. Denote the vector v_j as joint variable vector of the j^{th} joint-link train, the problem can be stated as follows: Given the desired platform pose x^a , the joint-link train forward kinematics model $x_j = f_j(v_j, u^a)$ $j=1,2,\dots,6$ and its nominal inverse kinematics model $v_j = g_j(x^a, u^a)$, find v_j^a such that $x^a = f_j(v_j^a, u^a)$.

As an initial guess for v_j , the nominal inverse kinematics of the platform can be used. Fortunately, for a practical Stewart platform, the nominal value for v_j is unique and can be easily found [16].

The problem is solved, for each joint-link train separately, as follows (the subscription j

is omitted for clarity and k is the iteration index):

1. Make an initial guess of v , using the nominal model of the joint-link train $v = g(x^a, u^a)$.
2. Compute an estimated platform pose $x_k = f(v_k, u^a)$.
3. Obtain a pose error vector Dx_k from x_k and x^a .
4. Compute a correction term $Dv_k = J^{-1} Dx_k$, where J^{-1} is pseudo inverse of the joint-link train Jacobian formulated by the nominal joint-link train model.
5. Update the estimated v_k by $v_{k+1} = v_k + Dv_k$.
6. Check whether the termination condition, $|Dx_k| < \epsilon$, is satisfied for all joint-link trains. If so, then $v^a = v_k$ and the link length vector, d , is retrieved from v^a . Otherwise go back to step 2.

For the reduced model, where only errors in joint locations and link offsets are considered, the compensation is straight forward. The inverse kinematic problem is solved by Eq. (1) where the identified values of b and p are used. Then the identified link offsets are added to resulted link lengths, d . These values are used to drive the platform to the required position.

5. SIMULATION STUDY

A simulation study was performed in order to verify the above algorithm. In this study a platform with the following dimensions: $R_b = 3\text{ft}$, $R_p = 1\text{ft}$; and links with length ranges between 4.5ft to 7.5ft, was used. The following uniformly distributed random errors were imposed on the nominal parameters:

1. Position error of the U-joints at the base:
 $|Db_x| < 0.24$ inch, $|Db_y| < 0.24$ inch, $|Db_z| < 0.12$ inch.
2. Position error of ball joints on the moving plate:
 $|Dp_x| < 0.012$ inch, $|Dp_y| < 0.012$ inch, $|Dp_z| < 0.012$ inch.
3. Link offset: $|Dd| < 0.024$ inch.

4. Manufacturing tolerance for both U-joints and ball-joints (D-H parameters): $|Dd| < 0.004$ inch, $|Da| < 0.004$ inch and $|Da| < 0.1$ □

Three different cases were investigated:

Case 1: The results of the previous paper indicate that the contribution of the joints manufacturing tolerances have a minor effect on the platform pose error (item 4 in the above list). Therefore, a reduced model in which these parameters were not considered can be used. In this model, the number of parameters that have to be identified is only 42 compared with 132 parameters in the full model. In this case 8 pose measurements were used (one more than the minimum required) and it was assumed that there is no measurement noise (the measurement of the platform pose is accurate). Figure 4 shows the convergence of the pose error norm as function of the number of iteration of the identification procedure. The pose error vector was divided into translation and orientation errors and their norms are shown in the figure. As shown, initially the translation error is about 0.8 inch and the orientation error is about 8 □. After only 4 iteration both error were practically reduced to zero.

Case 2: This case is the same as the previous one but a uniformly distributed random noise was added to the pose measurement in order to simulate measurement errors. The magnitudes of the measurement errors used in the simulation were adopted from experimental results obtained by using a theodolite as a measuring device. These errors are: 1) Pose position measurement error of ± 0.0024 inch; and 2) Pose orientation error of ± 0.0285 degree. Since the measurements are contaminated by noise, more pose measurements are required in order to improve the identification results from 'Least Square' point of view. As before, the parameters error vector has 42 elements: 24 elements due to errors in the location of the U-joint at the

base; 24 elements due to errors in the location of the Ball-joint at the plate; and 6 elements describing the link offsets. Figure 5 illustrates the convergence of the norm of the three sets of parameter errors as function of the number of measured poses. As expected, better identification results are obtained as the number of measured poses increases. However, very little improvement can be achieved once the number of measured poses exceeds a certain limit (about 20 in this case). To be more specific, Figure 6 provides the initial and the final (after identification) values of the parameter errors for the case of 28 measured poses and 4 iterations of the identification algorithm. In this figure the first 18 parameters correspond to errors on the base, the next 18 to errors on the plate and the last 6 to link offsets. The initial values are given by the difference between the actual and the nominal values while the final values are given by the difference between the actual and the identified values. As shown the large errors, mainly due errors in the location of the U-joint on the base, were reduced substantially, while small initial errors were slightly reduced or even increased due to the least square averaging effect. It should be emphasized that the errors can not be eliminated, even with large number of measurements, due to the existing measurement errors.

Case 3: In this case the full model, which includes 132 parameters, was simulated and used to identify the above 42 parameters, assuming that the other parameters are accurate. This simulation was performed in order to verify the effectiveness of the algorithm and therefore no measurement noise was included and only 8 measured poses was used. Figure 7 shows that the parameters converges to its true value after 4 iterations and figure 8 shows that as a result the pose error is essentially eliminated.

6. CONCLUSIONS

An effective algorithm for the identification of the kinematic parameters of a Stewart platform has been presented and verified through simulations. The algorithm can be applied to both the reduced and the full models which differ by the number of parameters needed to be identified. An compensation procedures, for the full and reduced models, are also presented. The algorithm was tested using simulated measurements which include realistic measurement noise and as a result, the platform pose error was reduced by at least one order of magnitude.

The proposed identification algorithm is computationally expensive and has to be performed on a powerful computer for practical implementation. If m parameters have to be identified, and s measured poses are used, the forward kinematics problem of the accurate model, which is also solved iteratively by numerical methods, has to be solved $s(m+1)$ times for each iteration of the identification algorithm. For example, if only 42 parameters are being identified using 8 measured poses, 344 forward kinematics solutions have to be obtained. For this case, about 11 hours of SUN 3/260 CPU time are needed for each iteration. However, since this procedure has to be performed once (or periodically) the computation cost is not a major concern. On the other hand, the compensation algorithm has to be executed in real time and therefore the use of the full model will require the use of a powerful computer as part of the platform controller. To eliminate this need it is recommended to use the reduced model for compensation.

7. ACKNOWLEDGMENTS

This work was supported by grants from the Technological Research and Development Authority of Florida. The authors wish to

thank Dr. Jim Pohl for his helpful comments in writing this manuscript.

8. REFERENCE

1. Stewart, D., "A Platform With Six Degree of Freedom," *Proceedings of the Institution of Mechanical Engineering*, Vol.180, Part 1, No. 15, 1965-66, pp.371-386
2. Fichter, E.F., "A Stewart Platform-Based Manipulator: General Theory and Practical Consideration," *Journal of Robotic Research*, Summer 1986, pp.157-182
3. Hunt, K.H., "Structural Kinematic of In-parallel-actuated Robot-arms," *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol.105, Dec.1983, pp.705-711
4. Nanua, P., Waldron, K.J. and Murthy, V., "Direct Kinematic Solution of a Stewart Platform," *IEEE Transaction on Robotics and Automation*, Vol.6, No.4, Aug. 1990, pp.438-443
5. Huang, M. Z., "On the Kinematics of Parallel-chain Platform Manipulator," *Proc. The Forth Annual Conference on Recent Advance in Robotics*, Boca Raton, Florida, May 1991, pp. 251-256
6. Nguyen, C. C. and Pooran, F.J., "Kinematic Analysis and Workspace Determination of a 6 DOF CKCM Robot End-effector," Elsevier Science Publishers B.V. 0378-3804/89, 1989, pp. 283-294
7. Gosselin, C., "Determination of the Workspace of 6-DOF Parallel Manipulators," *ASME Journal of Mechanical Design*, September 1990, Vol.112, pp. 331-336
8. Masory, O., Wang, J., "Workspace Evaluation of Stewart Platform," *Proc. ASME Design Technical Conf.*, Scottsdale, AZ, Sept. 13-16, 1992.
9. Lee, J. et al, "Computer Simulation of a Parallel Link Manipulator," *J. Robotics & Computer Integrated Manufacturing*, Vol. 5, No. 4, 1989, pp.333-342
10. Sugimoto, K., "Kinematic and Dynamics Analysis of Parallel Manipulators by Means

of Motor Algebra," *ASME J. of Mechanisms, Transmissions, and Automation in Design*, Vol. 109, pp. 3-7, 1987

21. Mooring, B.W., Roth, Z. S. and Driels M., *The Fundamentals of Robot Calibration*, John Wiley & Sons, 1991

11. Hoffman, R. and Hoffman, M.C., "Vibration Models of an Aircraft Simulation Motion Systems," in *Proc. 5th World Congress for the Theory of Machines and Mechanisms*, 1979 pp. 603-60618

12. McCallion, H., Johnson, G.R. and Phan, D.T., "A Compliance Device for Inserting a Peg Into a Hole," *The Industrial Robot*, June 1979

13. Fichter, E.F and McDowell, E.D., "A Novel Design For a Robot Arm," *Proc. Int. Computer Tech. Conf.*, San Francisco, California, Aug. 1980, pp. 250-256

14. Landsburger, S.E. and Sheridan, T.B., "A New Design For Parallel Link Manipulators," *IEEE Proc. System Man and Cybernetics Conf.*, Tuscon, Arizona, Nov. 1985, pp. 812-814

15. Sheridan, T.B., "Human Supervisory Control of Robot Systems," *Proc. Int. Conf. on Robotics and Automation*, San Francisco, California, Apr. 1986, pp. 802-812

16. Wang J. Masory O., "On the Accuracy of a Stewart Platform - Part I, The Effect of Manufacturing Tolerances", Submitted IEEE Conf. on Robotics and Automation, 1993.

17. Zhuang, H. and Roth, Z., "A Method for Kinematic Calibration of Stewart Platforms," ASME Annual winter meeting, Atlanta, GA, 1991, pp43-48

18. Roth, Z. S., Mooring, B. W. and Ravani, B., "An Overview of Robot Calibration," *IEEE J. Robotics and Automation*, Vol. 3, No.5, Oct. 1987, pp. 377-385

19. Wu, C.H., "A Kinematic CAD Tool For the Design and Control of a Robot Manipulator," *Int. J. Robotics Research*, Vol. 3, No.1, 1984, pp. 58-67

20. Hayati S., Tso, K., and Roston G., "Robot Geometry Calibration," *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol.2, 1988, pp. 947-951

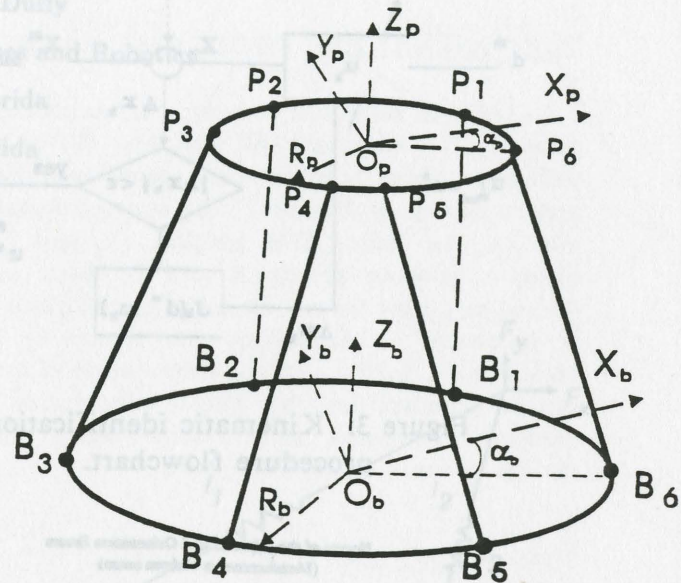


Figure 1: Stewart platform definitions.

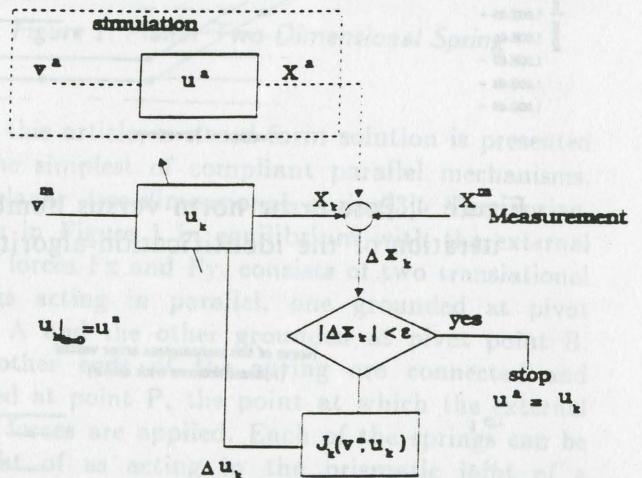


Figure 2: Flowchart of kinematic identification procedure for serial manipulators.

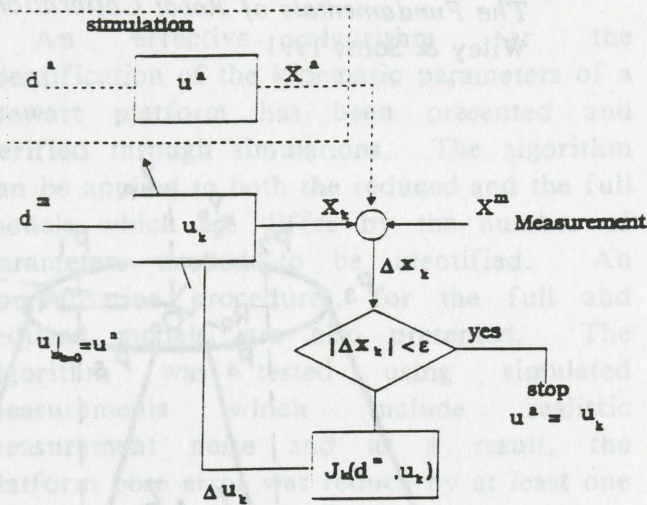


Figure 3: Kinematic identification procedure flowchart.

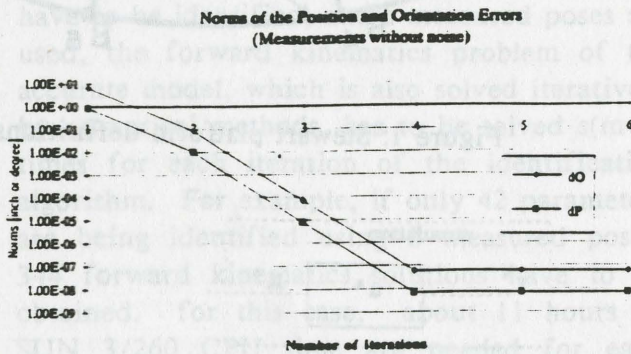


Figure 4: Pose error norm versus number of iteration of the identification algorithm.

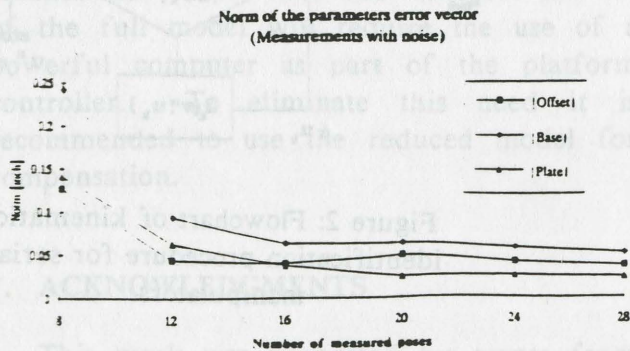


Figure 5: Parameters error vector norm versus number of measured poses.

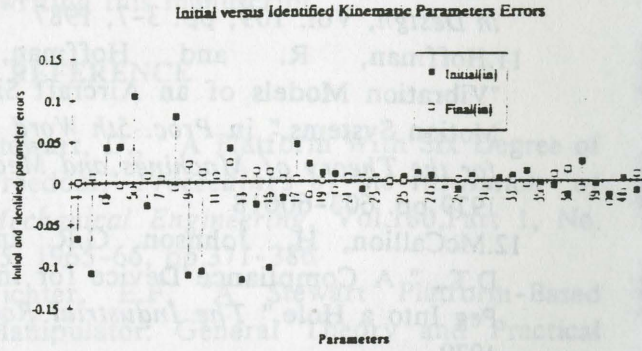


Figure 6: Initial versus final values of the kinematic parameters errors.

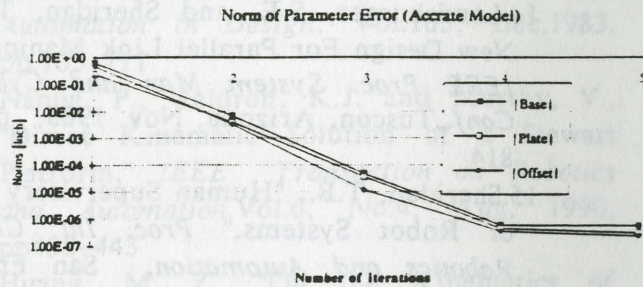


Figure 7: Parameters error vector norm versus number of measured poses.

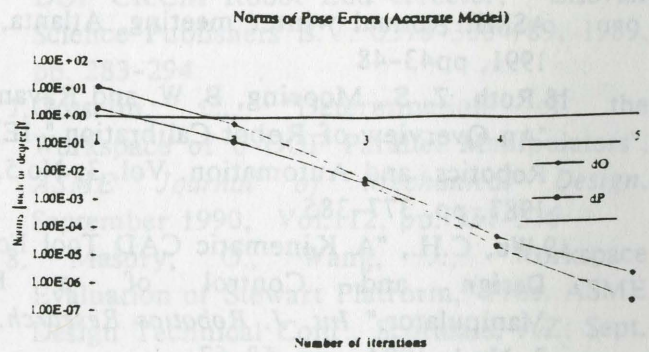


Figure 8: Pose error norm versus iteration number of the identification algorithm.

A REVERSE FORCE ANALYSIS OF A PLANAR TWO-SPRING SYSTEM

T. Pigoski and J. Duffy

Center for Intelligent Machines and Robotics

University of Florida

Gainesville, Florida

ABSTRACT

A closed-form reverse force analysis is performed on a planar two-spring system. The two springs are grounded to pivots at one end and attached to a common pivot at the other. A known force is applied to the common pivot of the system and resultant assembly configurations are determined. A sixth degree polynomial in the resultant length of one spring is derived. From this, six unique locations of the point of application of force are obtained. The results are verified numerically by performing a forward force analysis and displaying real solutions. It is clear that there are a maximum of six mathematical assembly configurations to obtain the desired force for such a spring system.

INTRODUCTION

Compliant parallel mechanisms have traditionally played a significant role in areas of force application such as mounting and suspension systems, and over the past few years, they have had an increasing impact in areas of robotic force control and large deformation, nonlinear finite element analysis. However, to this author's knowledge very little work has been done on closed-form, reverse force analysis of such mechanisms. Most of the literature has devoted itself to forward force analysis which is inherently simple (viz. it is required to compute the resultant forces given the spring leg lengths). Additionally, Vanderplaats[1] has investigated a nonlinear, iterative reverse analysis, whereby the displacement field of the structure under a given loading is found by minimizing the total potential energy of the system. Numerically, this method produces a single solution to the problem. Also, Seirig[2] has used an algorithmic approximation to obtain a single geometric solution for systems under small loads and undergoing small displacements from the unloaded configuration.

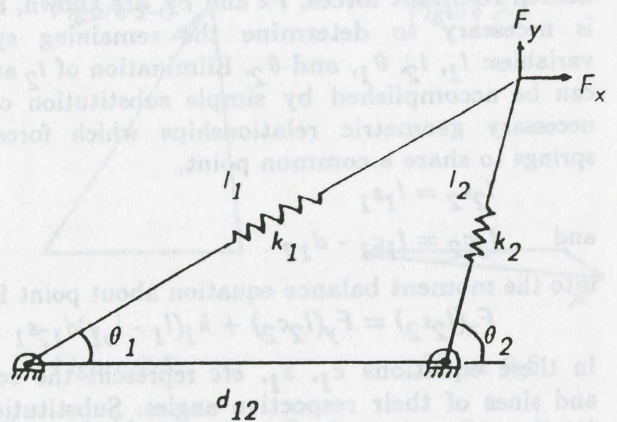


Figure 1: Planar Two-Dimensional Spring

In this article, a closed-form solution is presented for the simplest of compliant parallel mechanisms, the planar two-dimensional spring[3]. The spring, shown in Figure 1 in equilibrium with the external static forces F_x and F_y , consists of two translational springs acting in parallel, one grounded at pivot point A and the other grounded at pivot point B. The other ends of the spring are connected and pivoted at point P, the point at which the external static forces are applied. Each of the springs can be thought of as acting in the prismatic joint of a revolute-prismatic-revolute serial chain. For this mechanism, it is assumed that the free lengths and elasticity constants of the springs are known along with the distance between pivot points A and B.

The significance of this article is that a closed-form solution not only produces the exact number of assembly configurations for a desired resultant load, but also provides a simple tool for analyzing the motion of the pivot points for all six roots as the desired resultant load changes, neither of which has been done before. It is important to note that these solutions are derived mathematically and may

not always be realizable as practical assembly configurations. In many cases, negative or complex resultant spring lengths appear. Currently, these solutions are not physically realizable, but here they have been included for the purpose of discussion.

THE GENERAL 6TH DEGREE POLYNOMIAL

In order to solve the reverse analysis problem it is necessary to eliminate all but one of the system unknowns from the system equations. For the system shown in Figure 1, the spring constants, k_1 and k_2 , the spring free lengths, l_{o1} and l_{o2} , the distance between pivots A and B, d_{12} , and the desired resultant forces, F_x and F_y , are known, and it is necessary to determine the remaining system variables: l_1 , l_2 , θ_1 , and θ_2 . Elimination of l_2 and θ_2 can be accomplished by simple substitution of the necessary geometric relationships which force the springs to share a common point,

$$l_2 s_2 = l_1 s_1 \quad (1)$$

$$\text{and } l_2 c_2 = l_1 c_1 - d_{12} \quad (2)$$

into the moment balance equation about point B:

$$F_x(l_2 s_2) = F_y(l_2 c_2) + k_1(l_1 - l_{o1})d_{12}s_1 \quad (3)$$

In these equations c_1 , s_1 , etc represent the cosines and sines of their respective angles. Substitution of (1) and (2) into (3) produces the equation:

$$A_1 s_1 + B_1 c_1 = D_1 \quad (4)$$

where

$$A_1 = k_1(l_1 - l_{o1})d_{12} - F_x l_1$$

$$B_1 = F_y l_1$$

$$D_1 = F_1 d_{12}$$

Another equation of the form of (4) can be obtained by algebraic manipulation of the force balance equations:

$$F_x = k_1(l_1 - l_{o1})c_1 + k_2(l_2 - l_{o2})c_2 \quad (5)$$

$$F_y = k_1(l_1 - l_{o1})s_1 + k_2(l_2 - l_{o2})s_2 \quad (6)$$

Rearranging and substituting (1) and (2) produce:

$$F_x - F_c1 + k_2 d_{12} = -k_2 l_{o2} c_2 \quad (7)$$

$$F_y - F_s1 = -k_2 l_{o2} s_2 \quad (8)$$

where $F = \{(k_1 + k_2)l_1 - k_1 l_{o1}\}$. Squaring and adding (7) and (8) produces the equation:

$$A_2 s_1 + B_2 c_1 = D_2 \quad (9)$$

where

$$A_2 = 2F_y F$$

$$B_2 = 2(F_x + k_2 d_{12}) F$$

$$D_2 = (F_x + k_2 d_{12})^2 + F_y^2 + F^2 - k_2^2 l_{o2}^2$$

Substituting $G = (F_x + k_2 d_{12})$ and manipulating A_1 in equation (4) as follows:

$$A_1 = k_1(l_1 - l_{o1})d_{12} - F_x l_1 - (k_2 l_{o1} d_{12} - k_2 l_{o1} d_{12}) \\ = F d_{12} - G l_1$$

produce equations (10) and (11) of the form:

$$A_1 s_1 + B_1 c_1 = D_1$$

with

$$A_1 = F d_{12} - G l_1 \quad (10)$$

$$B_1 = F_y l_1$$

$$D_1 = F_y d_{12}$$

$$A_2 = 2F_y F \quad (11)$$

$$B_2 = 2GF$$

$$D_2 = G^2 + F_y^2 + F^2 - k_2^2 l_{o2}^2$$

Eliminating θ_1 between equations (10) and (11) derives the equation:

$$(A_1 B_2 - A_2 B_1)^2 = (A_1 D_2 - A_2 D_1)^2 + (B_2 D_1 - B_1 D_2)^2$$

which is a function only of l_1 and known parameters.

Substitution and simplification result in the following sixth order polynomial:

$$a_1 l_1^6 + a_2 l_1^5 + a_3 l_1^4 + a_4 l_1^3 + a_5 l_1^2 + a_6 l_1 + a_7 = 0$$

with coefficients

$$a_1 = -d_{12}^2 (k_1 + k_2)^6 + 2G d_{12} (k_1 + k_2)^5 \\ - (G^2 + F_y^2) (k_1 + k_2)^4$$

$$a_2 = 6d_{12}^2 k_1 l_{o1} (k_1 + k_2)^5 - 10G d_{12} k_1 l_{o1} (k_1 + k_2)^4 \\ + 4k_1 l_{o1} (G^2 + F_y^2) (k_1 + k_2)^3$$

$$a_3 = d_{12}^2 [2(2F_y^2 + G^2 - H) - 15k_1^2 l_{o1}^2] (k_1 + k_2)^4 \\ + 4G d_{12} [5k_1^2 l_{o1}^2 - 2F_y^2 - G^2 + H] (k_1 + k_2)^3 \\ + 2[G^2(3F_y^2 + G^2 - H) + F_y^2(F_y^2 + k_2^2 l_{o2}^2) \\ - 3(G^2 + F_y^2)k_1^2 l_{o1}^2] (k_1 + k_2)^2$$

$$a_4 = 4d_{12}^2 k_1 l_{o1} [5k_1^2 l_{o1}^2 - 4F_y^2 - 2G^2 + 2H] (k_1 + k_2)^3 \\ - 4G d_{12} k_1 l_{o1} [5k_1^2 l_{o1}^2 - 6F_y^2 - 3G^2 + 3H] (k_1 + k_2)^2 \\ + 4k_1 l_{o1} [k_1^2 l_{o1}^2 (G^2 + F_y^2) - G^2(3F_y^2 + G^2 - H) \\ - F_y^2(F_y^2 + k_2^2 l_{o2}^2)] (k_1 + k_2)$$

$$a_5 = \{3d_{12}^2 k_1^2 l_{o1}^2 [4(2F_y^2 + G^2 - H) - 5k_1^2 l_{o1}^2] \\ - d_{12}^2 (G^4 + H^2 + 2G^2 H + 4F_y^2 k_2^2 l_{o2}^2)\} (k_1 + k_2)^2$$

$$\begin{aligned}
& +\{2Gd_{12}k_1^2l_{o1}^2[5k_1^2l_{o1}^2-6(2F^2+G^2-H)] \\
& +2Gd_{12}[G^4+H^2+2G^2H]\}(k_1+k_2) \\
& -(G^2+F_y^2)k_1^4l_{o1}^4+[2G^2(3F_y^2+G^2-H) \\
& +2F_y^2(F_y^2+k_2^2l_{o2}^2)]k_1^2l_{o1}^2-(G^2+F_y^2)(G^4 \\
& +H^2+2G^2H)
\end{aligned}$$

$$\begin{aligned}
a_6 = & 2d_{12}^2k_1l_{o1}\{k_1^2l_{o1}^2[3k_1^2l_{o1}^2-4(2F_y^2+G^2-H)] \\
& +[G^4+H^2+2G^2H+4F_y^2k_2^2l_{o2}^2]\}(k_1+k_2) \\
& -2Gd_{12}k_1^5l_{o1}^5+4Gd_{12}(2F_y^2+G^2-H)k_1^3l_{o1}^3 \\
& -2Gd_{12}k_1l_{o1}[G^4+H^2+2G^2H]
\end{aligned}$$

$$\begin{aligned}
a_7 = & -d_{12}^2k_1^6l_{o1}^6+2d_{12}^2(2F_y^2+G^2-H)k_1^4l_{o1}^4 \\
& -d_{12}^2k_1^2l_{o1}^2(G^4+H^2+2G^2H+4F_y^2k_2^2l_{o2}^2)
\end{aligned}$$

where $G = F_x + k_2d_{12}$ and $H = F_y^2 - k_2^2l_{o2}^2$.

NUMERICAL EXAMPLE

Here a numerical example illustrates six real solutions. The inputs were as follows: $F_x = 1.000$, $F_y = 1.000$, $k_1 = 2.000$, $k_2 = 3.000$, $l_{o1} = 2.000$, $l_{o2} = 3.000$, and $d_{12} = 2.000$. Only the ratios are important here. Scaled input ratios will produce scaled similar triangles as solutions. The following normalized polynomial was obtained for these inputs:

$$\begin{aligned}
& l_1^6 - 8.000...l_1^5 + 15.120...l_1^4 + 26.112...l_1^3 - 112.728l_1^2 \\
& + 108.762...l_1 - 27.146... = 0,
\end{aligned}$$

which has the following six roots:

$$\begin{aligned}
l_1 = & 0.391..., 1.249..., 2.219..., \\
& 3.581..., 2.940..., -2.379...
\end{aligned}$$

Corresponding values of θ_1 were obtained from equations (10) and (11). Substitution into equations (5) & (6) then produced values for θ_2 . Finally, values for l_2 were obtained from equations (1) & (2). The six solution sets to this problem are as follows:

- 1: $l_1=0.391...$, $l_2=1.623...$, $\theta_1=346.3...$, $\theta_2=183.3...$
- 2: $l_2=1.249...$, $l_2=3.033...$, $\theta_1=223.2...$, $\theta_2=196.4...$
- 3: $l_1=2.219...$, $l_2=2.449...$, $\theta_1=289.2...$, $\theta_2=238.8...$
- 4: $l_1=3.581...$, $l_2=1.993...$, $\theta_1=343.8...$, $\theta_2=208.4...$
- 5: $l_1=2.940...$, $l_2=2.758...$, $\theta_1=295.4...$, $\theta_2=105.6...$
- 6: $l_1=-2.379...$, $l_2=0.383...$, $\theta_1=178.6...$, $\theta_2=351.1...$

Although here six solutions are obtained, only

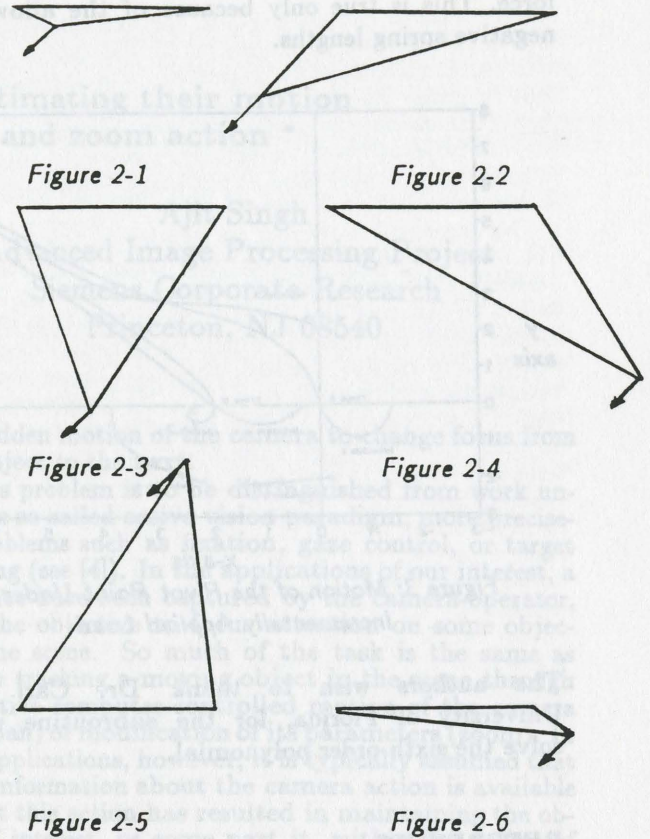


Figure 2: Mathematical Configurations For a Resultant Unit Force

five of these are practical as assembly configurations. Solution six contains a negative spring length which is currently unrealizable. Figure 2 shows all six mathematical solutions noting that in Figure 2-6, l_1 is a negative spring length.

Figure 3 shows the motion of the six roots as an incremental force acts on the pivot point at a forty-five degree angle (solutions obtained at a unit force are marked). Some important results can be derived from this figure. First, when a resultant force of zero is desired, four roots appear along the horizontal axis. These roots appeared in every example this author tried. The other two roots at zero force are the geometry of the system at the free lengths of the springs. Second, solutions 3 and 4 go complex at their point of convergence. This disappearance of real roots appeared in almost every example. It must be noted here that if $l_{o1} + l_{o2} < d_{12}$, two solutions begin as complex roots, and two more disappear as the force increases. Finally, the intersection of different root loci at different points along the force increment gives the appearance that for a unique pivot location there can be more than one resultant

force. This is true only because of the allowance of negative spring lengths.

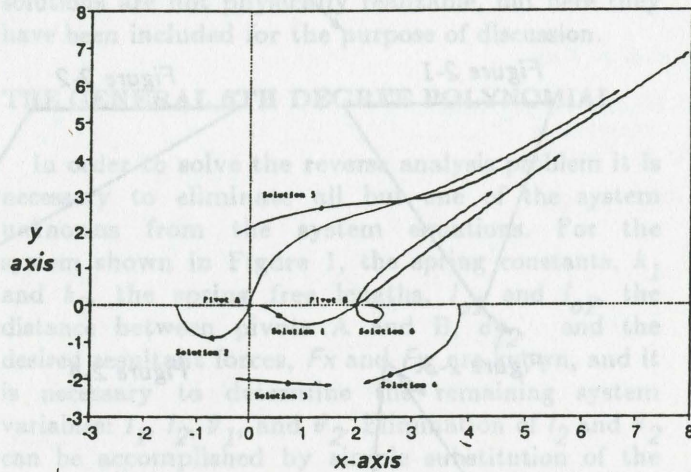


Figure 3: Motion of the Pivot Point Under an Incrementally Applied Load

The authors wish to thank Dr. Carl Crane, University of Florida, for the subroutine used to solve the sixth order polynomial.

REFERENCES

- [1] Vanderplaats, G., *Numerical Optimization Techniques for Engineering Design: With Applications*, McGraw Hill, New York, 1984
- [2] Seirig, A., *Mechanical System Analysis*, International Textbook, Pennsylvania, 1969
- [3] Griffis, M., 1991, "Kinesthetic Control: A Novel Theory for Simultaneously Regulating Force and Displacement," Ph.D. Dissertation, University of Florida

Segmenting moving objects and estimating their motion in the presence of camera pan and zoom action *

Shahriar Negahdaripour & Sulan Shi
Electrical & Computer Engineering Dept.
University of Miami
Coral Gables, FL 33124

Ajit Singh
Advanced Image Processing Project
Siemens Corporate Research
Princeton, NJ 08540

Abstract

We address the problem of segmenting an image to localize moving objects in the presence of camera rotation and zooming action. We compute the flow due to the camera action and subtract it out from the image motion to determine the flow of the moving object. This can be used to determine information, such as the motion and depth, about an object being tracked in a dynamic scene. We present results of experiments with real images to demonstrate the performance of the method.

1 Introduction

One of the difficult problems in the dynamic scene analysis is the detection of moving objects, when the camera is also in motion. Without camera motion, the problem is not so difficult to solve since, in most cases, simple difference operators can be applied to image sequences to isolate regions corresponding to moving objects. The difficulty when the camera also moves is mainly due to the fact that the motion of the camera relative to the scene induces a background optic flow, in addition to the image motion of the moving objects. While there is typically a discontinuity in the flow along the image boundaries of the background and the moving object (due to depth and/or motion discontinuity in the scene), one needs to overcome the chicken-and-egg problem associated with the computation of an accurate flow in the presence of unknown flow discontinuities. That is, the optical flow problem is easier to solve if these boundaries are known, but these cannot be determined until the flow is computed to isolate regions of flow discontinuity.

In this paper, we address this problem in a special case, where the goal is to track an object, and to maintain it within the field of view. A specific example is in video imagery, taken in settings such as TV telecast, which comprises of multiple moving objects. In addition, the camera zooms and pans frequently to focus on the objects of interest. Given a video segment, we would like to determine another video sequence that has been compensated for the effects of camera pan and zoom. Another objective is to simply index the sequences by the particular motion of the target object, such as moving to the right or to the left, or by

the sudden motion of the camera to change focus from one object to the next.

This problem is to be distinguished from work under the so-called active vision paradigm; more precisely, problems such as fixation, gaze control, or target tracking (see [4]). In the applications of our interest, a sequence have been captured by the camera-operator, with the objective to focus attention on some object in the scene. So much of the task is the same as that in tracking a moving object in the scene through the active computer-controlled motion of the camera (e.g., pan) or modification of its parameters (zoom). In such applications, however, it is typically assumed that some information about the camera action is available or that this action has resulted in maintaining the object of interest, or some part it, within the center of the image, thus zeroing out its image motion. Such information and constraints are used to determine other 3D properties of interest, such as the motion or depth of the target object, the camera motion, etc. In our applications, neither the operator actions are known nor they are controlled to produce a precise motion of the image in some region (e.g., zero motion of the image center). To summarize, one of the important objectives in problems in the active vision paradigm is to simplify the 3D reconstruction problem through known or controlled camera activities. In our problem, the 3D reconstruction problem becomes more difficult to solve as a result of the operator action.

2 Related Work

There is a tremendous number of contributions to the vision literature on problems involving the computation of the motion of the camera relative to a scene. Of these, a small percentage deal with the situation where both the camera and some objects in the scene move simultaneously. Where this problem is addressed, the goal is typically either to detect a moving object, not to compute its motion, or it is assumed that the camera motion is known or controlled. For example, [5] propose methods for the detection of moving objects, based on the statistics of the directions of the optical flow vectors, when motion of the camera is known, its rotation or translation is known, or an object is being tracked such that some tracking point has zero flow.

In our approach to the problem, we will not exploit such constraints. This is not to say that such approaches are not attractive. On the contrary, we

*Funding for this research was provided by NSF BCS-9017179.

believe that a robust method for extracting information from time-varying imagery should exploit all available information or constraints. The main reason is that in some application domains, such as video imagery, the camera action is not fully controlled and the tracked objects move in arbitrary ways, in contrast to in robotics or industrial applications where the object or camera motion is more restricted.

Our method is based on modeling the optic flow due to the camera motion, and using the discrepancy between the model and the data. Based on this, we detect regions of the image where the flow is inconsistent with the model, which we expect to be due to the motion of the tracked object. This is consistent with some earlier work (e.g., [1], [2]) for the segmentation of the scene into regions corresponding to different objects. However, we now allow for the camera motion and zooming action, in addition to the motion of tracked object(s). Because of the arbitrary motion(s) of the tracked object(s), or their parts (we allow non-rigid motion), and its independency from the camera action, the optic flow due to a tracked object is not necessarily zero, as it is commonly assumed in some earlier work in object tracking or fixation. We will show how the optical flow due to moving objects and the camera pan/zoom action (and, in general, any rotational motion of the camera) can be separated. Once this is done, we can remove the component due to the camera pan and zoom, leaving us with a flow solely due to moving objects, which can be used to compute 3D motion of the object, indexing of the video sequence, etc.

3 Mathematical Background

Let $\{t, \omega\}$ denote the instantaneous rigid body motion of the camera relative to the scene, described by $Z(x, y)$, the depth of points on its surface from the camera. It is well-known that the relative motion of the camera with respect to the scene induces an image motion given by:

$$\dot{r}_m = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} = A\omega + \frac{1}{Z}Bt,$$

$$A = \begin{bmatrix} \frac{xy}{f} & -(\frac{x^2}{f} + f) & y \\ \frac{y^2}{f} + f & -\frac{xy}{f} & -x \\ 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \\ 0 & 0 & 0 \end{bmatrix}.$$

Further, if we allow for camera zooming action, there is a diverging flow whose magnitude is proportional to the distance of the image point from the origin:

$$\dot{r}_z = \frac{\dot{f}}{f}q,$$

where $q = (x, y, 0)^T$. The total flow due to the camera motion and zoom is:

$$\dot{r}_c = A\omega + \frac{1}{Z}Bt + \frac{\dot{f}}{f}q.$$

In this paper, we consider the situation where the camera is fixed in position, and is only allowed to rotate, in addition to the zooming action¹. Thus, the flow due to the camera motion and zoom is given by

$$\dot{r}_c = A\omega + \frac{\dot{f}}{f}q,$$

which is independent of the depth of background or object points.

3.1 Moving Objects

The previous equations for the image flow assume that the background is stationary. Suppose some objects in the image move. Let R_o describe the region of the image corresponding to moving objects, and let R_b denote the stationary background region. The optic flow in R_b is the result of the rigid camera motion as well as the zooming action. Using the previous expression, the flow in R_b is given by:

$$\dot{r}_b = \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ 0 \end{bmatrix} = A_b\omega + \frac{\dot{f}}{f}q_b,$$

where A , B , and q are defined at points $(x, y) \in R_b$. The optic flow in R_o is the result of the camera motion, the zooming action, as well as the motion of objects. If we denote the flow due to moving objects by (u_o, v_o) , then the total flow over R_o is given by

$$\dot{r}_o = \begin{bmatrix} \dot{x}_o \\ \dot{y}_o \\ 0 \end{bmatrix} = A_o\omega + \frac{\dot{f}}{f}q_o + \begin{bmatrix} u_o \\ v_o \\ 0 \end{bmatrix},$$

where A , B , and q are defined at points $(x, y) \in R_o$.

The objective is to determine the flow due to the camera panning motion and zooming action, and that due to moving objects. This requires segmenting the image into two regions R_o , R_b , and computing (u_o, v_o) over R_o . In general, camera rotation to track an object is restricted to panning motion. The solution procedure is the same whether the rotation is pan only, or it is any combination of pan, tilt, or spin around the optical axis.

The background flow can be written in the form:

$$\dot{r}_b = \begin{bmatrix} -fw_y + f\frac{x}{f} + fw_z\frac{y}{f} - fw_y\frac{x^2}{f^2} + fw_x\frac{x}{f}\frac{y}{f} \\ fw_x - w_z\frac{x}{f} + f\frac{y}{f} - fw_y\frac{x}{f}\frac{y}{f} + fw_x\frac{y^2}{f^2} \\ 0 \end{bmatrix}_{(x,y) \in R_b}$$

We assume the focal length f is known, initially (denote this $f(0)$). In practice, knowing $f(0)$, and computing $\dot{f}(t)$ as we discuss later, we can determine $f(t)$ by integration. With this assumption, we replace $\frac{x}{f}$ by x , and $\frac{y}{f}$ by y , without loss of generality. (In other

¹When translation is introduced, the problem is more complex and require special attention. This is beyond the scope of this paper.

words, x and y are measured in units of focal length.) Therefore, we can rewrite the above equation as:

$$\dot{r}_b = \begin{bmatrix} -fw_y + \dot{f}x + fw_z y - fw_y x^2 + fw_x xy \\ fw_x - w_z x + \dot{f}y - fw_y xy + fw_x y^2 \\ 0 \end{bmatrix}_{(x,y) \in R_b}$$

Similarly, over the region corresponding to moving objects, we have:

$$\dot{r}_b = \begin{bmatrix} -fw_y + \dot{f}x + fw_z y - fw_y x^2 + fw_x xy + u_o \\ fw_x - w_z x + \dot{f}y - fw_y xy + fw_x y^2 + v_o \\ 0 \end{bmatrix}_{(x,y) \in R_o}$$

In the background region, each flow component is given by a quadratic function with four independent coefficients in $a = (fw_x, fw_y, fw_z, \dot{f})$, where $a = (a_1, a_2, a_3, a_4)$. One can determine these coefficients using the flow information from a minimum of two points (each point provides us with two constraint equation, one for each of the x and y components of \dot{r}_b). Generally, a least-square formulation can be applied for more robust results. Once these are determined, it can be used to estimate the flow due to the moving object from

$$\begin{bmatrix} u_o \\ v_o \\ 0 \end{bmatrix} = \dot{r}_o - \begin{bmatrix} -fw_y + \dot{f}x + fw_z y - fw_y x^2 + fw_x xy \\ fw_x - w_z x + \dot{f}y - fw_y xy + fw_x y^2 \\ 0 \end{bmatrix}_{(x,y) \in R_o}$$

This requires, however, that we have segmented the image into regions corresponding to the background and the moving objects. We consider a scheme that gives us the segmented image and the camera rotation and zooming parameters simultaneously. The segmentation problem, as we see, depends on the distribution of the error values from a least-square error minimization problem we will formulate to determine the pan and zoom parameters.

3.1.1 Identifying Moving Objects

Simply, the idea is to take advantage of the facts that

- The flow is a quadratic function of image position over the background region.
- Over the region corresponding to moving objects, the flow consists of two components, one which is the same quadratic function of position as over the background region, and the second which is due to the motion of the object.

For the object, let us denote the first component as noise, and the second as the signal². If the signal component is small (object moves by only a small amount),

²Which of the two components is considered as noise and which is denoted signal depends on the objective. Since our goal is to segment the image into two regions, background and moving objects, it is the flow due to the object motion which allow us to do this, as we see soon; thus, the choice of the term signal for

then the flow is dominantly quadratic with the same coefficients as the flow over the background region; these coefficients, as we showed above, give the rotation and zooming action of the camera. Hence, we can fit a quadratic function to the flow over the whole image to determine the coefficients (this is the problem, where the flow due to the moving object can be treated as noise).

Since we want to segment the image, we hope that the flow due to the moving objects (signal) is large. In such a case, the fitting of a quadratic flow over the region corresponding to these objects will be erroneous. In special cases, it is possible that this flow is also quadratic such that the total flow (due to object motion and camera rotation and zooming action) remains quadratic. The important thing, however, is that the resulting quadratic will have different coefficients as the one that fits over the background region. More precisely, suppose that

$$\begin{bmatrix} u_o \\ v_o \end{bmatrix} = \begin{bmatrix} u_0 + u_x x + u_y y + u_{xx} x^2 + u_{xy} xy + u_{yy} y^2 \\ v_0 + v_x x + v_y y + v_{xx} x^2 + v_{xy} xy + v_{yy} y^2 \end{bmatrix}$$

We then have, for the flow over the moving objects,

$$\dot{r}_o = \begin{bmatrix} u'_0 + u'_x x + u'_y y + u'_{xx} x^2 + u'_{xy} xy + u'_{yy} y^2 \\ v'_0 + v'_x x + v'_y y + v'_{xx} x^2 + v'_{xy} xy + v'_{yy} y^2 \\ 0 \end{bmatrix},$$

where

$$\begin{aligned} u'_0 &= u_0 - a_2 u'_x = u_x + a_4 & v'_x &= v_x - a_3 \\ u'_y &= u_y + a_3 & v'_y &= v_y + a_4 \\ u'_{xx} &= u_{xx} - a_2 & v'_{xx} &= v_{xx} \\ u'_{xy} &= u_{xy} + a_1 & v'_{xy} &= v_{xy} - a_2 \\ u'_{yy} &= u_{yy} & v'_{yy} &= v_{yy} + a_1 \end{aligned}$$

This is different from the flow over the background region:

$$\dot{r}_b = \begin{bmatrix} -a_2 + a_4 x + a_3 y - a_2 x^2 + a_1 xy \\ a_1 - a_3 x + a_4 y - a_2 xy + a_1 y^2 \\ 0 \end{bmatrix}$$

We use this fact, the difference in the flow coefficients over the two regions, to segment the image. To do this, we consider a region around each image point $\{i, j\}$. For convenience, a square region R_{ij} of size $n \times n$ is used (n can be as small as two pixels, but should be larger for more robust results). We hypothesize that the region is a part of the stationary background. In this region, we compute, from a least-square formulation, the flow coefficients a by minimizing

$$E(a^{ij}) = \sum_{(x,y) \in R_{ij}} [\dot{x} - (-a_2 + a_4 x + a_3 y - a_2 x^2 + a_1 xy)]^2$$

this component of the flow. The more different the flow due to moving objects relative to that from pan and zoom, the better we can segment the two. However, if we want to determine the parameters for the rotation and the zooming action from a least-squares formulation over the whole image, then the flow due to the moving object would be considered as noise.

$$+[\dot{y} - (a_1 - a_3x + a_4y - a_2xy + a_1y^2)]^2,$$

where $a^{ij} = (a_1, a_2, a_3, a_4)$ for pixel $\{i, j\}$. This is a quadratic error function, and the minimization problem has a closed-form solution for the flow coefficients. We repeat the computation for each image point.

The following observations are in order:

- If the hypothesis is true, the flow model is exact, and thus the estimate \mathbf{a} for the flow coefficients should be accurate, up to the accuracy in the estimated flow ($\{\dot{x}, \dot{y}\}$ in the region). Furthermore, the total error in the region $E(\mathbf{a}^{ij})$ will be small.
- If region R_{ij} contains a moving object, the flow coefficients will be different from the ones for the background region. Further, since the flow model is not exact in most cases (because of the flow due to the moving object), the total error for the region $E(\mathbf{a}^{ij})$ is expected to be large. In very special cases, the flow model may still be accurate (in which case the error will be small), but the flow coefficients are still different.
- If the patch R_{ij} is on the boundary of a moving object and the background, the flow coefficients are such that they correspond to an average flow for the two regions. We expect the total error for the region $E(\mathbf{a}^{ij})$ to be larger than that for a stationary background region.

Based on these observations, we expect the points in the background to give a good estimate of the true flow coefficients corresponding to the camera pan and zoom, \mathbf{a} . Further, the error in the quadratic fit for these points is expected to be small. The estimates of the coefficients from the regions corresponding to the moving object will be erroneous, and the error in the fit is expected to be large. Points for which the computation of the flow coefficients are done in square regions that include the boundary between the background and the moving object have error values in between those for the background and moving object points. We can thus segment the image based on some appropriate threshold T for the squared-error in the quadratic model, $E(\mathbf{a}^{ij})$. That is, we label a point as a background point, if $E(\mathbf{a}^{ij}) < T$. Other points with larger errors than T will be labeled as object (or non-background) points. We can improve the results by checking for consistency in the estimated flow coefficients over all background points. We can do this by testing if

$$|\mathbf{a}^{ij} - \bar{\mathbf{a}}|^2 \leq \epsilon,$$

where $\bar{\mathbf{a}}$ is the average flow coefficient vector over all points labeled as background points, and ϵ is some error threshold. If this test fails, the point is not considered in the next computation. Note that the mislabeling is inevitable for some points near the boundary of the object and background. In fact, this is a similar problem to segmenting a noisy binary image into the background and object regions, by choosing an appropriate threshold (if we treat the error as noise-corrupted gray values in a binary image). In most cases, there is some mislabeling of the boundary points

due to spread in the distribution of the error values. However, this is not so important since the goal at this point is to identify a large portion of the background. Therefore, we need to set the threshold low enough to make sure that only background regions have been included. Once this is done, the next computational stage is to determine the flow coefficients over all points labeled as background by minimizing

$$E(\mathbf{a}) = \sum_{(x,y) \in \hat{R}_b} [\dot{x} - (-a_2 + a_4x + a_3 - a_2x^2 + a_1xy)]^2 + [\dot{y} - (a_1 - a_3x + a_4y - a_2xy + a_1y^2)]^2,$$

where \hat{R}_b is the region labeled as the background. Let us denote the solution as $\hat{\mathbf{a}} = (\hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{a}_4)^T$. This gives the estimated solution for the camera rotation and zooming parameters. We can now subtract it out from the total flow over the region labeled as moving objects (the non-background region):

$$\begin{bmatrix} u_o \\ v_o \\ 0 \end{bmatrix} = \dot{\mathbf{r}}_o - \begin{bmatrix} -\hat{a}_2 + \hat{a}_4x + \hat{a}_3y - \hat{a}_2x^2 + \hat{a}_1xy \\ \hat{a}_1 - \hat{a}_3x + \hat{a}_4y - \hat{a}_2xy + \hat{a}_1y^2 \\ 0 \end{bmatrix}.$$

It is conceivable that some of these points are background points. Where this is the case, the residual flow estimated from the above equation should be very small.

Points which passed the error test, but not the consistency check in the flow coefficients, are typically points on the boundary. We can adapt a heuristic for how to treat them more intelligently. Alternatively, another approach is to use edge information, if available. Points on the background side of the edge are labeled as background, and those on the other side are labeled as moving object points. Or we can determine, for each point, the discrepancy in the flow with that predicted by the pan and zoom model. This is different from the error computed earlier, which depended on the total error in a square region around the point (and tends to smooth out the error function over the boundary). Based on the new error value, we can decide how to label the point. We have not implemented any of these ideas yet, but plan to do so to improve the segmentation.

4 Experimental Results

In this section, we present results of four experiments with real images. These experiments demonstrate our proposed scheme for decomposing the optical flow field resulting from a moving object in the presence of camera pan and zoom action, in order to extract the moving object from a background. All experiments were done on a 64x60 image. The optical flow fields were computed using the method described in [3] based on a generalized brightness change constraint equation (any other method may be used). The size of the square regions of their computation of the flow coefficients is 8 pixels on each side. The threshold used for segmentation is the first minimum in the error histogram.

The scene for this example is shown in Fig.1. Fig.1a is the original (before subsampling) reference image

(512×512), showing a piece of rock on a texture background. The experiment involves both rotation about y -axis (pan) and the zooming of the camera (The image after motion is not shown here). No prior knowledge of the real values of the pan and zooming parameters was available, because the camera was manually turned by a small angle about the y -axis and the lens focal length was manually changed (zooming). The motion of the object is along the y -axis direction.

Fig.1b is the computed flow field. The flow coefficients were computed as described in section 2.1.1; that is, by minimizing, within a square region centered at each image point, the square of the discrepancy between the computed flow and that predicted by the quadratic model. Fig.1c is the histogram of the errors for all the image points. In this graph, the x -axis indicates the error and y -axis is the number of points which have the same error value. A threshold of $T = 1.8$ was used to separate the object from the background. Points where $E < T$ were labeled as the background. All other points were labeled as the object. No consistency check was applied because we have yet to determine an accurate estimate of the flow coefficients (due to inaccurate knowledge of camera parameters). Despite this, good segmentation has been obtained (see Fig.1d which is the segmented object, and the flow due to object motion after the flow due to pan and zoom is subtracted out). The explanation for this behavior is as follows: The estimation of 3D parameters from 2D image data depends on the knowledge of calibration parameters, both intrinsic and extrinsic. In our case, the important parameters are the intrinsic ones, the focal length, the image center, and the y to x aspect ratio. We have used coarse estimates of these parameters in our algorithm. Any error in the knowledge of these parameters affects the estimates in the 3D camera pan and zoom parameters. Despite inaccurate estimates of these parameters within each square window, the error in the quadratic flow fit is small because the flow model is still accurate. Therefore, we can do segmentation based on the error alone, even though the actual flow coefficients may not be accurate. Yet, it is encouraging to know that the method is insensitive to accurate knowledge of these parameters.

Fig.1e shows the flow field for the moving object superimposed on the image. Fig.1f is the flow over the whole image after subtracting out the camera pan and zoom flow. Over the background region, this is the discrepancy between the computed flow and that predicted by the rotation and zooming model. Over the object region, it is the same flow shown in Fig.1d.

The next three experiments are simulations of various events during the Desert Shield and Storm operations. Fig.2a is the image before motion of the camera and the object, a toy soldier, on a sandy surface (background). The object moves a short distance along the x -axis direction. This experiment was designed in such a way that the flow due to camera pan and the object motion are very similar (parallel to the x -axis) in the moving object area, and the net flow due to object motion is quadratic. Fig.2b is the computed optical flow field. Fig.2c is the histogram of the errors. A threshold of $T = 1.5$ was used for segmentation. Fig.2d is

the flow field for the moving object, after subtracting out the camera pan and zoom. Fig.2e is the same superimposed on the image. Fig.2f shows the flow over the whole image after subtracting out the camera pan and zoom flow.

The next two experiments show that the consistency of the results with a large depth discrepancy between the object and the background. Fig.3 shows the results of the third experiment. The object is an attacking soldier moving along the x -axis direction. Fig.3a is the image before motion. Fig.3b is the computed optical flow field. Fig.3c is the histogram of errors. A threshold of $T = 1.5$ was used for this case. Fig.3d is the flow field for points labeled as belonging to the moving object. The same is shown in Fig.3e, superimposed on the image. Fig.3f is the difference flow between the computed flow and that predicted by the pan and zoom model, over the whole image. Fig.4 shows similar results for a toy truck and soldier with a threshold of $T = 2$ for segmentation.

5 Summary

We have studied the problem of segmenting the image of a scene to localize moving objects with the camera undergoing a simultaneous rotational motion and zooming action. One particular application is in video imagery, comprising of multiple moving objects. In addition, the camera may zoom and pan frequently to focus on some objects of interest.

We have showed how the segmentation of the flow due to the moving objects and due to the camera action can be achieved by determining the camera zooming and pan parameters from the flow in a region, expected to be the stationary background. These parameters are found in closed form by minimizing the discrepancy between the computed flow and that predicted due to a zooming and panning motion. Experimental results have been given to support the analysis.

References

- [1] Adiv, G. "Determining 3-D Motion and Structure from Optical Flow Generated by Several Moving Objects," *IEEE PAMI*, Vol 7, No 4, July, 85.
- [2] S. Negahdaripour, S. Lee, "Motion recovery from image sequences using only first order optical flow information," *Int. Journal Computer Vision*, July, 92.
- [3] S. Negahdaripour, C.H. YU "A Generalized brightness change model for computing optical flow," to be presented at *ICCV*, Berlin, Germany, May, 93.
- [4] Swain, M.J., & M. Stricker, "Promising directions in active vision," Final report of the NSF Workshop on Active Vision, University of Chicago, August 5-7, 91.
- [5] Thompson, W.B., & T.C. Pong, "Detecting moving objects," *Proc. ICCV*, June, 87.

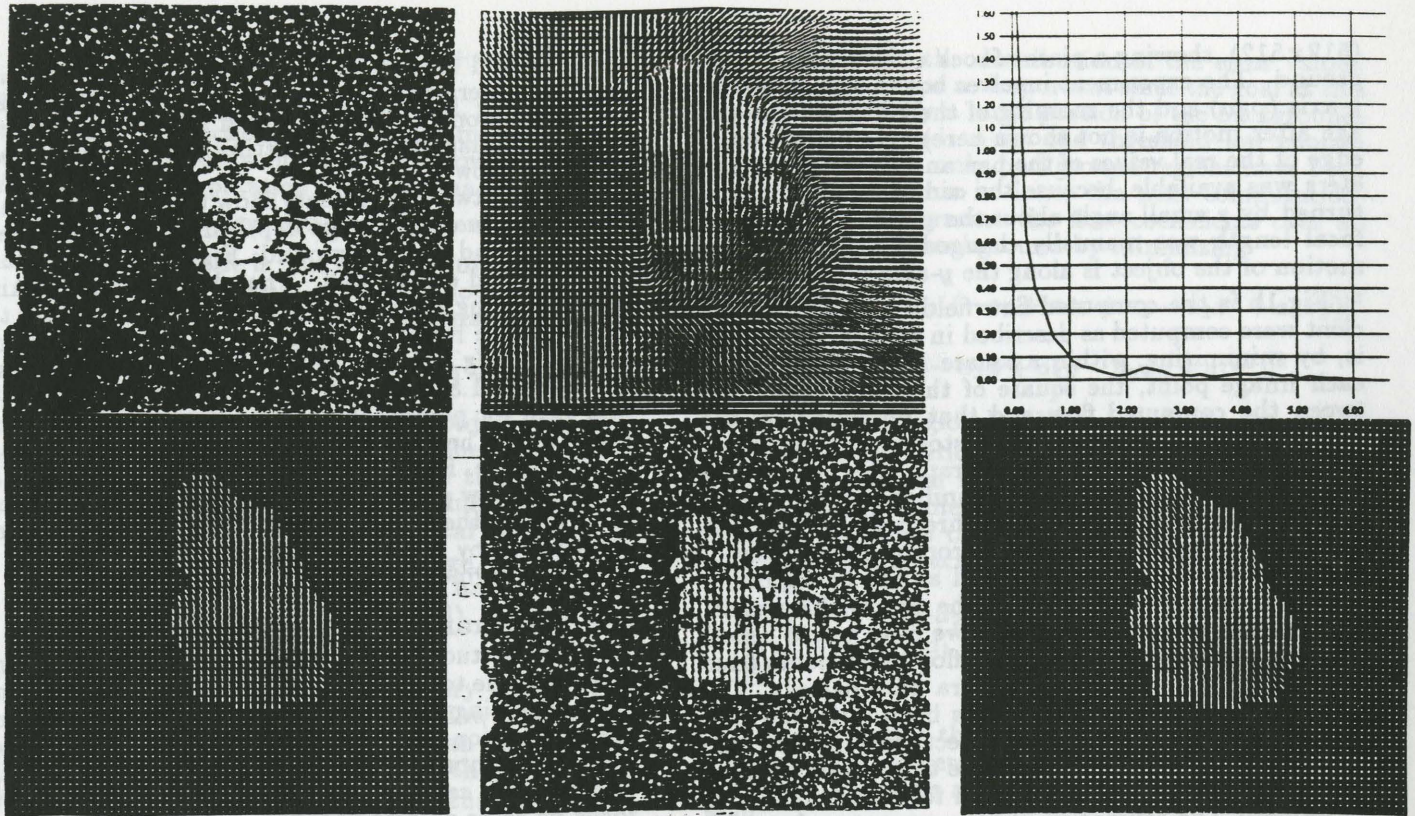


Figure 1. Results of first experiment with rock image.

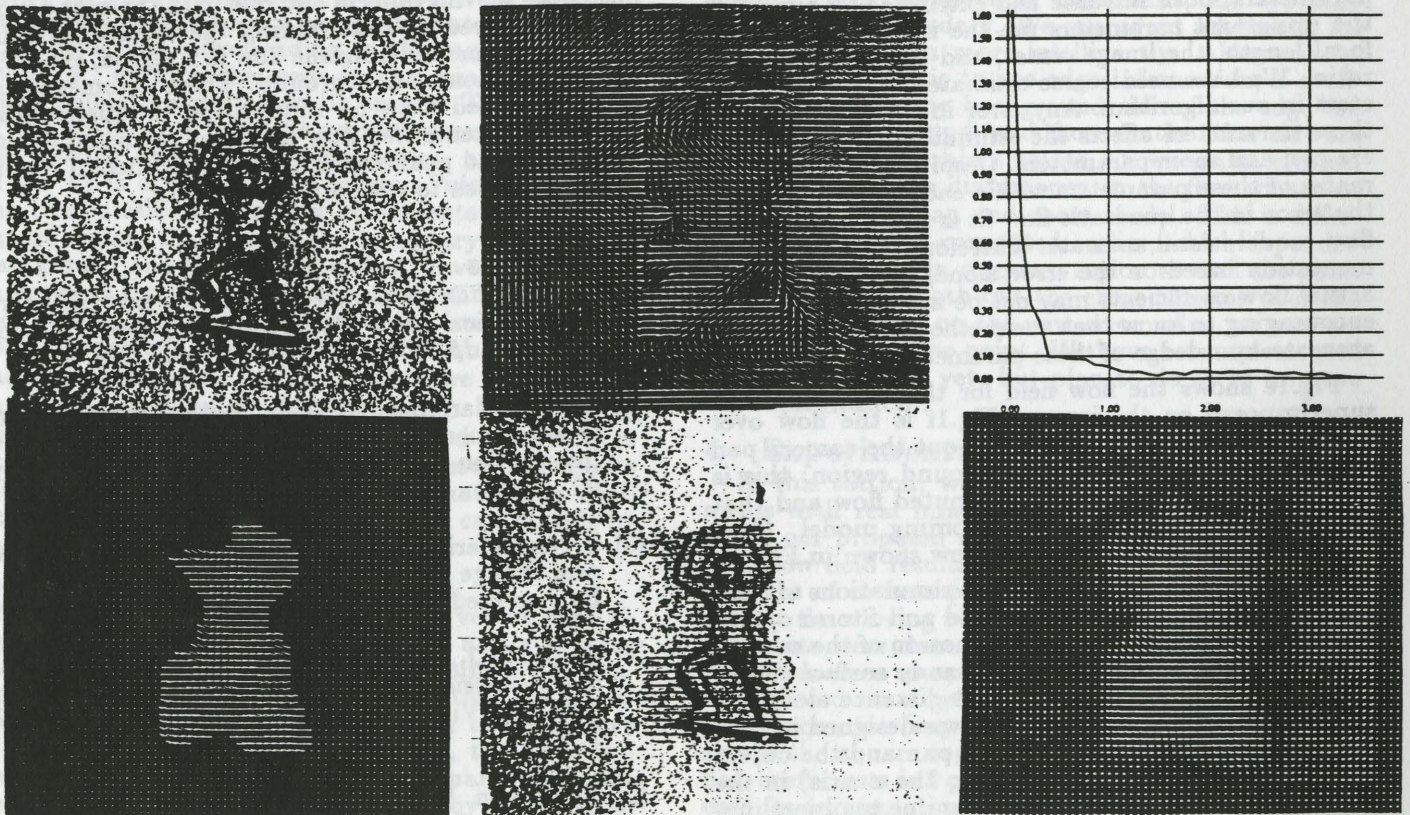


Figure 2. Results of second experiment with soldier on sand.

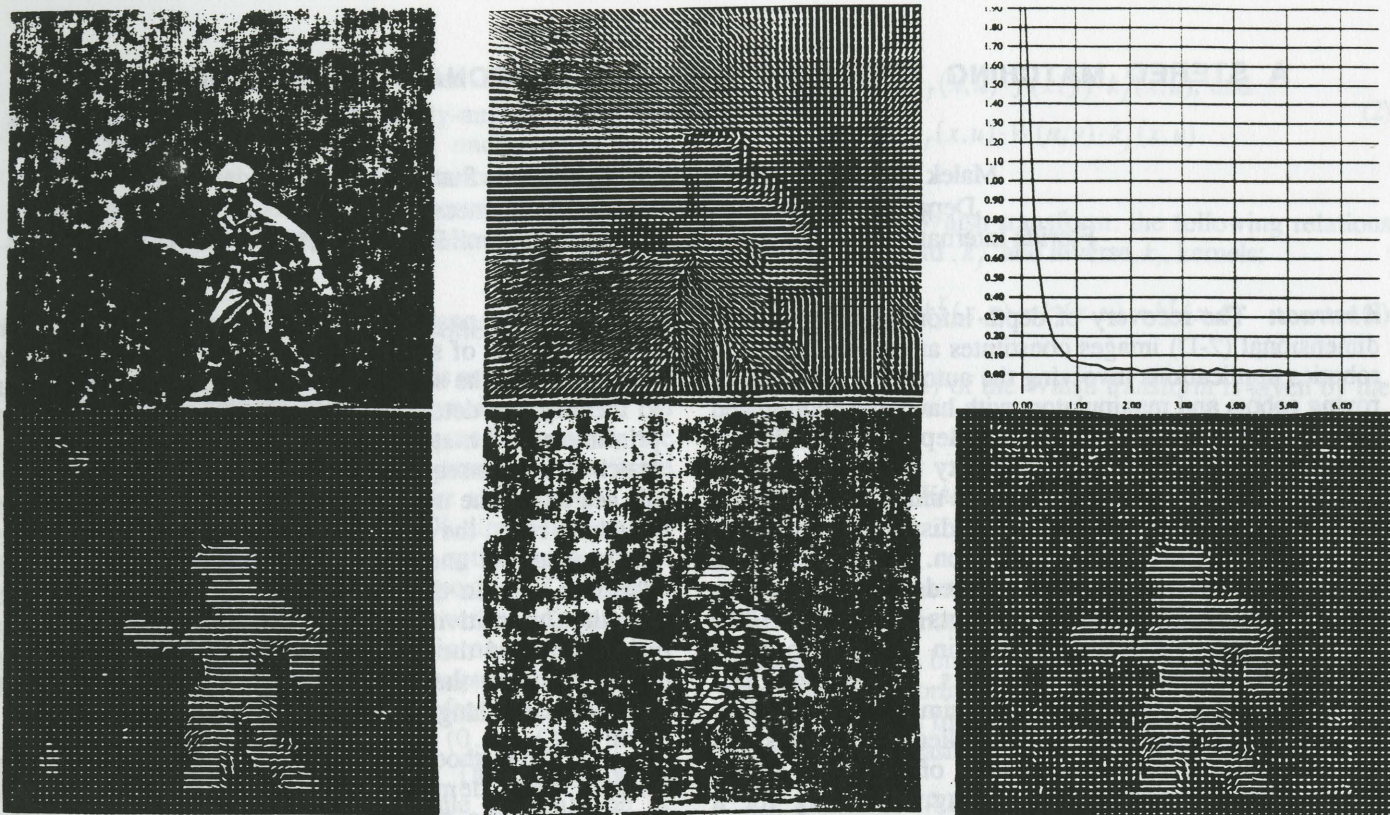


Figure 3. Results of third experiment with walking soldier.

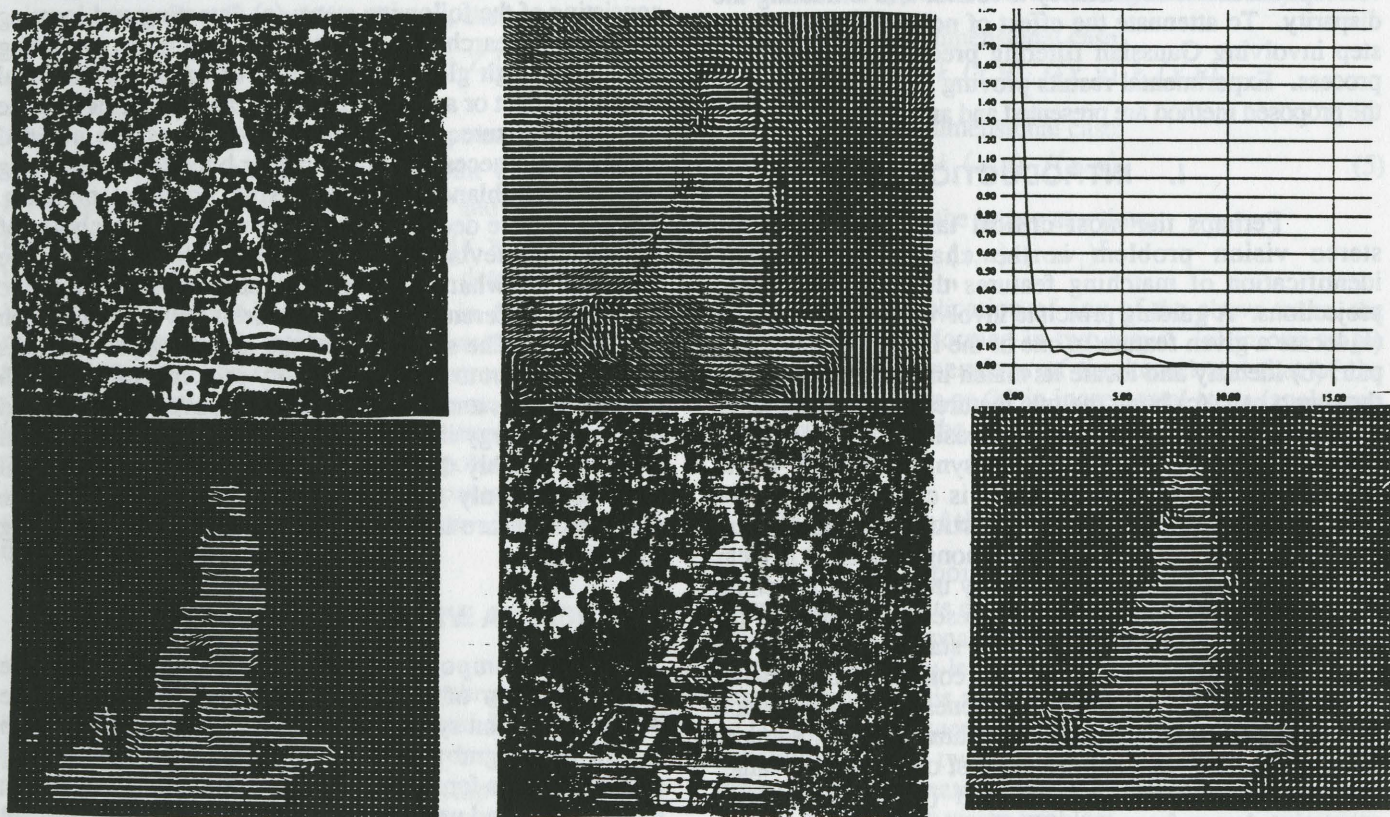


Figure 4. Results of fourth experiment with soldier and truck.

A STEREO MATCHING TECHNIQUE USING ORTHOGONAL TRANSFORMATIONS

Malek Adjouadi, Frank Candocia, and Habibie Sumargo
Department of Electrical and Computer Engineering
Florida International University, University Park, Miami, Florida, 33199

Abstract: The recovery of depth information from two-dimensional (2-D) images constitutes an important task in robotics applications involving the automated guidance of roving robots and manipulators with hand-eye coordinated systems. It is well known that depth information is inversely proportional to the disparity measure sought in the 2-D images. This paper presents a unique stereo feature matching method that extracts the disparity measure in order to recover the depth information. In this method, a stereo pair of images are transformed row for row into strings carrying the Walsh coefficients as attributes. The choice of the Walsh coefficients in contrast to other orthogonal transform coefficients is due to their computational simplicity and their simple interpretation. The stereo matching technique applied here adopts and integrates all the fundamental steps of the stereo vision problem, namely: feature detecting, searching for a potential match, bringing the two strings into correspondence if consistency is found, and extracting the disparity. To attenuate the effect of noise a preprocessing step involving Gaussian filtering precedes the matching process. Experimental results proving the effectiveness of the proposed method are presented and analyzed.

I. INTRODUCTION

Perhaps the most critical task inherent of the stereo vision problem is the characterization and identification of matching features through their stereo projections. A guiding principle involves three basic steps: (a) locate a given feature in one of the images of the stereo pair; (b) identify and locate its match in the other image of the stereo pair; (c) bring the two features to correspondence and measure their disparity. These basic steps sound rather simple to achieve, and they are in a synthetic world. In the real world, however, this problem is complicated by the presence of noise, geometric distortions, occlusion, and discontinuity in the illumination, among others. It is clear that certain constraints must be fully utilized to overcome these drawbacks. Most critical of these constraints are: (a) the camera system must be very stable satisfying the epipolar line constraint to allow for a controlled and ordered feature search strategy; (b) the uniqueness constraint, which in its definition states that a given feature point or element in the image may be assigned at most one disparity value, confines the search strategy; and (c) the continuity constraint drawn from the cohesiveness of matter, which brings some credence to filtering as a preprocessing step to attenuate the effect of noise points [1, 2].

In the past few years various techniques with various degrees of success have been reported [3-8]. The main theme in the solution of the stereo vision problem is to identify and determine the most appropriate matching primitives and matching rules to identify the potential matches, and consequently derive the disparity measure. In the search for the most effective matching primitives and matching rules, the important aspects of simplicity in the implementation and processing time remain as critical issues [6, 7]. In this search also, it is evident that the stability (insensitivity to small variations) of the matching primitives, and the practical soundness of the matching rules constitute the key issues for a successful overall search and matching strategy.

The method presented here addresses these pressing issues and demonstrates through experimental implementations the soundness and feasibility of the method. This method is designed as an orderly process consisting of the following steps: (a) detecting and locating features, (b) searching for a potential match, (c) providing support through global consistency to declare a potential match a correct or an incorrect match, and (d) extracting the disparity measure. Throughout this process, we avoided setting any unnecessary thresholds or bounds in measuring feature resemblance or in assessing potential matches. Still, subjective decisions had to be made in the choice of the standard deviation of the Gaussian filter, and in the threshold of what constitutes a feature point from the amplitude difference in the first derivative (first Walsh coefficient). The soundness of the matching primitives is due in great part to the orthogonality principle of the Walsh transformation, and to the fact that the overall search and matching strategy is designed as an integrated and gradual process of highly discriminating steps, where in the final analysis, not only that the features are matched, but the disparity measure is directly extracted from the matching result itself.

II. A VISION SYSTEM MODEL

An important motivating factor in the implementation of the proposed stereo method, is the design of a vision system model, wherein the disparity map is now an integral part of the input to the many ensuing visual processes for scene segmentation, object recognition, interpretation and understanding. This vision system model is shown in Figure 1. The inputs to the main visual processes are the input image(s) $f(x, y)$, the disparity map

$d(x, y)$, and t for time-varying images. In many robotics applications, using this dimensionally-augmented system (2½-D image representation) will undoubtedly yield enhanced image analysis and interpretation. Such a vision system model will be ideal for the automated guidance of roving robots and manipulator tasks with hand-eye coordinated systems.

III. GEOMETRY OF THE CAMERA MODEL

A single camera is used in this method to simulate the two-parallel-camera model shown in Figure 2. In displacing the camera, an attempt is made to satisfy the epipolar line constraint. A viewer-centered coordinate system and positive image planes are assumed. In this displacement of distance B in the X direction, another attempt was to maintain the camera's optical axes parallel in the two positions, pointing in the positive Z direction. With this camera configuration, the vertical disparity is zero, and the horizontal disparity is non-zero, and is inversely proportional to the depth information. With a positive image plane assumed, the reference point is the lens center defined at point $(x, y, z) = (0, 0, -f)$, with f denoting the focal length of the lens. Thus, the disparity measure is established through simple triangulation as follows:

$$\text{since } Z_2 = Z_1, \text{ and } X_2 - X_1 = B, \text{ then} \quad (1)$$

$$x_2 - x_1 = d(x_1, y_1) = (f \cdot B) / (f + Z_1)$$

where, $d(x_1, y_1)$ is the disparity at point (x_1, y_1) , and $Z_1 = Z_1(x_1, y_1)$ is the depth value at point (x_1, y_1) . It can be noted at this point that since f and B are known quantities, depth Z at any given point (x, y) in the image plane can be estimated once $d(x, y)$ is determined. It is thus evident, that the important task is in determining the disparity measure. It is important to stress at this point that the epipolar line constraint is the strongest and most critical constraint applicable to the stereo matching process. For this reason, a stable, well-calibrated stereo-head system should be utilized. Violation of the epipolar line constraint yields mismatches as well as unexpected matches of resembling features which are in clear violation of the uniqueness constraint.

IV. APPLICATION OF THE WALSH TRANSFORM

Given that the conditions of orthogonality, kernel symmetry, and kernel separability (equal functionality of the kernel in (x, u) and (y, v)) are met in the Walsh transform [9], the following relationship is established relating the transform $W(u, v)$ of an image $f(x, y)$:

$$W(u, v) = k_f(x, u) \cdot f(x, y) \cdot k_f(x, u), \text{ and} \quad (2)$$

$$f(x, y) = k_f(x, u) \cdot W(u, v) \cdot k_f(x, u)$$

Note that in the Walsh transform, the following relations hold for the forward, k_f , and inverse, k_i , kernels:

$$k_f(x, u) = k_f^T(x, u) = k_i(x, u) = k_i^T(x, u) \quad (3)$$

The kernel $k_f(x, u)$ of the Walsh transform is given by the relation:

$$k_f(x, u) = WAL(c, r) = \prod_{i=0}^{n-1} (-1)^{\beta_i(c) \beta_{n-1-i}(r)} \quad (4)$$

where c and r denote the column and row number of the Walsh transformation matrix, respectively. $\beta_j(p)$ represents the j^{th} bit in the binary form of the number p . The Walsh transform is thus a matrix consisting of a set of orthonormal basis functions, with function values of either +1 or -1. These basis functions are mutually orthonormal.

Before we provide an interpretation of the Walsh coefficients, let us indicate at this point that the Walsh kernel can be applied in the following different ways:

(a) in the two-dimensional case:

$$W(u, v) = k_f(x, u) \cdot f(x, y) \cdot k_f(x, u)$$

(b) in the one-dimensional case:

$$W(u, y) = k_f(x, u) \cdot f(x, y) \quad (5)$$

(c) one row of the image at a time:

$$W_r(u) = k_f(x, u) \cdot (\text{row}_i)^T$$

In the application of any of the above tasks, the process can be performed in two ways: (a) in one single step where the dimensions of the kernel and that of the image are the same; or (b) in incremental steps (analogous to convolution) if the kernel dimensions are smaller than that of the image.

4.1 Description of the Walsh Coefficients

If two points $p_1(x, y)$ and $p_2(x, y)$ are corresponding points on the two image planes of the stereo pair, then it is reasonable to assume and to expect that the neighborhood gray-levels of both points have the same characteristics. This assumption and expectation is based on both (a) the uniqueness constraint; and (b) the continuity constraint. These two constraints constitute the driving force in the application of the Walsh coefficients with respect to the stereo vision problem.

(a) Interpretation of the Walsh Coefficients

Assume the following notation in a simple example of an $n \times n$ Walsh kernel applied to a given image. Consider the following:

$$k_f(x, u) \cdot f(x, y) \cdot k_f(x, u) = A \cdot k_f(x, u) = W \quad (6)$$

The weight factor associated with the kernel is purposely ignored at this point. Let us also define the following differential operators:

$$\frac{\partial}{\partial x} = [1 \quad -1]; \quad \left(\frac{\partial}{\partial x} \right)_{2p} = [1 \quad 1 \quad -1 \quad -1]$$

$$-\left(\frac{\partial}{\partial x} \right)_{2p} = [-1 \quad -1 \quad 1 \quad 1]; \quad \text{and} \quad \frac{\partial^{2*}}{\partial x} = [1 \quad -1 \quad -1 \quad 1]$$

$\frac{\partial}{\partial x}$ denotes the first derivative of two neighboring points,

and $\left(\frac{\partial}{\partial x} \right)_{2p}$ denotes the first derivative of two two-point

neighborhoods. The (2^*) notation is used in view of the operator's resemblance to the second derivative operator

given by $\frac{\partial^2}{\partial x^2} = [1 \quad -2 \quad 1]$.

Given the above defined operators, the following relationships are thus established. In these relationships, it is assumed that the Walsh kernel is sequency ordered.

$$W_{i0} = \sum_{j=0}^{n-1} A_{ij}$$

$$W_{i1} = [A_{ij}; \quad j = 0, 1, \dots, n-1] \cdot \left(\frac{\partial}{\partial x} \right)_{(n/2)p}^T$$

$$W_{i2} = [A_{ij}; \quad j = 0, 1, \dots, n-1] \cdot \left(\frac{\partial^{2*}}{\partial x^2} \right)_{(n/4)p}^T$$

$$W_{i3} = [A_{ij}; \quad j = 0, 1, \dots, n-1] \cdot \bigcup_{n/4} \left(\frac{\partial}{\partial x} \right)_{2p}^T$$

$$W_{i4} = [A_{ij}; \quad j = 0, 1, \dots, n-1] \cdot \bigcup_{n/4} \left(\frac{\partial^{2*}}{\partial x^2} \right)^T \quad (7)$$

⋮

$$W_{i(n-1)} = [A_{ij}; \quad j = 0, 1, \dots, n-1] \cdot \bigcup_{n/2} \left(\frac{\partial}{\partial x} \right)^T$$

Note that if the weight factor $(1/N)$ is considered in the Walsh kernel, then A_{ij} will be the mean gray-level value of the j^{th} column of the image and W_{00} will be the mean gray-level value of the image. All A_{ij} elements will

be affected by the weight factor $(1/N)$, and all W_{ij} elements will be affected by the weight factor $(1/N^2)$. It should also be noted that for $j = 0, 1, \dots, n-1$, the elements $(A_{ij}; i = 0, 1, \dots, n-1)$ are functionally equal to the elements $W_{ij} (W_{ij}; j = 0, 1, \dots, n-1)$ for $i = 0, 1, \dots, n-1$. The difference is that the A_{ij} elements are the one dimensional Walsh transform of a given column of the image, and the W_{ij} elements are the two-dimensional Walsh transform of the $n \times n$ image where n is the size of the Walsh kernel used.

Clearly, higher order coefficients may not be that evident to formulate given large Walsh kernels. However, there is an interesting relationship which can be established upon review of the Walsh basis functions obtained through the relation: $k_f(x, u) \cdot k_f^T(x, u) = W$. The elements W_{kl} of Matrix W satisfy the relation: $W_{kl} = W_{0k} \cdot W_{l0}$. An illustrative example of an 8×8 ordered-matrix of Walsh basis functions is shown in Figure 3.

(b) Significance of the Walsh Coefficients

The coefficients of the Walsh series have the following merits: 1) The coefficients are mutually independent; therefore, the results of matching the coefficients are mutually independent. 2) The computation of the coefficients is simple, involving only addition and subtraction operations. Furthermore, the decimation-in-time approach can be implemented to realize the fast Walsh transform. 3) The features present in the image are embedded in all the coefficients, but in very specific ways. It is noted however, that the higher order coefficients yield smaller amplitudes, and can be considered as carrying less weight in their use as matching primitives.

The application of the Walsh coefficients presented in this study serve two purposes: (1) Through the first and second derivatives which were related to the first and second derivatives, the feature locations are extracted by locating a peak in the first derivative and a zero-crossing in the second derivative, and (2) through the complete set of strings of incremental ξ -varying Walsh coefficients of the rows of the stereo-pair of images, these features are brought to correspondence in order to obtain the disparity measure.

V. TRANSFORMATION OF THE STEREO IMAGES TO WALSH ATTRIBUTES

In order to apply the string-to-string matching process which is described in the next section, a stereo-pair of 2-D images are first decomposed one row at a time into 1-D Walsh-transformed profiles composed of incremental ξ -varying Walsh coefficients. We let $A_k(\xi)$ and $B_k(\xi)$ denote the strings of the left and right k^{th} row of the stereo-

pair of images, and whose attributes are the incremental ξ -varying Walsh coefficients:

$$\begin{aligned} A_k(\xi) &= [a_0(\xi) \ a_1(\xi) \ a_2(\xi) \ \dots \ a_{n-1}(\xi)]^T \\ B_k(\xi) &= [b_0(\xi) \ b_1(\xi) \ b_2(\xi) \ \dots \ b_{n-1}(\xi)]^T \end{aligned} \quad (8)$$

where T denotes the transpose of the vectors. The variable ξ is used as a measure to denote the location of an image feature.

The ξ -varying Walsh coefficients are determined for any given row as follows:

consider the k^{th} row of an $N \times N$ image given by:

$$\text{row}_k = [x_{k0} \ x_{k1} \ x_{k2} \ \dots \ x_{k(N-1)}] \quad (9)$$

then, using an $n \times n$ Walsh kernel and conforming to formulation 5(c), the ξ -varying coefficients are determined as follows:

for $i = 0, 1, \dots, n-1$

$$a_i(\xi) = \sum_{j=0}^{n-1} \text{WAL}(i, j) \cdot x_{i(j+\xi)}, \quad \text{where } \xi = 0, 1, \dots, N-n \quad (10)$$

and $x_{i(j+\xi)}$ denote the elements of the given row.

The same results can be obtained using matrix representation of the row such that a direct matrix multiplication of the Walsh kernel by the row yields the desired results. In this case, we have the following representation:

$$\text{row}_k = \begin{bmatrix} x_{k0} & x_{k1} & \dots & x_{k(N-n)} \\ x_{k1} & x_{k2} & \dots & x_{k(N-n+1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k(n-1)} & x_{kn} & \dots & x_{k(N-1)} \end{bmatrix}$$

then,

$$\begin{aligned} A_k(\xi) &= [a_0(\xi) \ a_1(\xi) \ a_2(\xi) \ \dots \ a_{n-1}(\xi)]^T \\ &= \text{WAL}(i, j) \cdot \text{row}_k, \quad \xi = 0, 1, \dots, N-n \end{aligned} \quad (11)$$

where ξ is a given column of row_k

An illustration of these ξ -varying Walsh coefficients is given in Figure 4.

VI. THE STRING-TO-STRING MATCHING PROCESS

The overall string-to-string matching process consisted of the following steps:

- (1) Detect the features and identify their locations.
- (2) Determine the features with potential matches.
- (3) Search for global consistency to bring support to these potential matches.
- (4) Extract the disparity measure for matched features.

This process may sound complicated but is actually simple to implement with a very efficient execution time. These results are analyzed in section VII.

6.1 Feature Detection and Location Extraction

Image features are said to exist where noted peak values found in the first Walsh coefficient (first derivative) coincide with the zero-crossing found in the second coefficient (which is analogous to the second derivative). The variable ξ provides the location of the feature. An example of feature location extraction of a given row is given in Figure 5.

6.2 Stereo Matching Based on The Relational Amplitude Characteristics of Features

The following stereo matching approach adopts in a unique way the steps outlined in the beginning of section VI. The implementation consists of the following steps:

Step1: Determining a reference row to begin the feature search strategy: at a given k^{th} row of the stereo pair of images, determine which of row, in the left or in the right, has the least amount of features detected. Denote this row as the reference row. If both left and right rows have equal amounts of features, choose one of them arbitrarily.

Step2: Search for features with specific relational characteristics: Using the reference row, detect the feature location ξ_{max1} where the first derivative $a_1(\xi)$ has a peak of maximum value in absolute amplitude (such a feature is considered the most prevalent feature). Similarly, detect the feature location ξ_{max2} where the first derivative $a_1(\xi)$ has a peak of second largest value in absolute amplitude. Given these feature locations, establish the following strings for a later comparison:

$$\begin{aligned} A_{\text{ref}}(\xi_{\text{max1}}) &= [a_0(\xi_{\text{max1}}) \ a_1(\xi_{\text{max1}}) \ \dots \ a_{n-1}(\xi_{\text{max1}})] \\ A_{\text{ref}}(\xi_{\text{max2}}) &= [a_0(\xi_{\text{max2}}) \ a_1(\xi_{\text{max2}}) \ \dots \ a_{n-1}(\xi_{\text{max2}})] \end{aligned} \quad (12)$$

Step3: Search for a potential match: A potential match in the other row (non-reference row) of the stereo pair of rows is determined when the following relation holds:

$$A_{ref}(\xi_{\max \vartheta}) - B_{nref}(\xi_{p\vartheta}) = \min [A_{ref}(\xi_{\max \vartheta}) - B_{nref}(\xi_j)] \quad (13)$$

where $j = 1, 2, \dots, k_f$

where k_f denotes the number of features found in the non-reference row. For the *min* operation of the vector difference above, let

$$\psi(j) = A_{ref}(\xi_{\max \vartheta}) - B_{nref}(\xi_{p\vartheta}) = \sum_{i=0}^{n-1} |a_i(\xi_{\max \vartheta}) - b_i(\xi_j)| \quad (14)$$

where $j = 1, 2, \dots, k_f$

The potential feature match is that one which is found at location ξ_j to satisfy the relation:

$$\min [\psi(j), \quad j = 1, 2, \dots, k_f] \quad (15)$$

The variable ϑ is used as $\vartheta = 1$ when checking for a potential match of the feature found at location $\xi_{\max 1}$, and $\vartheta = 2$ when checking for a potential match of the feature found at location $\xi_{\max 2}$.

Step4: Verify if the consistency constraint is satisfied: The consistency constraint is based on the fact that if features are to match, it is expected that their disparities will be similar in the left and right images of the stereo pair. The following disparity measures are thus compared:

$$d_{ref} = |\xi_{\max 1} - \xi_{\max 2}|, \quad \text{and} \quad d_{nref} = |\xi_{p1} - \xi_{p2}| \quad (16)$$

If $d_{ref} \cong d_{nref}$ within an acceptable offset error ϵ due to camera distortion, the potential match is declared a correct match. When a correct match is found, the features matched will be removed from any subsequent matching in order to satisfy the uniqueness constraint. It is noted at this point that in the process of going back to *step 1* above for the next match, the former location $\xi_{\max 2}$ will now be found to be the new $\xi_{\max 1}$.

Step 5: Extract the disparity measure: The disparity measure extracted is a direct result of *step 4*. Thus, when consistency is found, the disparity d is simply

$$d = (d_{ref} \cong d_{nref})$$

VII. EXPERIMENTAL RESULTS

Two examples of the experimental results obtained using the Walsh-based matching approach described in this paper are shown in Figures 6 and 7. An analysis of the results is given in Table I. All the images used were of size 256x256 pixels. The results show that a high percentage of the features detected were matched. The execution time of the overall process for the many images considered ranged approximately from 4 to 15 seconds. This time was feature amount and filtering dependant. The disparity maps of the examples shown are displayed as white bars in the horizontal direction. It should be noted that the more features detected the more is the likelihood for mismatches. It is also worthy to note that the execution time is not proportional to the amount of features detected. The percentage increase in execution time is significantly less than the percentage increase in the amount of features detected.

Table I. Stereo Matching Results.

Image	σ	Points To Match	Points Matched	% Matched	Execution Time (sec)
Objects	1.5	2123	1947	91.71	14.82
Halfway	1.0	2136	1796	84.08	14.14

VIII. CONCLUSION

This paper establishes the merits of the application of Walsh coefficients toward a solution to the stereo vision problem. These coefficients are used as attribute functions of an incremental variable ξ not only to extract feature locations, but to bring these features to correspondence as well. The computational simplicity of the implementation along with the excellent results obtained make this approach a very attractive one in its application to the stereo vision problem. The required processing time of the overall computer implementation of this Walsh-based method depended of course on the complexity of the scene, that is on the amount of features to be matched. This execution time included the execution time of the Gaussian filtering algorithm.

It should be stated that the method presented here does not extend to treat the problem of surface reconstruction, but determines disparity at extracted feature locations. The contribution of this study is in the fact that by using the orthonormal and linearly-independent Walsh coefficients as incremental ξ -varying one-dimensional functions, both the uniqueness and the ordering constraints are integrated into a single matching process. Furthermore, since the first and second Walsh coefficients relate to the first and second derivative, they are also used in extracting exactly the features locations.

REFERENCES

- [1] D. Marr and T. Poggio, "A computational Theory of Human Stereo Vision", Proc. of the Royal Society of London, Vol. 204, pp. 301-328, 1979.
- [2] D. Marr, *Vision*, Freeman Publishing Co., San Francisco, California, 1982.
- [3] U. R. Dhond and J. K. Aggarwal, "Structure from Stereo - A Review", IEEE Trans. on Systems, Man, and Cybernetics, Vol. 19, No. 6, November/December, 1989.
- [4] W. E. L. Grimson, "An Implementation of a Computational Theory of Human Stereo Vision", Proceedings of the Image Understanding Workshop, College Park, Maryland, pp. 128-149, April 1980.
- [5] M. Kass, "Computing Visual Correspondence", Proceedings of the DARPA Image Understanding Workshop, pp 54-60, Arlington, Virginia, June 1983.
- [6] Y. Ohta, and T. Kanade, "Stereo by intra- and inter-Scanline Search Using Dynamic Programming", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 7, No. 2, pp. 139-154, March 1985.
- [7] M. Drumheller, and T. Poggio, "On Parallel Stereo", Proceedings of the IEEE, Robotics and Automation, 1986.
- [8] D. Nitzan, "Three-Dimensional Vision Structure for Robot Applications", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 10, No. 3, pp. 291-309, May 1988.
- [9] N. Ahmed and K. R. Rao, "Orthogonal Transforms for Digital Signal Processing", Springer-Verlag, New York, 1975.

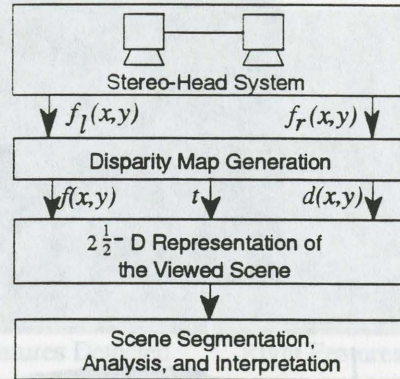


Figure 1. Scene understanding using 2 1/2 - D image representations.

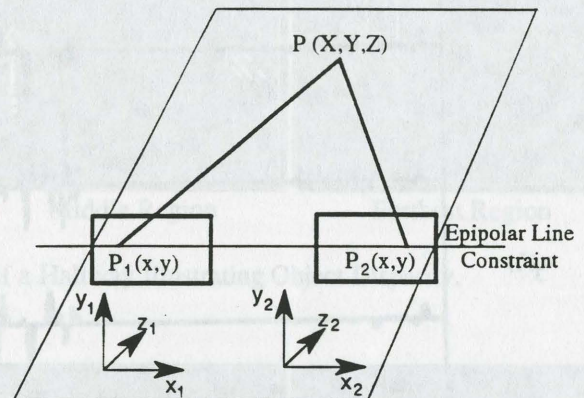


Figure 2. Image projection in stereo vision satisfying the epipolar geometry.

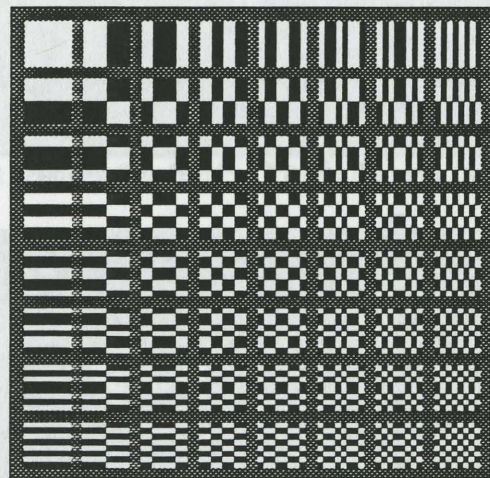
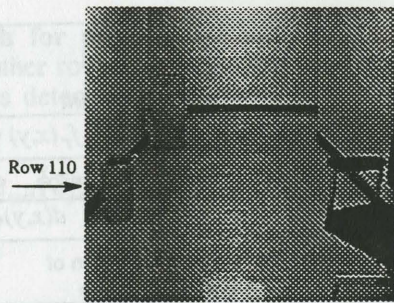
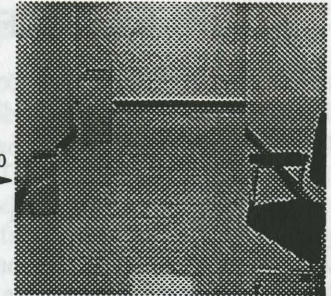


Figure 3. Ordered Walsh Basis Functions



Left Image



Right Image

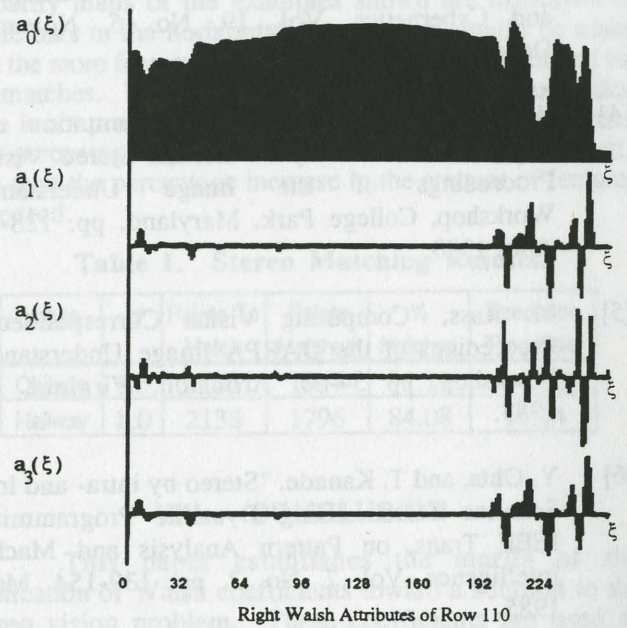
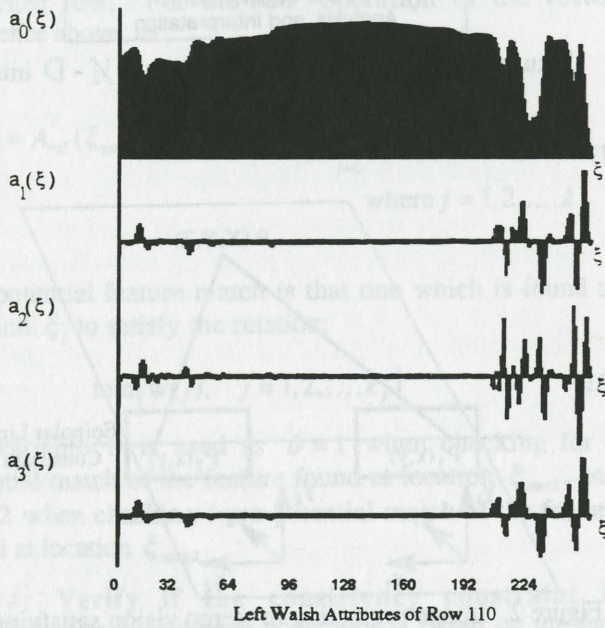


Figure 4. Walsh Attributes of a Given Row (row 110).

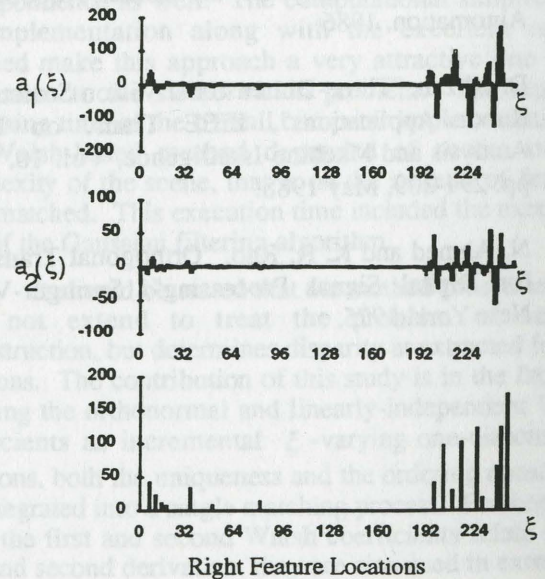
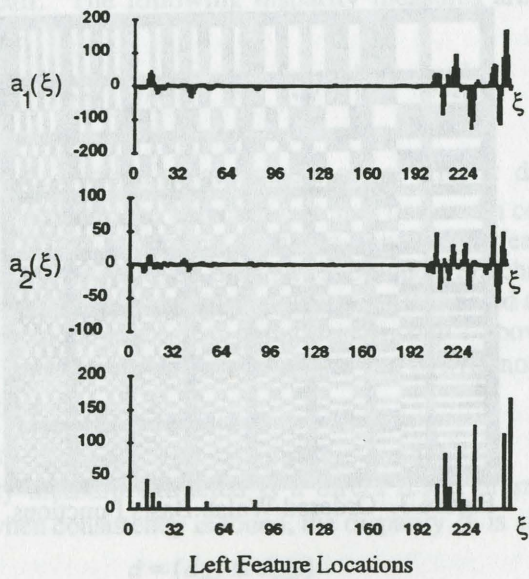


Figure 5. Feature Detection and Location Through the First and Second Walsh Coefficients of Above Images.

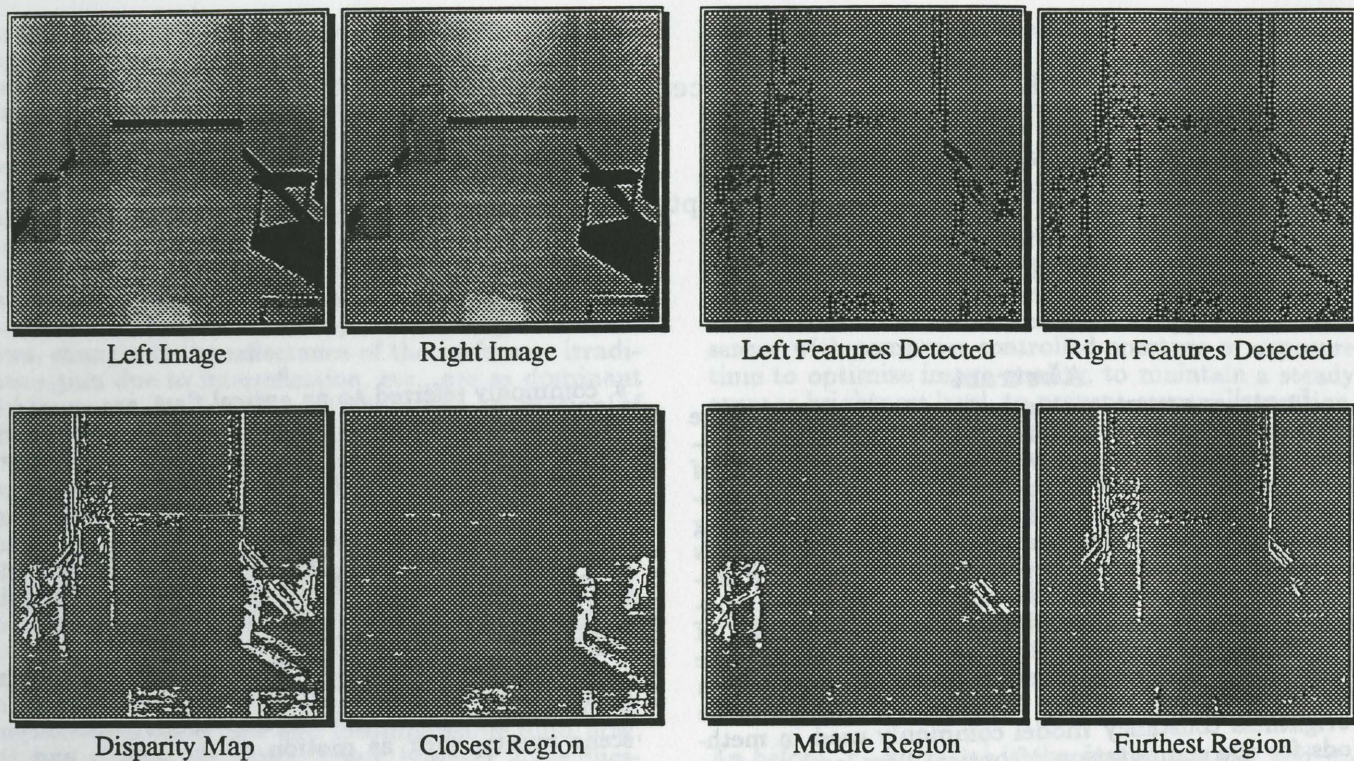


Figure 6. Scene Disparity and Segmented Regions of a Hallway Illustrating Object Disparity.

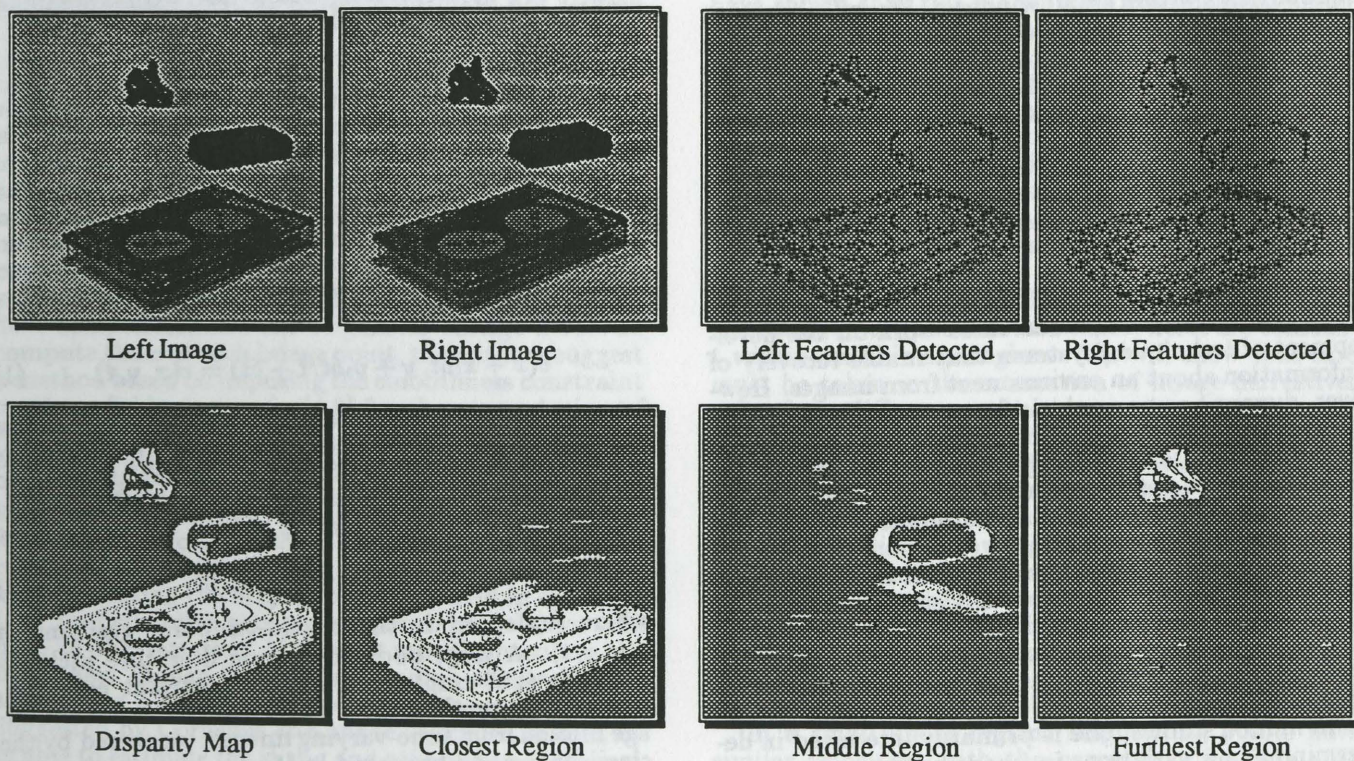


Figure 7. Scene Disparity of Objects and Segmented Regions Illustrating the Objects' Disparity.

Computing Optical Flow for Scenes with Time-Varying Brightness *

Shahriar Negahdaripour
Electrical & Computer Engineering Dept.
University of Miami
Coral Gables, FL 33124

Chih-Ho Yu
Harbor Branch Oceanographic Inst.
5600 U.S. 1 North
Fort Pierce, FL 34946

Abstract

In earlier papers, we proposed a generalized image motion constraint equation for the computation of optical flow in applications where the image brightness of a scene point is allowed to vary with time due to illumination non-uniformity, light source motion, shading effects, specular reflection, interreflection, etc. This constraint equation is derived from a differential image brightness model which allows linear transformation, described by a multiplier and an offset field, of the brightness of a scene point from one instant of time to the next. In a range of experiments, we have shown the improved performance of our model relative to the brightness constancy model commonly used in methods for the computation of optical flow, and another model proposed for computing optical flow for shaded scenes. In this paper, we investigate some inherent properties of the model in an implementation which assumes the constancy of image motion and transformation fields within small windows. We will show that care must be taken in the estimation of image derivatives using finite difference methods to avoid a bias in the solution due to events that contribute to brightness variations in the form of a multiplier field. We suggest a simple modification to overcome this problem. We also show how a bias in the brightness transformation fields can be removed. Experiments with real images are presented to evaluate the performance of our method.

1 Introduction

In computer vision, researchers work on the development of techniques/systems that enable recovery of information about an environment from images. However, dynamic scene analysis for uncontrolled environments is a difficult task since many events take place simultaneously, each having its impact on the image. Some examples are variations in the geometric and/or radiometric properties of the illumination source and the sensor, of the scene object (location, orientation, and/or reflectance properties), as well as the medium characteristics. Most computer vision researchers agree that the development of a model, of how important scene events in a particular environment affect the image, is necessary to recover accurate quantitative information about the scene.

In motion studies, one is primarily interested in determining how variations in image brightness pattern-

*Funding for this research was provided by NSF under grants IRI-8910967 and BCS-9017179.

s, commonly referred to as *optical flow*, are caused by the motion of an object (or parts of, or points on, the object) relative to the imaging device. Many methods for solving this problem have been proposed, categorized as differential, matching, energy-based, and phase-based methods (e.g., see [1] for a description and comparison of some of these methods). This paper is mainly concerned with the analysis of a differential method for the computation of image motion based on a model proposed earlier by the authors.

In addition to the image flow, we compute brightness variation of points on the surfaces of the scene which encode useful information about time-varying scene events, such as motion of the source and the camera, the surface reflectance properties, etc. The reader should take special note of the fact that the model is not restricted to the differential formulation investigated in this paper. In fact, a correlation-based approach is being investigated to deal with large image displacements, which will be reported elsewhere.

2 Background

Past research work on the computation of optic flow from time-varying imagery has mainly dealt with scenarios where optic flow is dominantly due to motion effects. In doing so, it has been typically assumed that, in the absence of other scene events, the brightness of a surface patch remains relatively constant as the surface moves in the environment relative to the viewer. This is the so-called *brightness constancy assumption*, which can be written

$$e(x + x_t \delta t, y + y_t \delta t, t + \delta t) = e(x, y, t), \quad (1)$$

where $e(x, y, t)$ is the image brightness function of a point $[x, y]$ at time t , and $r_t = [x_t, y_t]$ is the image velocity of r (optic flow) from one instant of time t to the next, $t + \delta t$. A first-order approximation of this model results in the classical optical flow constraint equation:

$$e_x x_t + e_y y_t + e_t = 0, \quad (2)$$

where $[e_x, e_y, e_t]$ is the spatio-temporal gradient of $e(x, y, t)$. A large body of motion literature includes methods that employ the above constraint equation, in one form or the other, to recover the scene or image motion from time-varying images, pioneered by the classical method proposed in [4].

Conceptually, solving the optic flow problem using the brightness constancy model is equivalent to tracking the images of the markings on the surface, which

are expected to maintain their brightness values in image sequences. The model is exact where the scene surface is Lambertian, and is either stationary as the camera moves, or its motion is dominantly translational parallel to the image plane. For other motions, the model seems to be a good approximation where the scene surface has rich texture or markings, such that brightness changes due to shading or surface foreshortening effects are negligible compared to those due to motion effects (e.g., see results in [15]). It is not a good model in applications where brightness variations due to illumination non-uniformity and shading, cast shadows, changes in the reflectance of the surface or irradiance gain due to interreflection, etc., are as dominant as (or more dominant than) those due to motion of the camera relative to the scene. Typically, the contribution due to these events become more significant in regions of the image that correspond to surface patches lacking strong texture or markings.

Some researchers have proposed dynamic image models which allow relaxation of the brightness constancy assumption (e.g., [3,7,14]). Essentially, these authors propose new image flow equations that are expected to take into account variations in the surface irradiance and/or radiance due to various scene events. Schunck's model [14] is derived from an analogy between optic flow and incompressible fluid flow, studied in fluid mechanics. Many, however, have questioned the validity of such an analogy, and furthermore no experimental results have been given for its support. The models proposed in [3] and [7] are essentially the same. Cornelius & Kanade [3] suggest that the brightness of an object point need not remain the same from one sampling time to the next. To accommodate this, they introduce an "image brightness change field," de/dt , which can be nonzero:

$$\frac{de}{dt} = e_x x_t + e_y y_t + e_t, \quad (3)$$

and is expected to take into account any variations in the brightness of a surface patch from one image to the next. The addition of the new term results in three unknowns at each image point, the two image motion field components and the brightness change field. To compute these at each image point, the authors suggest a method based on imposing the smoothness constraint on these fields, just as in [4,9] for the computation of the optical flow components based on the brightness constancy assumption.

Mukawa [7] starts with the Phong's shading model to arrive at a constraint equation for the image flow induced by the motion of shaded surfaces. His derivation results in the additional of a luminance difference term, representing the brightness change field in equation (3). To distinguish between these two models, we take note of the fact that Mukawa's model describes the brightness change in terms of shading effects, rather than being a simple correction term, as in the model suggested by Cornelius & Kanade. Because of this, he is able to estimate the shape and reflectance coefficient of the surface, as well as the illumination direction, in addition to image motion (see [8]).

We have proposed methods for the computation

of optic flow based on a generalized differential image brightness model [10,11]:

$$e(x+x_t \delta t, y+y_t \delta t, t+\delta t) = M(x, y, t)e(x, y, t) + C(x, y, t), \quad (4)$$

which incorporates effects from events which change the image brightness of a point through a linear transformation. For example, if image brightness is modeled as the product of an illumination field and the surface reflectance field, changes in these from one time instance to the next can be captured by the multiplier field M . Other examples include applications of a sensor with computer-controlled aperture or exposure time to optimize image quality, to maintain a steady average brightness level, to prevent camera saturation, or to increase signal-to-noise ratio when illumination drops. In comparison, variations in irradiance gain due to interreflection or specular reflection, as well as saturation of the sensor in the dark or bright regions of the image due to large increase or decrease in illumination level may be modeled through the offset term, C .

In [11], we derived the first-order Taylor series expansion of the model to arrive at a constraint equation for the computation of differential image flow:

$$e_t + e_x x_t + e_y y_t - e m_t - c_t = 0. \quad (5)$$

As before, $r_t = [x_t, y_t]$ is the image motion, and m_t and c_t are temporal variations from one and zero, respectively, of the multiplier and offset fields, associated with the brightness change of a particular scene point from one frame to the next. In another paper [12], we have shown that this model gives considerably better estimates of image motion than the brightness constancy model (e.g. [4]), and improved results compared to Mukawa's shading model, using a range of experiments with real scenes. This included a comparison of the results with and without motion between frames under light source motion and varying intensity, specular reflection, inter-reflection, illumination attenuation in underwater imagery, etc.

The objective of this paper is to investigate some inherent properties of the solution from a particular algorithm, assuming that the optic flow and the transformation (multiplicative and offset) fields are constant within small image regions. We will show that care must be taken in the estimation of image derivatives using finite difference methods to avoid a bias in the solution induced by multiplicative scene brightness variations. We suggest a simple modification to overcome this problem. We also show how a bias in the brightness transformation fields can be removed. This is useful in applications where the transformation fields encode information about the light source motion or intensity change, surface reflectance properties, etc. Experiments with synthetic data and real images are presented to evaluate the performance of our method.

3 Least-Square Solution

In a small image region, neighboring points have similar velocities, with the exception of points near motion or depth discontinuity boundaries. Similarly, the transformation fields vary slowly in most regions of the image, except near boundaries of illumination or

reflectance discontinuity, cast shadows, etc. It can be assumed that the optic flow and transformation fields are (approximately) constant within small image regions, which we take to be square windows of size $n \times n$ centered at each image point, for convenience. A least-squares formulation is then employed to estimate the four fields x_t , y_t , m_t , and c_t at the center of the square ¹.

Minimizing the sum squared-error in the constraint equation over a $n \times n$ image region (square region is used for convenience), it is easy to show that the solution is obtained from:

$$\sum_{i=1}^n \sum_{j=1}^n \begin{bmatrix} e_x^2 & e_x e_y & -e_x e & -e_x \\ e_x e_y & e_y^2 & -e_y e & -e_y \\ -e_x e & -e_y e & e^2 & e \\ -e_x & -e_y & e & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ m_t \\ c_t \end{bmatrix} = \sum_{i=1}^n \sum_{j=1}^n \begin{bmatrix} -e_x e_t \\ -e_y e_t \\ e e_t \\ e_t \end{bmatrix} \quad (6)$$

Provided that the 4×4 matrix is not singular, a solution can be derived in closed form. The computation is repeated for each image point. The solution of equation (6), as well as most other gradient-based method for computing image motion, requires the estimation of image gradients using finite difference methods. In our method, special attention should be paid to choosing the appropriate mask. More precisely, a symmetric mask should be used to avoid certain biases in the solution, as we will show next.

3.1 Discrete Gradients

Let $e(i, j, k)$ denote the image function in the discrete domain. The correspondence between (i, j, k) and (x, y, t) is given by $x = i\delta x$, $y = j\delta y$, and $t = k\delta t$, where $\{\delta x, \delta y\}$ and δt are the spatial and temporal sampling intervals. Without loss of generality, we will assume $\delta x = \delta y = \delta t = 1$.

A simple mask to estimate derivatives is the first difference operator. Horn [5] suggests the use of $2 \times 2 \times 2$ triplets (Chapter 12). For example,

$$e_x \approx \frac{\begin{bmatrix} e(i+1, j, k) - e(i, j, k) + \\ e(i+1, j+1, k) - e(i, j+1, k) + \\ e(i+1, j, k+1) - e(i, j, k+1) + \\ e(i+1, j+1, k+1) - e(i, j+1, k+1) \end{bmatrix}}{4} \quad (7)$$

Similar expressions can be written for e_y and e_t .

To simplify analysis involving gradient expressions, we will use a notation equivalent to the Z -transform

¹Parametric models have commonly been used in various vision problems to formulate a well-posed problem (e.g., [2,6,13]), as in the optic flow computation to overcome the aperture problem. Higher-order models may be assumed, however, for a small enough region, a zeroth-order (constant) model is typically sufficient for most regions of the image. This corresponds to computing an average flow and transformation fields in a local neighborhood and tends to smooth the results. While it may not give an acceptable solution over motion and depth discontinuity boundaries, it typically gives more robust results in most other regions of the image, particularly in regions of weak texture.

operations. If we let $e(i, j, k) \rightarrow E$, we then have

$$e(i+1, j, k) \rightarrow \alpha E, \quad e(i, j+1, k) \rightarrow \beta E, \quad e(i, j, k+1) \rightarrow \gamma E,$$

where α , β , and γ are shifting operators. Similarly, we have $e(i-1, j, k) \rightarrow \alpha^{-1}E$ for reverse, as well as $e(i+1, j+1, k) \rightarrow \alpha\beta E$ for cascaded shifting operations.

Using our notation, we have

$$e_x \rightarrow \frac{\left[\begin{array}{l} (\alpha E - E) + (\alpha\beta E - \beta E) + \\ (\alpha\gamma E - \gamma E) + (\alpha\beta\gamma E - \beta\gamma E) \end{array} \right]}{4} = (1 + \beta)(1 + \gamma)(\alpha E - E)/4. \quad (8)$$

Similarly, we obtain

$$e_y \rightarrow (1 + \alpha)(1 + \gamma)(\beta E - E)/4, \quad (9)$$

$$e_t \rightarrow (1 + \alpha)(1 + \beta)(\gamma E - E)/4. \quad (10)$$

3.2 Modeling of Image Brightness

The coefficients in equation (6) depend on the particular image being processed. To determine the general behavior of the solution, we need to consider a "representative" image. Alternatively, we may assume a statistical model for the image function, and approximate the necessary expressions using their expectations.

We assume brightness e to be a random variable with mean M . To describe its distribution, we model it as a two-dimensional first-order stationary Markov process with the expectations $\mathcal{E}(E, E) = \mathcal{E}(E^2) = R_0^2$, $\mathcal{E}(E, \alpha E) = R_1^2$, and $\mathcal{E}(E, \alpha\beta E) = R_2^2$. Here, R_0^2 is the auto-correlation of the brightness, R_1^2 is the cross-correlation of two adjacent pixels in 4-connectedness configuration, and R_2^2 is the correlation of two diagonally-connected adjacent pixels. The brightness values of non-adjacent pixels are uncorrelated. Therefore, we have, for example, $\mathcal{E}(\alpha^{-1}E, \alpha E) = \mathcal{E}(\beta^{-1}E, \beta E) = [\mathcal{E}(E)]^2 = M^2$. We may assume higher-order and/or other statistical models. Alternatively, we may assume that the brightness of two adjacent pixels are independent (therefore, $R_1^2 = R_2^2 = M^2$). However, these will not be necessary, as the assumed model is sufficient to demonstrate our point.

The coefficients in equation (6) are the sum of certain moments of image gradient within image area under consideration, assumed to be a square window of size $n \times n$ centered at each pixel. If we make $N = n^2$ large enough in the statistical sense, we can approximate these coefficients using the expectation of the corresponding moments. Based on this, we have

$$\left(\mathcal{E} \left[\begin{array}{cccc} e_x^2 & e_x e_y & -e_x e & -e_x \\ e_x e_y & e_y^2 & -e_y e & -e_y \\ -e_x e & -e_y e & e^2 & e \\ -e_x & -e_y & e & 1 \end{array} \right] \right) \begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{m}_t \\ \hat{c}_t \end{bmatrix} = \mathcal{E} \begin{bmatrix} -e_x e_t \\ -e_y e_t \\ e e_t \\ e_t \end{bmatrix} \quad (12)$$

Here, \hat{x}_t , \hat{y}_t , \hat{m}_t , and \hat{c}_t denote the expected values of the sought after unknowns. We will use x_t , y_t , m_t and

c_t to denote the true values. Calculation of the expectations in equation (12) using the assumed statistical brightness model is a tedious exercise in algebra. We consider two cases, next.

3.2.1 Case 1: No motion

Let us assume that there is no motion between frame k and $k + 1$, and that the only brightness variations are due to the transformation fields. If the algorithm is unbiased, we should obtain $\hat{x}_t = x_t = 0$, $\hat{y}_t = y_t = 0$, $\hat{m}_t = m_t$, and $\hat{c}_t = c_t$. We will see that the estimated parameters are biased when the $2 \times 2 \times 2$ templets are used in the discrete computation of image gradient.

We have

$$\mathcal{E}(e_x) = \mathcal{E}(e_y) = 0.$$

The summations in equation (6) are performed over an image area, not over a time interval. Therefore, we need only the statistics of brightness (and various derivatives) over an image area, not a time period. Statistics of terms involving time derivative can be obtained using the optical flow constraint equation. Using our brightness change model with $x_t = y_t = 0$, we have

$$e_t - Em_t - c_t = 0.$$

Therefore, we obtain

$$\mathcal{E}(e_t) = m_t M + c_t. \quad (13)$$

It can be readily shown that $\mathcal{E}(e_x e_y) = 0$. Evaluating the expectations of other terms, we obtain:

$$M \hat{x} = d, \quad (14)$$

where $\hat{x} = (\hat{x}_t, \hat{y}_t, \hat{m}_t, \hat{c}_t)^T$,

$$M = \begin{bmatrix} m_{11} & 0 & m_{13} & 0 \\ 0 & m_{22} & m_{23} & 0 \\ m_{31} & m_{32} & R_o^2 & M \\ 0 & 0 & M & 1 \end{bmatrix}, \quad (15a)$$

$$d = \begin{bmatrix} 0 \\ 0 \\ m_t(R_o^2 + 2R_1^2 + R_2^2)/4 + c_t M \\ m_t M + c_t \end{bmatrix}. \quad (15b)$$

and

$$m_{11} = m_{22} = (2 + m_t)^2 (R_o^2 - R_2^2)/4,$$

$$m_{13} = m_{31} = m_{23} = m_{32} = (2 + m_t)(R_o^2 - R_2^2)/4.$$

We know that $x_t = y_t = 0$, but the first two equations give $\hat{x}_t = \hat{y}_t = -\hat{m}_t/(2 + m_t)$. Furthermore, the transformation fields are:

$$\hat{m}_t = f m_t, \quad \text{and} \quad \hat{c}_t = (m_t - \hat{m}_t)M + c_t, \quad (16)$$

where

$$f = \frac{(R_o^2 + 2R_1^2 + R_2^2 - 4M^2)}{2(R_o^2 + R_2^2 - 2M^2)}.$$

We see that the estimated flow \hat{x}_t and \hat{y}_t are erroneous as long as $m_t \neq 0$. The estimated multiplier field is

not correct unless $R_o^2 = R_1^2 = R_2^2$. Also, \hat{c}_t will not be correct unless $\hat{m}_t = m_t$. We note that the source of the bias is the multiplier field. In the absence of brightness changes due to events that are modeled as a brightness scaling, there would be no such bias. For example, there is no bias due to an event that contributes an offset in brightness from one frame to the next. This is not an issue in methods based on the brightness constancy model, where the transformation field are assumed to be zero. Needless to say, the optical flow estimates from these methods will be erroneous where brightness variations due to some scene events are significant.

The erroneous results are caused by phase shifting in discrete gradient calculations. The biases in the optic flow may be eliminated using a symmetric mask, however, the errors in the transformation fields remain, which will be hard to remove. A simple correction which resolves both problems is to use the average brightness \bar{e} over the same $2 \times 2 \times 2$ support region of the gradient templet cubic block, in the least-square solution in equation (6), as we show next².

For the average brightness, we have $\bar{e} \rightarrow (1 + \gamma)(1 + \beta)(1 + \alpha)E/8$. Replacing E with \bar{e} in equation (12), and recomputing the expectations of each coefficient, we obtain

$$M_m = \begin{bmatrix} m_{11} & 0 & 0 & 0 \\ 0 & m_{22} & 0 & 0 \\ 0 & 0 & m_{33} & m_{34} \\ 0 & 0 & m_{43} & 1 \end{bmatrix}, \quad d_m = \begin{bmatrix} 0 \\ 0 \\ d_3 \\ d_4 \end{bmatrix}, \quad (17)$$

where

$$m_{11} = m_{22} = (2 + m_t)^2 (R_o^2 - R_2^2)/4,$$

$$m_{33} = (2 + m_t)^2 (R_o^2 + 2R_1^2 + R_2^2)/16,$$

$$m_{34} = m_{43} = (2 + m_t)M/2,$$

$$d_3 = m_t(2 + m_t)(R_o^2 + 2R_1^2 + R_2^2)/8 + c_t(2 + m_t)M/2,$$

$$d_4 = m_t M + c_t.$$

Solving these equations, we obtain:

$$\hat{x}_t = \hat{y}_t = 0, \quad \hat{m}_t = \frac{2m_t}{2 + m_t} \quad \text{and} \quad \hat{c}_t = c_t. \quad (18)$$

The recovered optical flow and the offset fields (actually, their expected values) are now correct. While the recovered multiplier is not correct, the true value can be computed by a simple recalculation $m_t = 2\hat{m}_t/(2 - \hat{m}_t)$. The estimated multiplier m_t from equations (16) or (18) is biased, however a significant difference should be emphasized. The earlier equation depends on the statistics of the image which needs to be estimated from the data. In the latter equation, the dependency is only on the actual m_t . Hence, a correction can be made readily, as above.

²The is equivalent to making the center of the templet as the pixel location, hence introducing the necessary symmetry which results in removing the phase shifting. The distinction of whether the pixel location is at the center of the cubic block or at the lower bottom corner is irrelevant if the optic flow constraint equation does not involve the pixel location or its brightness, as in the brightness constancy model.

3.2.2 Case 2: Motion Recovery

We now consider the interesting case where image motion is nonzero, and investigate the error in the estimated motion and transformation fields. We consider a one-pixel motion in x direction, that is, $x_t = 1$. The motion in y -direction can be studied similarly, and can be combined with the analysis for the x motion to study any arbitrary image motion. The one-pixel assumption is necessary to obtain meaningful results from spatio-temporal image gradients, because of the assumed statistical model of the image function. For slowly varying brightness functions (and band-limited images), larger motions are allowed subject to the sampling theorem. We further assume that $c_t = 0$, without loss of generality, since bias in the solution is a result of the multiplier field. Therefore, we have

$$e(i, j, k + 1) = (1 + m_t)e(i - 1, j, k),$$

which yields

$$\gamma E = (1 + m_t)\alpha^{-1} E.$$

Repeating the steps we followed in the previous case, we arrive at:

$$M_m x_m = d_m,$$

where

$$M_m = \begin{bmatrix} m_{11} & 0 & 0 & 0 \\ 0 & m_{22} & 0 & 0 \\ 0 & 0 & m_{33} & m_{34} \\ 0 & 0 & m_{43} & 1 \end{bmatrix}, \quad d_m = \begin{bmatrix} d_1 \\ 0 \\ d_3 \\ d_4 \end{bmatrix}. \quad (20)$$

We have

$$m_{11} = \left[\frac{(1 + m_t + m_t^2)R_o^2 + (1 + m_t)R_1^2 - m_t^2 R_2^2 - 2(1 + m_t)M^2}{(1 + m_t)R_1^2 - m_t^2 R_2^2 - 2(1 + m_t)M^2} \right] / 4,$$

$$m_{22} = [(3 + 3m_t + m_t^2)R_o^2 + (1 + m_t)R_1^2 - (2 + m_t)^2 R_2^2] / 4,$$

$$m_{33} = \left[\frac{(3 + 3m_t + m_t^2)R_o^2 + (7 + 7m_t + 2m_t^2)R_1^2}{(4 + 4m_t + m_t^2)R_2^2 + 2(1 + m_t)M^2} \right] / 16,$$

$$m_{34} = m_{43} = (2 + m_t)M / 2,$$

$$d_1 = -(1 + m_t)(R_o^2 + R_1^2 - 2M^2) / 4,$$

$$d_3 = m_t(2 + m_t)(R_o^2 + 2R_1^2 + R_2^2) / 8, \quad \text{and} \quad d_4 = m_t M.$$

Solving these equations, we obtain

$$\hat{x}_t =$$

$$\frac{(1 + m_t)(R_o^2 + R_1^2 - 2M^2)}{(1 + m_t + m_t^2)R_o^2 + (1 + m_t)R_1^2 - m_t^2 R_2^2 - 2(1 + m_t)M^2}, \quad (21)$$

and $\hat{y}_t = 0$. Equation (21) shows that there is an error in the the estimated flow. When the multiplier is relatively small, this error is negligible, and can be approximates as

$$|\text{error}| = \left(\frac{R_o^2 - R_2^2}{R_o^2 + R_1^2 - 2M^2} \right) m_t^2.$$

As we see through examples of various experiments, the new error in the "modified" algorithm is much smaller than the bias in our "non-modified" method (using the average brightness within the cubic templet instead of the original formulation). We are currently investigating other corrections to remove the bias or any other error totally.

4 Experiment Results

In earlier papers [11,12], we have presented results of a variety of experiments to demonstrate the performance of the proposed model in comparison to other models. In this paper, we restrict attention to experiments, which we have conducted to support the analytical results presented in this paper. In particular, we investigate the performance of the modified algorithm in a variety of experiments with synthetic and real images.

Some experiments involve brightness variations in an image sequence due to illumination change, change of camera aperture, or exposure time, without any motion. A robust optical flow method should not only give accurate estimates of image motion, when these exist, but also should produce no motion, when the image brightness variations are due to events other than motion. Other experiments involve brightness variations due to both motion and illumination effects.

4.1 Sand Surface; no motion

Figure 1a shows the image of a sandy surface, and figure 1b shows the histogram of the brightness values (a Photometrics camera with 12-bit gray level resolution was used). The second image was generated synthetically by doubling the brightness of the first image. There is no motion from one frame to the next. Therefore, we have $x_t = y_t = c_t = 0$ and $m_t = 1$.

Figure 1c-d show the histogram of the estimated flow components over the whole image using the non-modified method. As explained, this is an erroneous flow as a result of the brightness variation due to the synthesized multiplier field. Figure 1e-f shows the flow determined from the modified method. The improvement in the results is immediately noticeable, as the bias has been eliminated. Table 1 is a summary of the results based on the average of the flow and the transformation fields over the whole image. The average recovered optic flow from the (non-modified) algorithm is $\hat{x}_t = -0.276$ and $\hat{y}_t = -0.273$.

To determine the expected flow based on our theoretical analysis, we estimated the average brightness of the first image, $M = 155.8$ (also see the brightness histogram in figure 1b), and the auto- and cross-correlations $R_o^2 = 2.469 \times 10^4$, $R_1^2 = 2.448 \times 10^4$ and $R_2^2 = 2.441 \times 10^4$. Using equations (16), we obtain $\hat{m}_t = 0.873$ and $\hat{x}_t = \hat{y}_t = -0.28$. The estimates based on the modified solution (17) is $\hat{x}_t = -0.26 \times 10^{-6}$, $\hat{y}_t = -0.56 \times 10^{-7}$, $\hat{m}_t = 1.00$, and $c_t = -.054$. We note that both estimates, from the non-modified and modified methods, are consistent with the theoretical results.

We have verified that the bias in the flow does not occur in the presence of an offset field, by increasing the brightness of every point in the sand image by 100

gray levels. The optic flow was zero, as expected (not shown here due to lack of space).

4.2 Office Ceiling; no motion

We repeated the previous experiment with a sequence of real images of an office ceiling. After the first image (figure 2a) was taken, the camera aperture was increased to take the second image (figure 2b). Again, there is no motion between the two images. That is, the true optic flow is $x_t = y_t = 0$. The histograms of the two images are shown in figures 2c-d. As we expect, they have a very similar shape. The histograms of m_t and c_t , based on a linear regression model, are shown in figures 2e-f. The average values of the transformation fields over the whole image are $m_t = 0.492$ and $c_t = -7.62$. The average brightness change due to the multiplier field is $0.492 \times 979.2 \approx 482$ gray levels, while that due to the offset field is only -7.6 gray levels (here, $M = 979.2$ is the average brightness of the first image). This confirms our expectation that an aperture change causes brightness variations dominantly due to a multiplicative field.

Figure 2e-f show the histograms of the components of the estimated flow based on the non-modified method, and figure 2g-h shows the same for the modified method. Average results have been tabulated in Table 2. We note that the average flow from the non-modified algorithm is $\hat{x}_t = -0.156$ and $\hat{y}_t = -0.135$. This is an erroneous flow as a result of the brightness variation due to the multiplier field. The estimate based on the theoretical analysis using average brightness $M = 979.2$, average auto-correlation $R_0^2 = 1.04 \times 10^6$, and cross-correlations $R_1^2 = 1.02 \times 10^6$ and $R_2^2 = 1.00 \times 10^6$, is $\hat{x}_t = \hat{y}_t = -0.185$. While there is some discrepancy between these two estimates, we should remind the reader that the latter is determined from equations derived from an assumed statistical image model, which we used only to demonstrate the existence of the bias. The true value of the bias can vary from image to image based on the texture contents (gray-level variations and statistics). The estimates from the modified method are $\hat{x}_t = 0.016$ and $\hat{y}_t = 0.037$.

Similarly, the transformation fields were computed to be $\hat{m}_t = 0.5$ and $\hat{c}_t = -14.7$. It should be noted that c_t is typically a few orders of magnitude larger than other three parameters (x_t , y_t , and m_t). When average image brightness is about 1,000 or more (12-bit image), the error in estimated c_t ($\delta c_t = 7.08$) is less than one percent. Also note that the error in average brightness change due to the offset term has been compensated by an approximately equal, but opposite in sign, error in the average multiplier ($\delta m_t = 0.008$), scaled by the average image brightness $M = 979.2$. This can be easily explained, since the effect of an offset on image brightness variation of a point is equal to that of a multiplier scaled by the brightness value at that point. Over a region, this holds approximately true if the actual values are replaced by the average values over that region. Consequently, while there is some error in the transformation fields, the total brightness change $m_t E + c_t$, as well as the optical flow, are computed with good accuracy.

4.3 Varying Texture; no motion

The purpose of this experiment was to investigate the variations in the bias flow due to the scene texture content, and in the solution after the modified method is used. A plane with four different textured areas was imaged (see figure 3). The first area, the upper left, is a piece of blue cloth with white dots. The second area, the lower left, is a random drawing. The third area, the upper right, is a piece of paper with printed letters. The fourth area, the lower right, is a patch of carpet with fine texture.

Two images were taken, with the 12-bit Photometrics camera, without motion, but the exposure time was changed. The exposure time was 0.2 and 0.1 second for the first and second image, respectively. Therefore, the first image is brighter than the second. The brightness histogram of the four areas are shown in figure 4a-h. Figure 4a-b show the brightness histogram of the first area in the first and second images, figure 4c-d shows the same for the second areas, and so on. The differences in the distributions for the four regions can be seen.

The results of the experiment are shown in Table 3. The "true" transformation fields were estimated by a linear regression model applied to the brightness values in the first and second images. For the four areas, the computed multiplier m_t were -0.470 , -0.475 , -0.475 , and -0.468 . The offset term c_t was 12.2, 11.9, 12.3, and 12.1, for the four areas.

The biases in the estimated flow components from the non-modified method are about 0.28 pixel. The recovered multiplier field \hat{m}_t varies from -0.377 to -0.452 , mainly due to different texture contents (remember the dependency of the solution of the multiplier field on the brightness statistics, for example, the auto and cross correlations in our first-order model). When the modified method is used, the recovered flow is near zero as it should be. The recovered multiplier field, \hat{m}_t , for the four regions are consistent (varying from -0.641 to -0.646), however biased as explained earlier. The corrected m_t values were computed from $\hat{m}_t / (2 - \hat{m}_t)$. These are more consistent with the estimates from the linear regression model. Part of the error in the multiplier field is compensated by the error in the offset term, such that the total error in the brightness change, from reduced exposure time, is reduced. In other words, while we have over-estimated the brightness reduction due to the multiplier field we have over-estimated the brightness gain due to the offset term. For example, the brightness change error, for region one, due to the multiplier effect is $-(0.470 - 0.486) \times 671 = 10.7$ gray levels, and due to the offset term is $(12.2 - 30.4) = -18.24$ gray levels. The total error is only -7.5 gray-levels, compared to an average e_t value of -303 , or compared to an average brightness value of 671 in the first image.

4.4 Sand Surface; motion

This experiment involves the motion of the camera platform relative to the sand surface, illuminated by a point source. After taking image 1, the camera was moved along the x -direction by approximately 0.45 pixels. Furthermore, several images were taken at the second position, each after varying, both increasing

and decreasing, the light source intensity by different amounts (we refer to these as image 2. i , $i = 1, 2, \dots, 6$, which are not shown due to lack of space). The results of this experiment are given in Table 4. The first two columns correspond to image sequences where the illumination was decreased (that is, lower illumination from image 1 to images 2.1 and 2.2). The third column corresponds to a case where the illumination was steady over both frames (same illumination from 1 to 2.3). The last three columns involves image sequences where the illumination was increased (from 1 to 2. i , $i = 4, 5, 6$). These results have been tabulated in the order of increasing illumination, in the first to the last column.

A linear regression model was used to determine the average multiplier and offset fields for each sequence. (To do this, we compared the brightness of image 2. i , $i = 1, 2, \dots, 6$ with image 2.3. Since there is no relative motion between these images, the brightness change is due to change in the illumination).

Figure 5 shows the recovered x_t for the two methods -vs- the multiplier field. These results show that the brightness change due to varying illumination has some effect on the recovered optical flow x_t when the modified method is used, but it significantly affects the estimate from the non-modified method. For the non-modified method, there is an obvious pattern where the motion is under-estimated for decreasing illumination, and over-estimated for increasing illumination. From Table 4, the behavior of the estimated y_t (expected true y_t is zero) is less clear, however less variations occur when the modified method is used.

4.5 Sand Surface; synthesized motion

Earlier analytical results showed that even with the modified algorithm, there is a error in the optical flow estimate proportional to the square of the multiplier field, when the multiplier field is small. In this experiment, we investigate this error.

We use the sand image of experiment 1. The brightness values are then multiplied by $(1 + m_t)$, where m_t is varies from -0.4 to 0.4 . We then shift the image by one pixel in the x -direction; that is $x_t = 1$.

Both the non-modified and modified methods were used to estimate the image motion. Table 5 shows the results. It can be seen that the error in x_t is less than 3% for $-0.2 \leq m_t \leq 0.2$. Further, figure 4 shows that the estimates from the modified method are in good agreement with the analytical solution of equation (21). We should again emphasize that the analytical solution was derived based on the assumed statistical model of the spatial brightness variations in the image, which may or may not agree with the actual image data in a given situation.

5 Summary

We had proposed a new model for the computation of motion from time-varying imagery in the presence of brightness variations of points on the surface of the scene due to scene events (shading and shadow effects, surface reflectance changes due to object motion, etc.), illumination effects (change in light source position or intensity), or camera setups (change in aperture or exposure time). Many of these are events that

contribute to the variations of brightness in images of natural uncontrolled/unstructured environments. The model allows variations of brightness from one frame to the next, in the form of a linear transformation, characterized by a multiplier and an offset field.

Earlier papers emphasize the performance of a particular method we had proposed for the computation of optical flow based on this model, in comparison to other methods. In this paper, we investigated some characteristics of the solution of this method. In particular, we have shown analytically the existence of a bias in the solution in the presence of multiplicative-type brightness changes (nonzero multiplier field), due to phase shifting in the estimation of brightness derivatives from non-symmetric masks. For example, in the absence of any motion, the algorithm gives an erroneous flow in the diagonal direction, proportional to the multiplier field. We proposed a simple correction in the implementation of the algorithm to remove this bias. In the presence of motion, the same algorithm gives some error in the flow estimates, which is relatively small for small brightness variations (typically, of the order of 5% or less for a brightness variation as large as 30%, $m_t = \pm 0.3$). We have also shown how errors in the transformation (multiplier and offset) can be corrected. Experimental results were presented to verify the theoretical developments.

The emphasis of this paper was on the computation of the flow and improving the accuracy of the solution. Other work currently under investigation include the use of the transformation fields to learn about physical events that give rise to them. An application of particular interest to us is in the operation of deep sea vehicles, which carry their own light source. For example, motion of the vehicle towards the scene results in an increase in the brightness level of the image, proportional to the motion. This information can be used to determine the vehicle motion, and to support computations based on the image motion. The proposed method is also practical where the camera aperture or exposure time is computer-controlled to maintain a steady average image brightness level, to prevent camera saturation, or to increase signal-to-noise ratio when illumination drops.

References

1. Baron, J.L., D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt, "Performance of Optical Flow Techniques," TR-299, Dept. Comp. Science, University Western Ontario, March, 1992.
2. Besl, P., and R.A. Jain, "Segmentation through Variable-Order Surface Fitting," *IEEE PAMI*, Vol 10, No 2, March, 1988.
3. Cornelius, N., and T. Kanade, "Adapting Optical Flow to Measure Object Motion in Reflectance and X-Ray Image Sequences," *ACM SIGGRAPH/SIGART Interdisciplinary Workshop on Motion: Representation and Perception*, Toronto, Canada, April, 1983.
4. Horn, B.K.P. and B.G. Schunck, "Determining

Optical Flow," *Artificial Intelligence*, Vol. 17, 1981.

5. B.K.P. Horn, *Robot Vision*, MIT Press, Cambridge, Massachusetts, 1986.

6. Longuet-Higgins, H.C., and K. Prazdny, "The Interpretation of a Moving Retinal Image," *Proc. Royal Society of London B*, Vol 208, 1980.

7. Mukawa, N., "Motion Field Estimation for Shaded Scenes," *Proc. ICIP*, Singapore, September, 1989.

8. Mukawa, N., "Estimation of Shape, Reflection Coefficients and Illumination Direction from Image Sequences," *Proc. ICCV*, Osaka, Japan, December 1990.

9. Nagel, H.H., "Displacement Vectors Derived from Second-Order Intensity Variations in Image Sequences," *CVGIP*, Vol 21, No 1, January.

10. Negahdaripour, S., A. Shokrollahi, and M. Genert, "Relaxing the Brightness Constancy Assumption in Computing Optical Flow," *Proc. ICIP*, Singapore, September, 1989.

11. Negahdaripour, S., C. H. Yu, and A. Shokrollahi, "Recovering Shape and Motion from Undersea Images," *IEEE Oceanic Engineering*, Vol 15, No 3, p189-198, 1990.

12. Negahdaripour, S., and C. H. Yu, "A Generalized Brightness Change Model for Computing Optic Flow," *ICCV*, Berlin, Germany, May, 1993.

13. Pentland, A., "Linear Shape from Shading," *Int. Journal Computer Vision*, Vol 4, No 2, 1990.

14. Schunck, B.G., "Image Flow: Fundamentals and Future Research," *Proc. CVPR*, San Francisco, CA, June, 1985.

15. Verri, A., and T. Poggio, "Motion Field and Optical Flow: Qualitative Properties," *IEEE PAMI*, Vol 11, No 5, May, 1989.

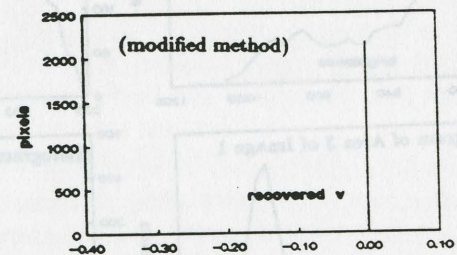
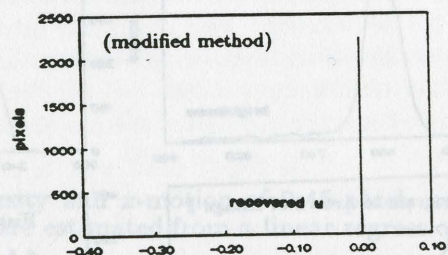
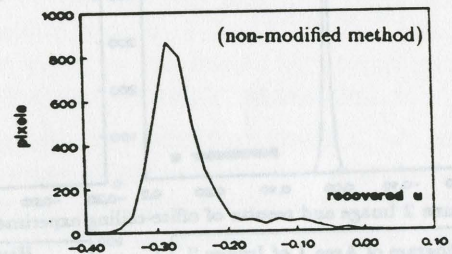
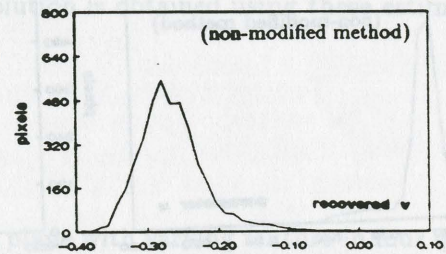
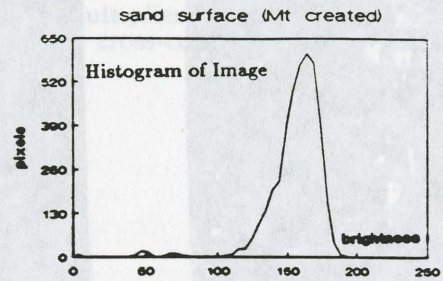
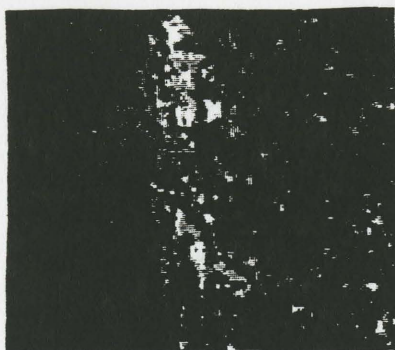
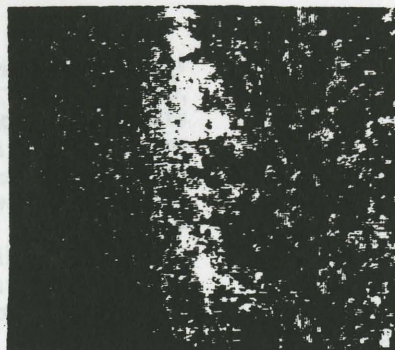


Figure 1 Image and results of the sand-image experiment described in Section 4.1.



(a) First Image of Office Ceiling



(b) Second Image of Office Ceiling

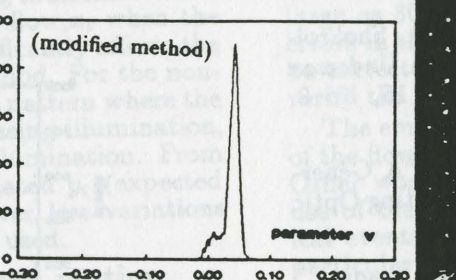
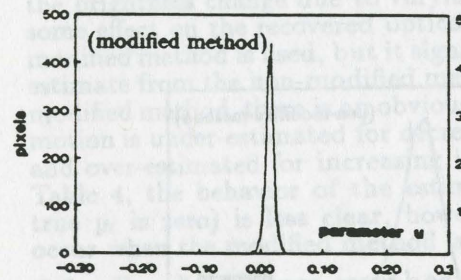
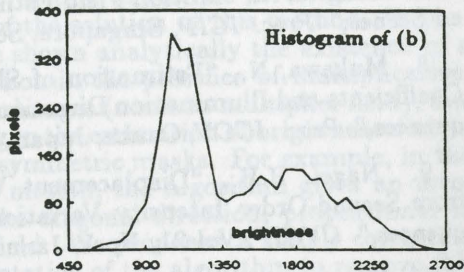
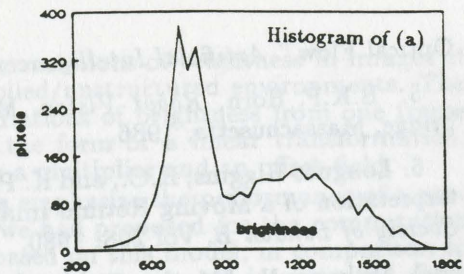
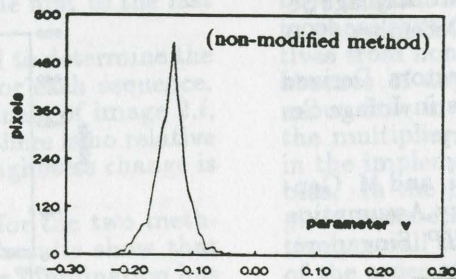
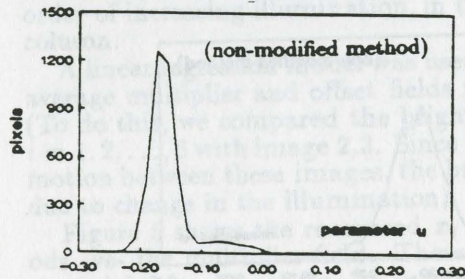


Figure 2 Image and results of office-ceiling experiment described in Section 4.2.

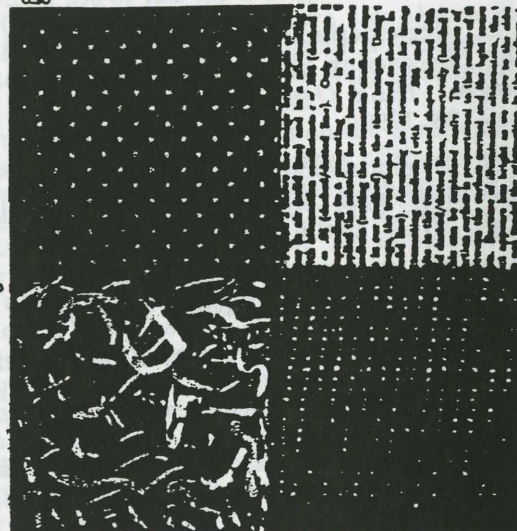
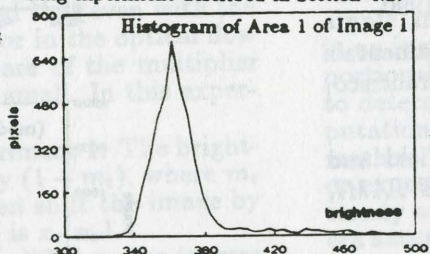
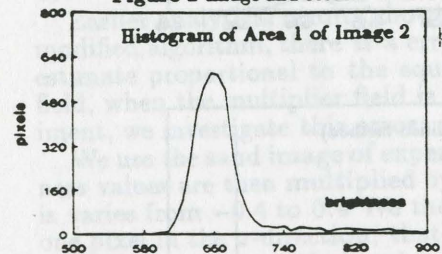


Figure 3 Image of four different texture areas described in section 4.3.

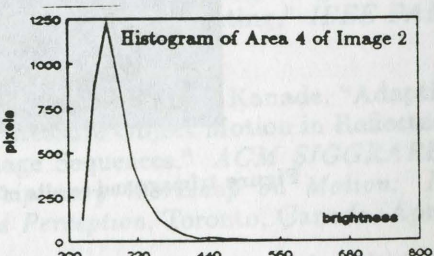
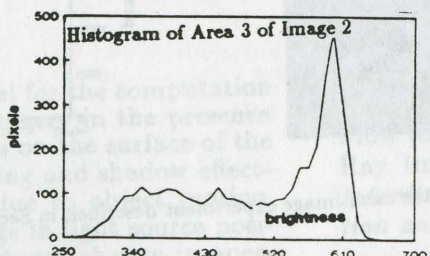
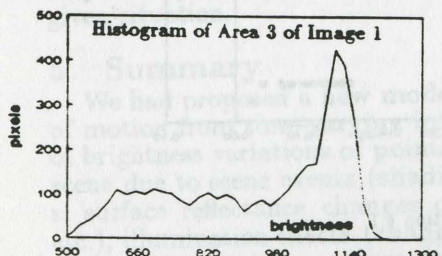
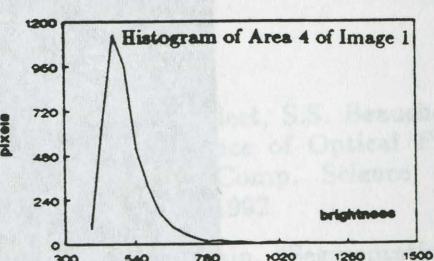
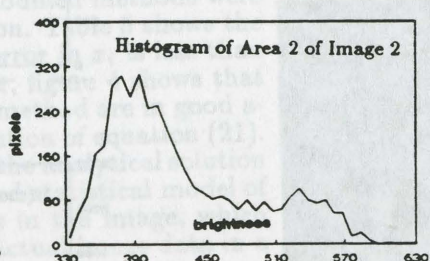
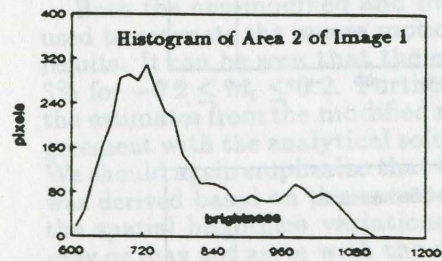


Figure 4 Brightness histograms of four different texture areas described in section 4.3.

Table 1. Results of the sand-image experiment involving a synthesized multiplier field $m_t = 1$; the average image brightness is $M = 155.8$, and the estimated average auto-correlation and cross-correlation were calculated to be $R_0^2 = 2.469e + 4$, $R_1^2 = 2.448e + 4$, and $R_2^2 = 2.441e + 4$.

solution	x_t	y_t	m_t
true	0	0	1.
analytical	-0.28	-0.28	0.831
non-modified	-0.276	-0.273	0.84
modified	-0.26e-6	0.56e-7	1.

Table 2. Results of the office-ceiling experiment with change of camera aperture and no motion; average image brightness is $M = 979.2$, estimated average auto-correlation and cross-correlation were calculated to be $R_0^2 = 1.040e + 6$, $R_1^2 = 1.021e + 6$, and $R_2^2 = 1.004e + 6$. The analytical solution is obtained using these estimate

solution	x_t	y_t	m_t	c_t
true	0	0	0.492	-7.62
analytical	-0.185	-0.185	0.465	15.7
non-modified	-0.156	-0.135	0.448	32.3
modified	0.016	0.037	0.500	-14.7

Table 3. Recovered flow and transformation fields for the image of a plane with varying texture in four quadrants. There is no motion between frames and the "true" transformation fields were computed from the two images using a linear regression model. The average brightness of the four regions in the first image are 671., 794., 937., and 514., and the average e_t values are -303., -364., -433., and -228., respectively.

	area 1	area 2	area 3	area 4
true m_t	-0.470	-0.475	-0.475	-0.468
true c_t	12.2	11.9	12.3	12.1
non-modified				
x_t	0.29	0.30	0.28	0.25
y_t	0.28	0.28	0.27	0.24
m_t	-0.440	-0.452	-0.431	-0.377
c_t	-7.1	-4.3	-28.1	-34.5
modified				
x_t	0.004	0.004	0.003	0.005
y_t	-0.009	-0.010	-0.009	-0.006
m_t	-0.642	-0.644	-0.641	-0.646
corrected m_t	-0.486	-0.487	-0.485	-0.488
c_t	30.4	30.4	28.6	30.6

Table 4. Estimated flow for sand-image with varying source intensity and x -motion of 0.45 pixel, using non-modified modified methods. The "true" multiplier and offset fields were estimated from a linear regression model.

2nd image label	2.1	2.2	2.3	2.4	2.5	2.6
true m_t	-0.18	-0.077	0.00	0.029	0.12	0.20
true c_t	2.44	1.22	0.00	-1.16	-3.08	-4.51
non-modified						
x_t	0.29	0.35	0.41	0.47	0.54	0.58
y_t	0.11	0.06	0.02	-0.04	0.02	-0.02
modified						
x_t	0.43	0.42	0.42	0.47	0.48	0.46
y_t	0.03	0.02	0.02	0.04	0.05	0.05

Table 5. Estimated flow from the modified method for the sand-image with varying synthesized multiplier field (from -0.4 to 0.4) and one-pixel x -motion.

m_t	-0.4	-0.3	-0.2	-0.1	0.0	0.1	0.2	0.3	0.4
x_t	0.895	0.949	0.983	0.998	1.00	0.991	0.975	0.953	0.929
y_t	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

FLAT SURFACES: A VISUAL INVARIANT

DANIEL RAVIV

Robotics Center and Electrical Engineering Department
Florida Atlantic University (FAU), Boca Raton, FL 33431; and
Robot Systems Division
National Institute of Standards and Technology (NIST)
Bldg 220, Room B124, Gaithersburg, MD 20899

ABSTRACT

This paper deals with machine perception of flat surfaces. For an observer that undergoes only translational motion in parallel to a planar surface, we show that a nonlinear function of optical flow produces the same value for all points on this surface, i.e., there is an optical-flow based invariant for all points that lie on a flat surface. We discuss some potential uses of this invariant.

1. INTRODUCTION

The problem of constant perception despite varying visual sensations, i.e., how despite different images of the same scene, the world is perceived as stationary, has puzzled many researchers in the last five decades (see for example [1]). In an attempt to answer this question another basic question arises: Is there an image-sequence transformation under which permanent properties of the environment are preserved during the motion of the eye? [1,6,7]

This paper deals with machine perception of flat surfaces. For an observer that undergoes only translational motion in parallel to a planar surface, we show that a nonlinear function of optical flow produces the same value for all points on this surface, i.e., there is an optical-flow based invariant for all points that lie on a flat surface.

2. A FLAT SURFACE INVARIANT

The optical flow generated by a point in 3-D space as expressed in spherical coordinates is (See Appendix and [2-5]):

$$\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \frac{1}{R} \begin{bmatrix} -\frac{\sin\theta}{\cos\phi} & \frac{\cos\theta}{\cos\phi} & 0 \\ -\sin\phi \cos\theta & -\sin\phi \sin\theta & \cos\phi \end{bmatrix} \begin{bmatrix} -U-BZ+CY \\ -V-CX+AZ \\ -W-AY+BX \end{bmatrix} \quad (1)$$

Refer to Figure 1. Assume that the Z axis of the camera is perpendicular to the flat surface. For the case where the camera undergoes translation only along its optical axis (the Y axis) i.e., $A=B=C=U=W=0$, Equation (1) becomes:

$$\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \frac{1}{R} \begin{bmatrix} -\frac{\sin\theta}{\cos\phi} & \frac{\cos\theta}{\cos\phi} & 0 \\ -\sin\phi \cos\theta & -\sin\phi \sin\theta & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ -V \\ 0 \end{bmatrix} \quad (2)$$

For the restricted translational motion of the camera as mentioned above, the following geometrical relationship holds for all points on the flat surface:

$$R = \frac{Z}{\sin\phi} \quad (3)$$

By combining Equations (2) and (3) we obtain:

$$\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \frac{V}{Z} \begin{bmatrix} -\frac{\cos\theta \sin\phi}{\cos\phi} \\ \sin^2\phi \sin\theta \end{bmatrix} \quad (4)$$

From the last equation we obtain:

$$-\frac{\dot{\theta}}{\tan\phi \cos\theta} = \frac{\dot{\phi}}{\sin^2\phi \sin\theta} = \frac{V}{Z} \quad (5)$$

In Equation (5) V and Z are constant for all points on the flat surface, and thus $-\frac{\dot{\theta}}{\tan\phi \cos\theta}$ and $\frac{\dot{\phi}}{\sin^2\phi \sin\theta}$ are constant for all points on the

flat surface (they are also equal to each other). This also means that *any point on the flat surface produces the same values of some functions of optical flow which are, in our case, $-\frac{\dot{\theta}}{\tan\phi\cos\theta}$ and $\frac{\dot{\phi}}{\sin^2\theta\sin\theta}$* . The value obtained from these optical-flow-based functions is *invariant* under the given motion.

3. DISCUSSION

In this short paper we presented an optical-flow-based invariant of a flat surface for translational motion of a camera. The approach is a step toward the understanding of perception of 3-D rigid bodies.

This invariant can be used to build topographic maps from a camera that undergoes translation above a terrain, since for a horizontally translating camera each height above a certain reference level corresponds to a value of the optical-flow-based previously-described functions. This invariant can also be used to detect obstacles on a flat road using an on-board camera, since all points on the road will generate the same value of the previously expressed functions and any point above or below the road surface will produce a different value.

In this paper the invariant is derived for a special motion of the camera, where it undergoes translation only along its optical axis (the Y axis) and the Z axis of the camera is perpendicular to the flat surface. However, by reorienting the camera and applying appropriate calibration, a less restricted result can be obtained.

We are currently working on invariants for more general camera motion (in particular motions that involve fixation) and arbitrary shape objects.

4. ACKNOWLEDGEMENTS

The author would like to thank Jim Albus, Emie Kent, Marty Herman and Don Orser for fruitful discussions regarding this and other invariants.

5. REFERENCES

- [1] Cutting, J.E., *Perception with an Eye for Motion*, MIT Press, 1986.
- [2] Horn, B.K.P., *Robot Vision*, MIT Press, 1986.
- [3] Meriam, J.L., *Dynamics*, John-Wiley & Sons, 1975.
- [4] Raviv, D., and Albus, J.S., "A Closed-Form, Massively-Parallel Range-From-Image-Flow Algorithm", NISTIR-4450, 1990.
- [5] Raviv, D., "A Quantitative Approach to Camera Fixation", NISTIR-4324, 1990.
- [6] Raviv, D. and Albus J.S., "Representations in Visual Motion", NISTIR-4747, 1992.
- [7] Raviv, D., "Invariants in Visual Motion", NISTIR-4722, 1992.

APPENDIX: EQUATIONS OF MOTION AND OPTICAL FLOW

This appendix describes the equations that relate a point in 3-D space to the projection of that point in the image for general six-degree-of-freedom motion of the camera.

In the following analysis, we assume a moving camera in a stationary environment. Suppose the coordinate system is fixed with respect to the camera as shown in Figure A1. Assume a pinhole camera model and that the pinhole point of the camera is at the origin of the coordinate system. We derive the optical flow components in the spherical coordinates $(R-\theta-\phi)$. In this frame, angular velocities ($\dot{\theta}$ and $\dot{\phi}$) of any point in space, say P , are identical to the optical flow values at P' in the image domain. Figure A2 illustrates this concept: $\dot{\theta}$ and $\dot{\phi}$ of a point in space are the same as $\dot{\theta}$ and $\dot{\phi}$ of the projected point P' in the image domain, and therefore there is no need to convert angular velocities of points in 3-D space to optical flow. In Figure A2 the image domain is a sphere. However, for practical purposes the surface of the image sphere can be mapped onto an image plane (or other surface).

We start with the derivation of the velocity of a 3-D point in the XYZ coordinates (Figure A1). Let the instantaneous coordinates of the point P be $\mathbf{R} = (X, Y, Z)^T$ (where the super-

script T denotes transpose). If the instantaneous translational velocity of the camera is $\mathbf{t} = (U, V, W)^T$ and the instantaneous angular velocity is $\boldsymbol{\omega} = (A, B, C)^T$ then the velocity vector \mathbf{V} of the point P with respect to the XYZ coordinate system is:

$$\mathbf{V} = -\mathbf{t} - \boldsymbol{\omega} \times \mathbf{R} \quad (\text{A1})$$

or:

$$V_x = -U - BZ + CY \quad (\text{A2})$$

$$V_y = -V - CX + AZ \quad (\text{A3})$$

$$V_z = -W - AY + BX \quad (\text{A4})$$

where V_x , V_y , and V_z are the components of the velocity vector \mathbf{V} along the X , Y , and Z directions respectively.

To convert from $R\theta\phi$ to XYZ coordinates we use the relations:

$$X = R \cos\phi \cos\theta \quad (\text{A5})$$

$$Y = R \cos\phi \sin\theta \quad (\text{A6})$$

$$Z = R \sin\phi \quad (\text{A7})$$

Similarly, to convert from XYZ to $R\theta\phi$ coordinates we use:

$$R = \sqrt{X^2 + Y^2 + Z^2} \quad (\text{A8})$$

$$\theta = \tan^{-1} \frac{Y}{X} \quad (\text{A9})$$

$$\phi = \sin^{-1} \frac{Z}{\sqrt{X^2 + Y^2 + Z^2}} \quad (\text{A10})$$

In order to find the optical flow of a 3-D point in $R\theta\phi$ coordinates, we use the following relations and transformations:

Let v_R , v_θ , and v_ϕ be the components of the vector \mathbf{V} in spherical coordinates, and

$$V_{R\theta\phi} = \begin{bmatrix} v_R \\ v_\theta \\ v_\phi \end{bmatrix} \quad (\text{A11})$$

$$V_{XYZ} = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \quad (\text{A12})$$

Then:

$$V_{R\theta\phi} = [T_\phi][T_\theta]V_{XYZ} \quad (\text{A13})$$

where

$$[T_\theta] = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{X}{\sqrt{X^2+Y^2}} & \frac{Y}{\sqrt{X^2+Y^2}} & 0 \\ \frac{-Y}{\sqrt{X^2+Y^2}} & \frac{X}{\sqrt{X^2+Y^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (\text{A14})$$

and

$$[T_\phi] = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{X^2+Y^2}}{\sqrt{X^2+Y^2+Z^2}} & 0 & \frac{Z}{\sqrt{X^2+Y^2+Z^2}} \\ 0 & 1 & 0 \\ \frac{-Z}{\sqrt{X^2+Y^2+Z^2}} & 0 & \frac{\sqrt{X^2+Y^2}}{\sqrt{X^2+Y^2+Z^2}} \end{bmatrix} \quad (\text{A15})$$

Also:

$$V_R = \dot{R} \quad (\text{A16})$$

$$V_\theta = R \dot{\theta} \cos\phi \quad (\text{A17})$$

$$V_\phi = R \dot{\phi} \quad (\text{A18})$$

where dot denotes first derivative with respect to time.

Using equations (A2)-(A18) yields the following expressions:

$$\begin{bmatrix} R \dot{\theta} \cos\phi \\ R \dot{\phi} \end{bmatrix} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ -\sin\phi \cos\theta & -\sin\phi \sin\theta & \cos\phi \end{bmatrix} \begin{bmatrix} -U - BZ + CY \\ -V - CX + AZ \\ -W - AY + BX \end{bmatrix} \quad (\text{A19})$$

$$\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \frac{1}{R} \begin{bmatrix} -\frac{\sin\theta}{\cos\phi} & \frac{\cos\theta}{\cos\phi} & 0 \\ -\sin\phi \cos\theta & -\sin\phi \sin\theta & \cos\phi \end{bmatrix} \begin{bmatrix} -U - BZ + CY \\ -V - CX + AZ \\ -W - AY + BX \end{bmatrix} \quad (\text{A20})$$

As mentioned earlier, $\dot{\theta}$ and $\dot{\phi}$ of a point in space (i.e., the angular velocities in the camera coordinate system) are the *same* as the optical flow components $\dot{\theta}$ and $\dot{\phi}$ (Figure A2).

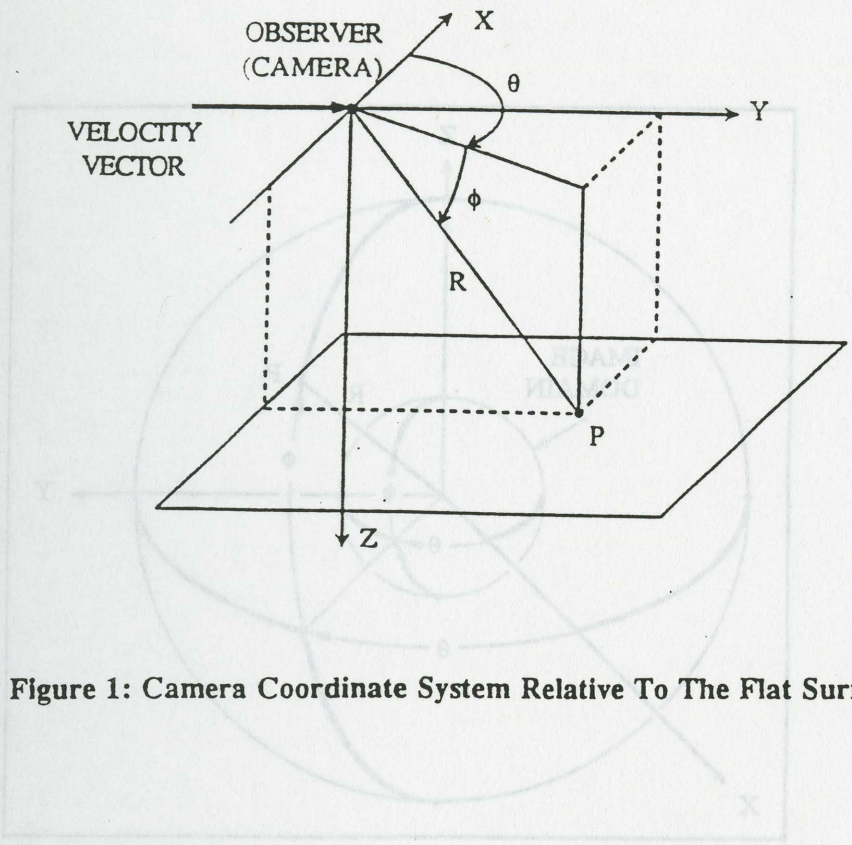


Figure 1: Camera Coordinate System Relative To The Flat Surface

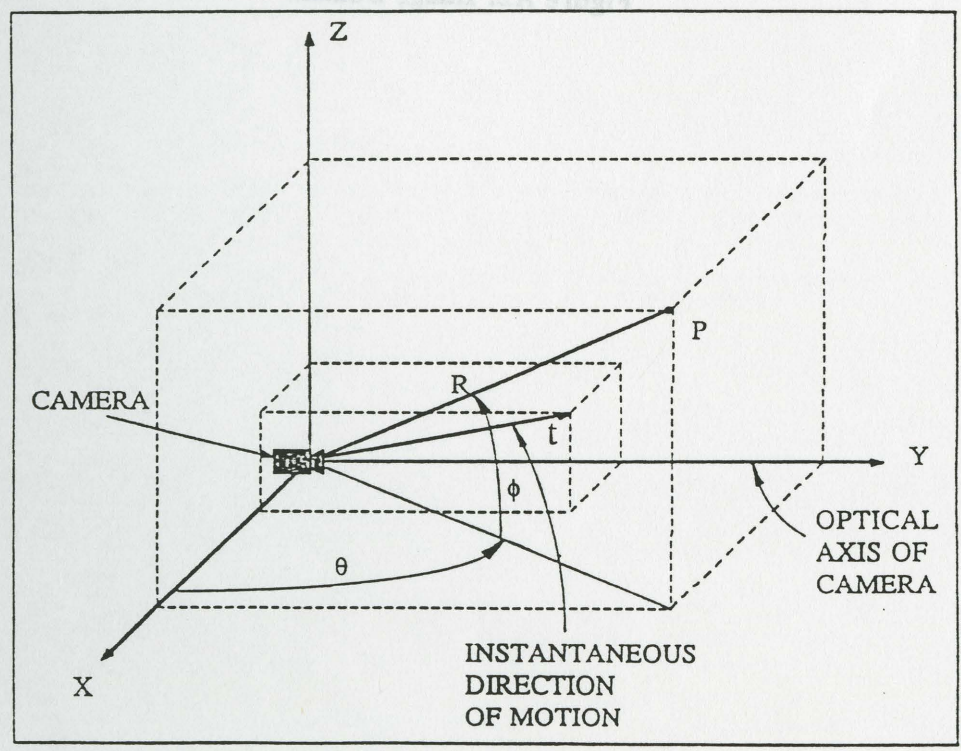


Figure A1: Camera Coordinate Systems

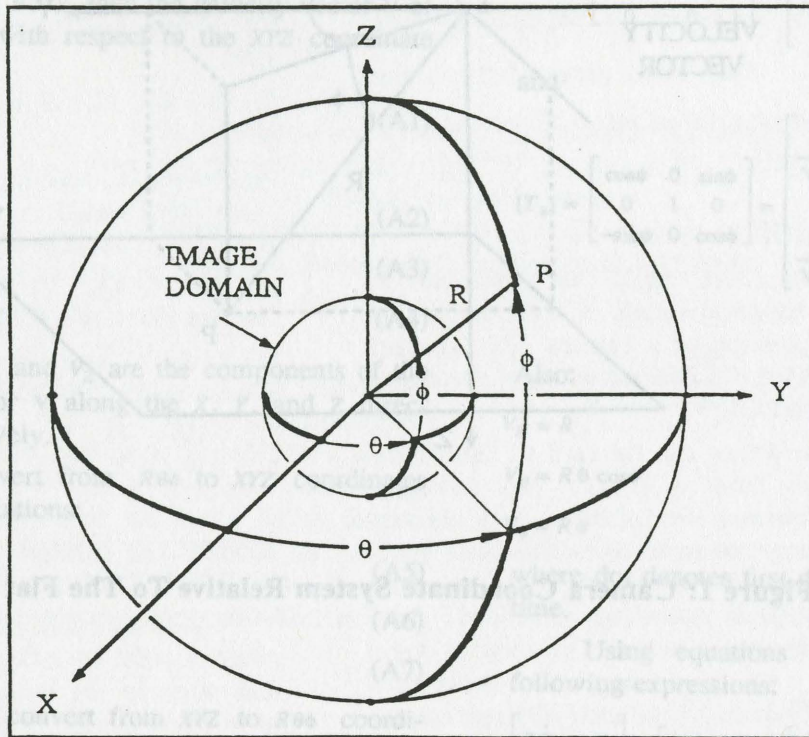


Figure A2: Image Domain

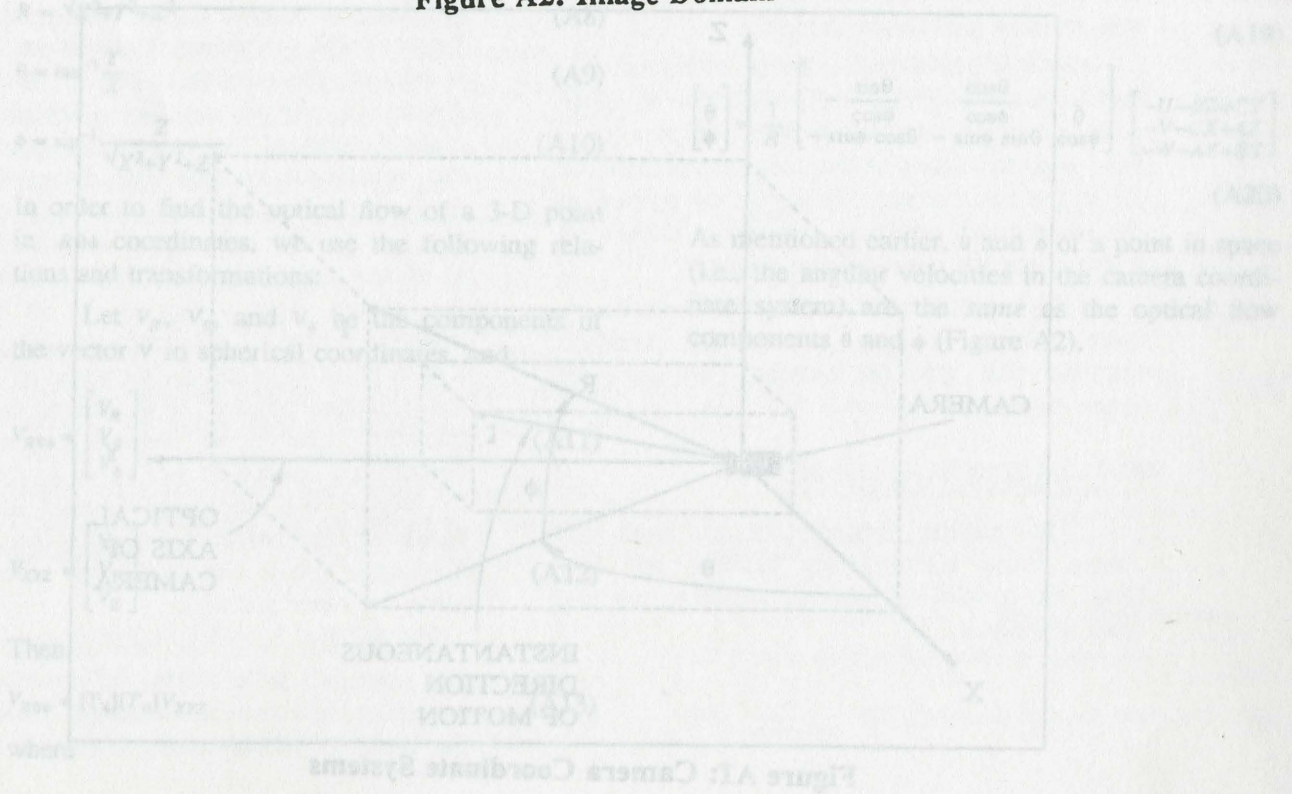


Figure A1: Camera Coordinate Systems

