USING DEEP LEARNING SEMANTIC SEGMENTATION TO ESTIMATE VISUAL ODOMETRY

by

Jason R Blankenship

A Thesis Submitted to the Faculty of

The College of Engineering and Computer Science

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Florida Atlantic University

Boca Raton, FL

May 2018

USING DEEP LEARNING SEMANTIC SEGMENTATION TO ESTIMATE VISUAL ODOMETRY

by

Jason Ryan Blankenship

This thesis was prepared under the direction of the candidate's thesis advisor, Dr. Hongbo Su, Department of Civil, Environmental and Geomatics Engineering, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the College of Engineering & Computer Science and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

SUPERVISORY COMMITTEE:

[illegible]
Hongbo Su, Ph.D.
Thesis Advisor

[illegible]
Evangelos I. Kaisar, Ph.D.

[illegible]
Sudhagar Nagarajan, Ph.D.

[illegible]
Yan Yong, Ph.D.
Chair, Department of Civil, Environmental and Geomatics Engineering

[illegible]
Stella N. Batalama, Ph.D.
Dean, College of Engineering and Computer Science

Diane Alperin
Diane Alperin, Ph.D.
Interim Dean, Graduate College

April 9, 2018
Date

ACKNOWLEDGEMENTS

I would like to express sincere gratitude to my advisory committee members, Dr. Sudhagar Nagarajan, Dr. Evangelos I. Kaisar, and my supervisor Dr. Hongbo Su, Department of Civil, Environmental and Geomatics Engineering. They have provided their guidance, have been patient and encouraging not just during the writing of this manuscript but also during my time as an undergraduate. I would also like to express gratitude to my lab mates for their support, especially Debojit Biswas for his immediate programming guidance. Lastly, I would like to express gratitude to Mr. Perry Weinthal for his creative inspiration and electrical lab support.

ABSTRACT

Author:	Jason Ryan Blankenship
Title:	Using Deep Learning Semantic Segmentation to Estimate Visual Odometry
Institution:	Florida Atlantic University
Thesis Advisor:	Dr. Hongbo Su
Degree:	Master of Science
Year:	2018

In this research, image segmentation and visual odometry estimations in real time are addressed, and two main contributions were made to this field. First, a new image segmentation and classification algorithm named DilatedU-NET is introduced. This deep learning based algorithm is able to process seven frames per-second and achieves over 84% accuracy using the Cityscapes dataset. Secondly, a new method to estimate visual odometry is introduced. Using the KITTI benchmark dataset as a baseline, the visual odometry error was more significant than could be accurately measured. However, the robust framerate speed made up for this, able to process 15 frames per second.

DEDICATION

I lovingly dedicate this research to my amazingly supportive wife. I could have never completed this journey without you.

USING DEEP LEARNING SEMANTIC SEGMENTATION TO ESTIMATE VISUAL ODOMETRY

TABLES

FIGURES

EQUATIONS

1. INTRODUCTION

1.1 Motivation

The topics of this research encompass various areas of computer vision, artificial intelligence, and photogrammetry. In the rising demand for autonomous vehicles, computer vision is an ever-growing field. Similar to how humans use their eyes to visually interpret the world around, the science of computer vision aims to give the same or better capability to a computer or machine. Comparable to this research, computer vision is being investigated exhaustively, i.e., augmented reality (Egodagamage, et al., 2018), autonomous drones and vehicles (Huang, et al., 2017), and even healthcare (Erdan, et al., 2016). Computer vision, photogrammetry and artificial intelligence (AI) often overlap in many research interests (Hartley, et al., 2003). The science and art of capturing reliable information about physical objects through the process of interpreting images is known as photogrammetry (American Society of Photogrammetry, 1980). Photogrammetry is used for creating maps and relied upon for producing various aerial surveys. Deep learning is another name for artificial intelligence (Goodfellow, et al., 2016). “Deep learning approaches to machine learning have had a major impact on speech recognition and computer vision” (Witten , et al., 2017). In this research we will explore various methods for estimating visual odometry along with various deep learning based image segmentation classification algorithms.

An important task in the arena of computer vision is semantic segmentation (Huang, et al., 2017). There are many settings that use semantic segmentation. One such is estimating street view scenes. This poses a severe problem, as street views often change in location and appearance. Even with the difficulties, this problem must be solved to support applications such as autonomous vehicles. In this section, the inspiration behind the research is discussed. The topics of this research encompass various areas of computer vision, artificial intelligence, and photogrammetry. When object recognition is not enough, and each object per-pixel must be located and identified, such as road, vehicle, etc. One of the main limitations with deep learning based algorithms is the significant amount of data required. Though, this challenge was overcome with the introduction of U-NET. However, with U-NET, real-time image processing is not present in the previously presented solution (Ronneberger, et al., 2015). By adding an atrous (Kendall, et al., 2016) layers, this will increase the frame rate to be closer to real-time processing.

There are various other methods for tracking a vehicles distance traveled or location. Wheel odometry is one of the most widely utilized methods to estimate the position of a vehicle (Borenstein J, 1996). This method counts the number of revolutions the wheel has made and based on this it estimates the distance traveled. This method is inexpensive; however, it is prone to error due to wheel slippages (Aboelmagd N, 2013). Another method uses an inertial measurement unit (IMU) (Rone W, 2013). This method measures the angle and forces with respect to gravity to estimate the position. An advantage of this method is that it does not require an external source, as it is a passive device. The disadvantage is that requires a double integration to estimate the position; therefore a small error will quickly accumulate (Wang D, 2014). GPS is an additional method used to determine location. This method utilizes a receiver and a minimum of four satellites. By using trilateration, ground receivers calculate their position based on the time difference between the satellites sending time and the time received on the receiver (Aboelmagd N, 2013). One of the main advantages that visual odometry has over traditional navigation systems, i.e., GPS, it is not affected by dropouts caused by obstacles and can operate where GPS is not available such as tunnels or extraterrestrial planet exploration. Visual odometry can be used to determine the distance traveled, and position of a camera. The current primary methods rely on a feature detector to detect key pixel in an image. Before reconstruction, a scene from a series of overlapping images, the position of each frame must be known. A crucial challenge in estimating the visual odometry is the image processing. Estimating visual odometry is a passive method for determining the cameras position and orientation. Visual odometry relies heavily on the feature extraction. Currently ORB and FAST are two prominent feature extractors. By reducing the amount of data, the feature detector must process; it will decrease the time required and advanced to the next frame sooner. Frame skipping is often a method that is used to stay real-time. When skipping frames, valuable information may not be captured. There have been many advancements in feature detection algorithms (Taketomi, et al., 2017). Yet, using the whole image for feature detection produces too many features and thus need to be reduced to only the important one. For this reason, this research is proposing to use the segmented images and also test the use of only the extracted lines. By using only, the edges from segmented images, this could decrease the time for feature extraction. By increasing the speed of feature extraction for estimating visual odometry, computer vision scientist working with autonomous vehicles will benefit.

1.2 Literature Review

In this section, a literature review camera calibration, deep learning image segmentation, feature detectors, and methods of estimating camera visual odometry are presented. Some matters, such as camera calibration are well established, and fewer studies were required. While others like methods of deep learning image segmentation, more focus has been paid due its recently. There's an extensive list of the literature of dense image segmentation methods. In this review, only a small subset is surveyed.

1.2.1 Camera Calibration

Camera calibration is needed to remove various distortions from both the camera sensor and lens, ensuring the accuracy of final measurements. There are multiple methods for camera calibrations (Heikkila, et al., 1997). Camera calibration of intrinsic metrics has been meticulously examined in previous research. A brief literature review on methods of camera calibration is presented in this section.

Heikkila, et al. (1997) also completed a brief literature review of various camera calibration methods and presented a four-step technique for camera calibration. Their concern was that there is too little research that focuses on the entire camera calibration process and mostly concentrated on model fitting. The four-step method was built upon a classic photogrammetric method in (American Society of Photogrammetry, 1980). The first parameter values in the two-step method are linearly calculated, then the ending values are found using nonlinear minimization (Heikkila, et al., 1997). Extended this process with a third step by adding control points whose projections are greater in size than one pixel. Finally, the fourth stage of the proposed procedure solves the image correction problem by using the new implicit model that interpolates the correct points based on the physical parameters. This method presented demonstrated a higher accuracy camera calibration that is advantageous in 3D measurements for camera-based applications such as computer vision mapping. Charge-coupled device (CCD) cameras have the accuracy capacity higher than 1/50 of the pixel size (Heikkila, 2000). In (Bhardwaj, et al., 2017) a method labeled as AutoCalib for the automatic calibration of traffic cameras was proposed. AutoCalib is a system that uses deep learning methods to extract critical features from traffic camera frames. Using a filtering and aggregation algorithm, they were able to derive and implement the needed calibration parameters automatically. This method produce was able to perform in real-world settings with a metric error less than 12%.

1.2.2 Segmentation and Classification

Image segmentation is completed by joining edges of homogeneous image pixels that are separated into regions. This area of image segmentation has been thoroughly studied, however, is a hot topic in computer vision. Some segmentation methods are threshold based, where a pixel or series of pixels meet some criteria, then it is segmented, similar to Canny (1986). This method uses a Gaussian smoothing to find edges along with local non-maxima suppression to remove false edges. However, is not able to be adaptive as it solely relies on thresholds. Other methods use a machine learning approach, relying on previously known data. Bayes' Theorem is a typical architecture for this type of approach, as it relies on previous outcomes based on prior sequences or pixel relationship to corresponding or surrounding pixels. Settings like nearest neighbor can be chosen, along with small selections from the imagery of known pixels. These contribute to the creation of a supervised machine learning model. This model is then used or applied to the entire image or even a series of analogous images.

Another type of supervised learning is the use of Deep Learning, similar to what is used in this research. Deep Learning can be viewed as a subsect of machine learning. Deep Learning uses a series of hidden layers each with a defined set of instructions then outputs to the next, then finally creates an output decision. These outputs can be as simple as determining the if an image contains a cat or a dog and as extensive as identifying each pixel as done in this research. By using multiple layers, each layer looks for specific feature example of Figure 2. The first layer is what a human may look at. However, the Deep Learning method might inspect for generalized features of edges on the first hidden layer, then corners or contours on the second and so forth, then finally an output layer or classification occurs (Goodfellow, et al., 2016).

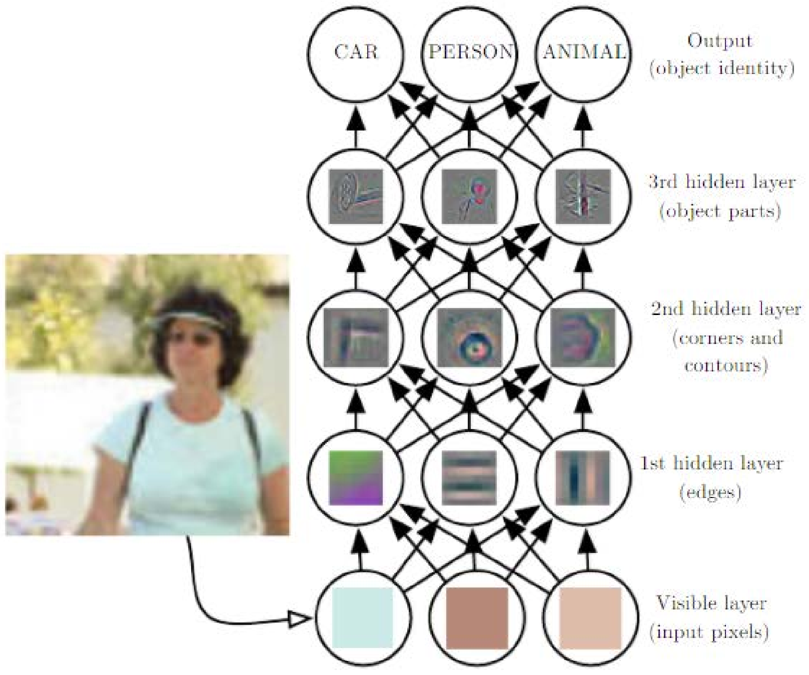


Figure 1 Deep Learning Layers, Source: Computer Vision for Multiple View Geometry

In (Kundu, et al., 2014) a method that uses a series of overlapping images and an SFM algorithm that generates a 3D point cloud is presented. Then the images are segmented and classified using 2D scene parsing method. The 2D segmented images are then overlaid onto the 3D point cloud as shown in Figure 1. Although the 3D sparse point cloud can be generated in real-time, however much post-processing is required in order the create the 3D segmentation portion.

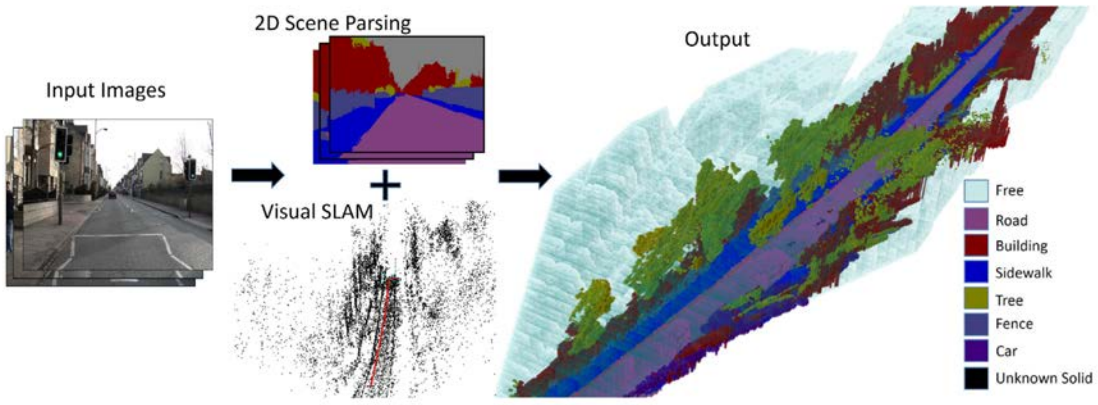


Figure 2 2D Segmentation overlaid on 3D point cloud, Source: Kundu, Li, Dellaert, Li, & Rehg

During the literature review of deep learning based image segmentation methods, a method named SegNet was found. One of the main limitations with deep learning based algorithms is the significant amount of data required. Though, this challenge was overcome with the introduction of U-NET (Ronneberger, et al., 2015). The key part of U-NET is the use of concatenate of the semantic mask. This enables a lower amount of required training data. When introduce, Ronneberger, et al. (2015) only used 30 images to achieve over 90% accuracy. Erdan, et al. (2016) also presented using the U-NET architecture however they used three-dimensional convolutional layers. The primary objective was to detect and identify brain tumors using Magnetic Resonance Imaging (MRI) brain scans. Due to the extensive amount of resources required by three-dimensional convolutional layers, tests of various settings were performed using two-dimensional convolutional layers initially to find the best optimization settings. The final experiments were performed using three-dimensional convolutional layers. With only 285 brain volumes and only 25 epochs, they were able to achieve accuracy over 70%. The first version of SegNet presented by Badrinarayanan, et al. (2015) proposed an original deep architecture for semantics pixel-wise image labeling. SegNet works by re-purposing previously trained image classification models for semantic segmentation. This is performed by applying an Atrous convolution with upsampled filters for feature extraction. Atrous convolution is also known as dilated convolution (KERAS and TensorFlow) or Atrous Spatial Pyramid Pooling (Chen, et al., 2017). The first introduction of the dilated convolution network was in (Yu, et al., 2016). Fundamentally this type of convolution supports increasing receptive fields without reducing the resolution or coverage via “dilating” the kernel by 1 – 1 pixels (Huang, et al., 2017). In (Bláha, et al., 2016) a method for joint 3D reconstruction and semantic labeling of large city model from both terrestrial and aerial images was presented. Knowing that the observed surface is only a 2D manifold in a 3D space, they used an adaptive multiresolution framework for semantic 3D reconstruction. This hierarchical scheme is based on the assumption that high spatial resolution and high numerical precision are only in regions that are likely to contain a surface. This method claims to save 98% memory and 95% time, while still maintaining the accuracy of the existing means of creating large 3D semantic models. Then fully connected conditional random fields were combined with deep convolutional neural networks to create accurate predictions of detailed segmentation maps. This proposed method significantly advanced over the PASCAL VOC 2010 semantic image segmentation benchmark. The architecture consists of a concatenation path that captures the context and a proportioned expanding path that enables accurate localization.

Advancing on the original SegNet, the Bayesian SegNet, a probabilistic pixel-wise semantic segmentation with a deep-learning backbone was presented in (Kendall, et al., 2016). This deep learning approach showed to be functional for smaller size data sets. One advantage the method has is the output of uncertainty that is produced. This is a visual way of representing the effectiveness of the model. Another positive note is the ability to be run in real-time on a GPU. The main contribution was expanding deep convolutional encoder-decoder neural network architectures to Bayesian-based convolutional neural networks.

1.2.3 Edge Detection and Visual Odometry

It is worth noting that Simultaneous Location and Mapping (SLAM), or Bundle Adjustment methods topics have previously been investigated. Tomasi, et al. (1992) presented a factorization method that can overcome the difficulty of inferring scene geometry and camera motion from a stream of images. Today this method is known as visual odometry. They displayed an accurate method that uses the rank theorem to factorize the measurement in shape and motion tracking. They factored the measurement matrix into two separate matrices; one for the rotation of the camera and the other for the object shape in a scene. Also since two of the three translations are taken care of in preprocessing, it can handle a partially completed matrix that resulted from tracking failures or distortions.

Sturm, et al. (1996) proposed using a factorization based algorithm to solve for the location and motion tracking of a series of sequential images and can be view as an extension of methods presented in (Tomasi, et al., 1992). However, in (Sturm, et al., 1996) the main technical contribution is a novel method to recover the unknown projective depths, using fundamental matrices and estimated from the image data. After normalizing the image coordinates, they estimated the matrices and epipoles. Subsequently, the scale factors were determined and rescaled the measurement matrix *W*. Matrix *W* was then balanced via scalar multiplication. Afterward, the Singular Value Decomposition (SVD) is computed, and the recovery of both the motion and shapes are completed. SVD is both an efficient and accurate method, being able to use in real time.

Later Poelman, et al. (1997) compared the previous orthographic projection factorization method with the newer Para perspective method. One of the difficulties with orthographic factorization method is that when objects move into, away from the camera, or across the scene, inaccurate motion results and errors in the shape occur. They discovered by adding iterations into the perspective refinement step can significantly reduce the perspective distortions. However, one of the leading complications with this method is the efficiency that it lacks and is not able to be used for a real-time application.

NASA first introduced visual odometry for the Mars Exploration Rovers (MER) project. Visual odometry was developed so that the rovers could navigate, as there is no global position sensor (GPS) satellites present (Helmickreceived, et al., 2004). Visual odometry uses a series of algorithms to find the location, position, and orientation of the camera.

Commonly a vital part of estimating visual odometry is feature detection. When humans navigate up or down stairs, the edges and corners are located before making a step. This is analogous to how a feature extraction or feature matching algorithm operates. At the heart of many computer vision technologies are feature point descriptors (Calonder, et al., 2010). There are multiple feature detector algorithms available. Like most approaches, some perform better than others. The Harris corner (Harris, et al., 1988) detector uses the differential of the corner weight and directly references the direction. It has been improved and adopted by many for other subsequent applications (Hartley, et al., 2003), (Rosten, et al., 2006). Also, worth mention, Scale Invariant Feature Transform (SIFT) converts data from images into scale-invariant coordinates to local features (Lowe, 2004). This is performed by generating numerous feature points that cover most of the image. These points are then extracted, stored in a database. On the following frame, this occurs again, and the two files are compared to find matches based on the Euclidean distance. Due to the comparison of files, SIFT computationally costly, thus not able to perform in real-time. Similarly, Speeded-Up Robust Features (SURF) is a scale and rotation invariant detector and descriptor that uses integral images for image convolutions (Bay, et al., 2007). This method is faster than SIFT, however similar to SIFT, the creators have patented it and there are use restrictions (Rublee, et al., 2011). The Features from Accelerated Segment (FAST) algorithm is a high-speed feature detector that was designed to be used as a tool to enable faster 3D scene reconstruction for SLAM problems (Rosten, et al., 2006). Explicitly, the FAST algorithm was specifically designed to be able to be used for full frame rate speed. Although this method is faster than previous methods and ready to repeat under various kinds of features, it is not refined. It is threshold dependent, not robust to high levels of noise and can only respond to the one-pixel width at certain angles. An apt feature point descriptor that uses binary channels is named accordingly. Binary Robust Independent Elementary Features

(BRIEF) was introduced in (Calonder, et al., 2010). Similarly to SURF, it uses the Hamming distance which has proved to be very efficient for computations. When compared to SURF on standard benchmarks, BRIEF shows to be similar or better (Calonder, et al., 2010).

Another feature detector is the Oriented FAST and Rotated BRIEF (ORB) (Rublee, et al., 2011) which was built on the existing framework of FAST. This feature extraction algorithm combines the efficient computation of oriented BRIEF on top of FAST. Unlike FAST, ORB is rotation invariant and resistant to noise. The inventors of ORB also contributed a BSD license for the computer vision community to enjoy it freely.

There have been many successful results for monocular setups over long distances that estimate visual odometry. In (Nist´er, et al., 2004) they presented one of the first real-time largescale visual odometry with monocular camera setup. This method works by tracking features over a specific number of frames, then triangulates the features track and makes estimations on the trajectory or visual odometry. This method was tested using a robotic platform with Differential Global Positioning System(DGPS) as the baseline. The error ranged from 1% to 5% pending on the environment. The homogenous setting, a meadow proved to have the highest error. Also, similar to (Nist´er, et al., 2004) and (Lhuillier, 2005), they too used RANSAC to remove outliers. In (Lhuillier, 2005) and (E. Mouragnon, 2006) a method that performed a bundle adjustment over a set of frames to recover the motion a pose was presented. In (Garc´ıa, et al., 2012) two different methods for monocular based visual odometry was presented. Firstly Appearance-Based Visual Odometry was presented. Using the whole image without any feature extraction, a real-time algorithm that computes an appearance-based map. The second method like (Nist´er, et al., 2004), used the feature based system to estimate the visual odometry. The error rate using feature proved to be more consistently low (less than one sigma), while the image based continued to increase as the frame increased (close to 2 sigmas). To improve upon the localization of autonomous vehicle driving in an urban setting. Ballardini, et al. (2016) presented a method that uses the façade of detected vehicles to produce an accurate lane-level localization of the vehicle. Localization is being performed using a probabilistic framework called "Road Layout Estimator." Using images from a stereo rig mounted camera system, a mathematical representation of building fronts is matched against buildings that are available on OpenStreetMap.

1.3 Objective

Several studies have been performed that segment and classify images using deep learning. More specifically a convolutional neural network (CNN) with U-NET framework was explored. During the downsampling of the model, each of the pixels is analyzed by the convolution layers, causing the speed of the model to slow. This research aims to improve the on the image processing speed while maintaining or improving the accuracy by implementing an Atrous layer into the existing U-NET framework. Also, investigations have been conducted to estimate the visual odometry of a camera.

When image segmentation and visual odometry are performed, it is common that they are performed separately. Then the data must then put realigned in a later step. This method is prone to introduce error during the realignment process and adds time delay to the process. This research intends to explore a new method that performs both of these steps in a sequence, thus reducing the chance of introducing error while decreasing the process time. By utilizing only, the edges from a semantic segmentation frame sequence, time could be saved while maintaining accuracy.

To perform this, the following tasks were carried out;

- Setup and Calibrate Monocular Camera Setup
- Create a new image segmentation and classification algorithm that combines the U-NET framework with Atrous convolution layers
- Create a process to extract edges from segmented images without losing the classifications
- Implement method presented in (Garc´ıa, et al., 2012) to compare the Visual Odometry estimation between using images versus using the edges of the segmented images.

1.4 Thesis Structure

The overall layout of the thesis consists of four chapters. In Chapter 1, a concise literature review was executed. Chapter 1 also provides information on the motivations and challenges for current in practice methods. The methodology is discussed in Chapter 2, which includes camera calibration, data preparation, image segmentation and classification, 2D line extraction, and estimation of each camera pose using visual odometry. Chapter 3 presents the experiments and findings of stages of the methodology. Chapter 4 states essential observations and advises future recommendations.

1.5 Presentations and Publications

Significant findings and research works carried out during completion of this research have been presented as a poster in the 2018 American Society of

Photogrammetry and Remote Sensing Annual Conference at Denver, Colorado, USA, under the title “Using Computer Vision to Create Perpetual Map.” A journal manuscript related to the findings of creating a new method for semantic segmentation titled “DilatedU-NET Image Segmentation” is being prepared and is expected to be ready for review by the end of April 2018.

2. METHODOLOGY

2.1 General

As previously mentioned each solution comes with both strengths and weaknesses. The method being presented functions by combining computer vision algorithms with photogrammetry. It is intended to overcome the need to look at a raw image more than once for visual odometry. The techniques used in this study relies on previous topics, i.e., image segmentation, camera calibration, deep learning, and photogrammetry.

Due to the multipart procedures of this research, the methodology is organized into three sub-categories as shown in Figure 3. The camera calibration section which includes; the video feed was input process; extraction of calibration parameters; and how the parameters are implemented will be presented first. The second section describes the creation process for defining the model weights along with other aspects of creating the image processing algorithm. Lastly, the method for estimating the camera position or visual odometry is presented.

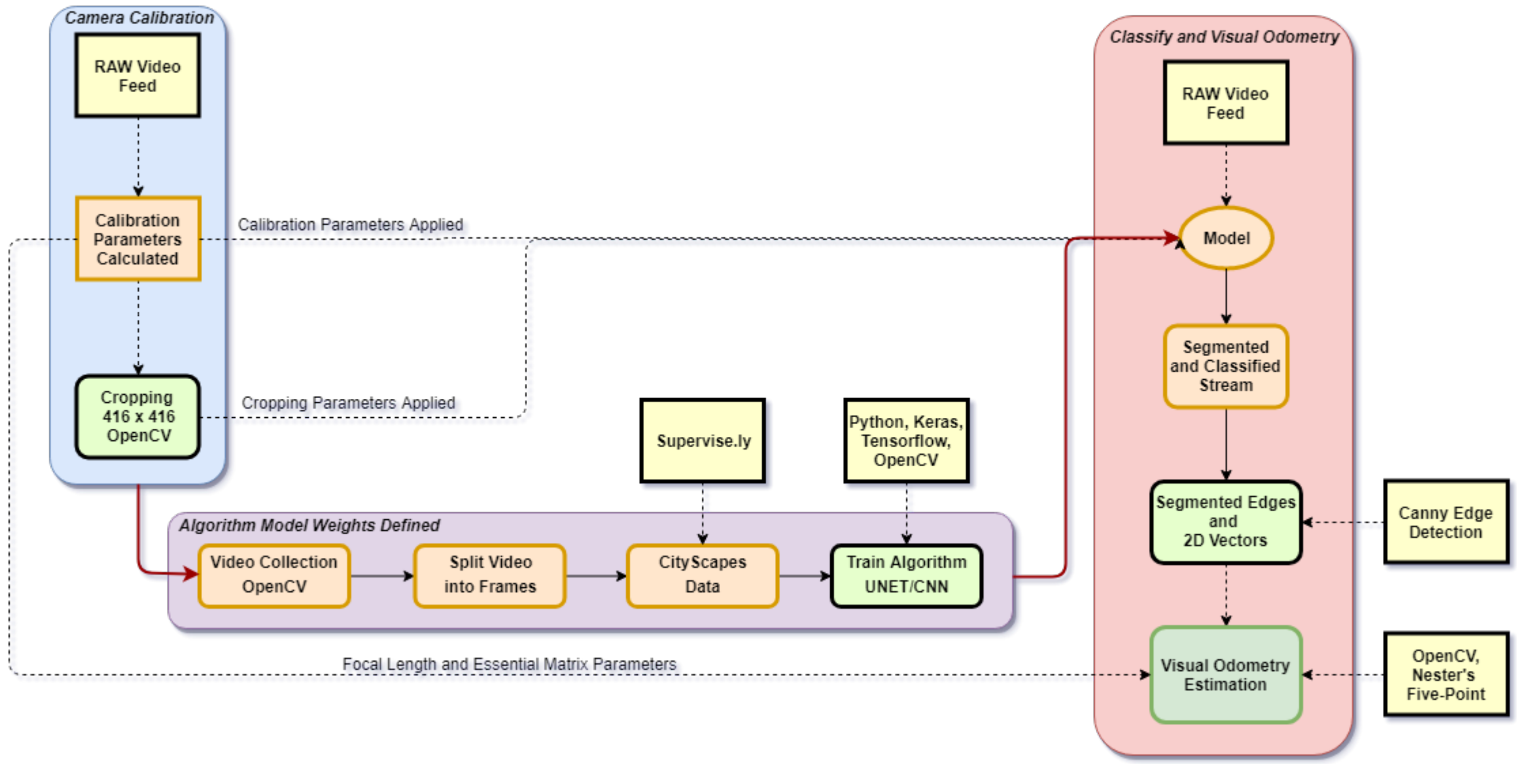

Camera Calibration
RAW Video Feed
Calibration Parameters Calculated
Cropping 416 x 416 OpenCV
Calibration Parameters Applied
Cropping Parameters Applied
Algorithm Model Weights Defined
Video Collection OpenCV
Split Video into Frames
CityScapes Data
Train Algorithm UNET/CNN
Supervise.ly
Python, Keras, Tensorflow, OpenCV
Focal Length and Essential Matrix Parameters
Classify and Visual Odometry
RAW Video Feed
Model
Segmented and Classified Stream
Segmented Edges and 2D Vectors
Canny Edge Detection
Visual Odometry Estimation
OpenCV, Nester's Five-Point

Figure 3 Overview of Methodology

2.2 Camera Calibration

Latency is the period it takes for an image to be captured and processed. Real-time processing is a high priority, and video streaming is one area where this could hinder the process. Realizing this issue, we needed to find a method that would both have low latency while maintaining the video quality. The solution for this was to use the OpenCV library to collect video stream. At 30 frames per second(FPS), no delay was present.

An essential part of any research involving video capture is having the camera calibration parameters, as this can introduce errors if not performed. In this section, the method used for camera calibration is presented. Image distortions can be introduced via irregularities in both camera sensors and lenses. Errors caused by these distortions will propagate the level of uncertainty throughout the entire procedure. In this section, the intrinsic calibration method is explained. Briefly stated, the distortions were taken care of by calibrating the camera using the OpenCV library similarly to (Y. M. Wang, 2010). The measurements of the camera lens independently and concerning the camera sensor are taken into consideration in order to find the intrinsic parameters.

Principally, two types of distortions occur within a camera, radial distortion, and tangential distortion. Radial distortion occurs when the rays of light entering the frame bend more at edges than in the optical center (Zhang, 2000). Figure 4 Radial distortion is a common issue with a pinhole style of camera, like the one being used for this research. Tangential distortion takes place when the lens is not perpendicular to the sensor of the camera. Often with lower quality cameras, tangential distortion occurs if the glue that holds the sensor is placed incorrectly or the lens does not seat completely, thus causing the sensor and lens not to be parallel.

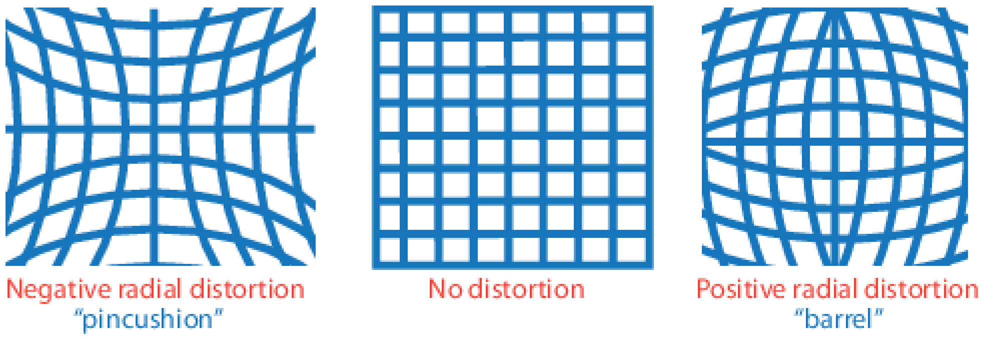


Figure 4 Radial Distortion, Source: www.mathworks.com

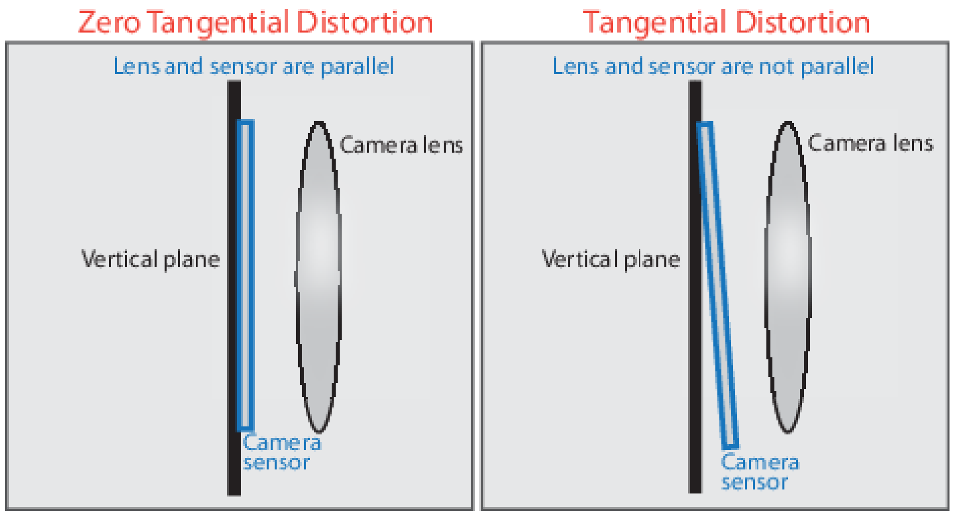


Figure 5 Tangential Distortion, Source www.mathworks.com

To correct the distortions first the existing conditions, need to be extracted using known parameters. The Radial distortion occurs when the lens of the camera is curved. For the radial distortion factors, equations 1 and 2 were used.

$$X_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \qquad (1)$$

$$Y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \qquad (2)$$

$X_{corrected}$ and $Y_{corrected}$ are the corrected output for each of the previous x and y pixels respectively.

k₁, *k₂*, *k₃* are the symmetric radial distortion coefficients, and *r* is the collinearity equation terms.

As previously mentioned, when the camera lens is not perfectly parallel to the face of the CCD sensor, tangential distortions occur. It is correct using equations 3 and 4.

$$X_{corrected} = x + [2p_1xy + p_2(r^2 + 2x^2)] \tag{3}$$

$$Y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2xy] \tag{4}$$

p_1, p_2 are the decentering distortion coefficients.

Finally using equation 5, we calculated the unknown focal length. (Camera Matrix)

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5}$$

Using the homographic coordinate system ($w = z$).

The unknowns f_x and f_y (focal lengths in *x* and *y*) and c_x and c_y locates the center of the camera sensor in pixel coordinates. If we use *a* as the aspect ratio and then $f_y = f_x * a$. We then have a single focal length *f*. Worth mentioning, in this research lens in use, is a fixed focal length.

During the calibration stage, it was recommended to have a minimum of 18 images. However, after testing with only 18 images, the radial distortion was still visible. After it was tested up to 100 images in intervals of 10 and after 50 images were used for calibration, no difference could be seen. Also, the correction parameters had minimal changes after 50 images. Even worth noting that after 30 images the time for processing began to increase rapidly. This process took a total of three hours; this includes testing the various parameters and implementation of the Python code. This step was performed for both cameras. This procedure was performed with the OpenCV 3.3.0 library using checkerboard patterns (OpenCV Dev Team, 2014).

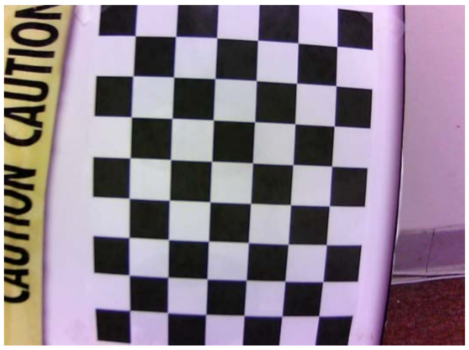


Figure 6 Uncalibrated Image

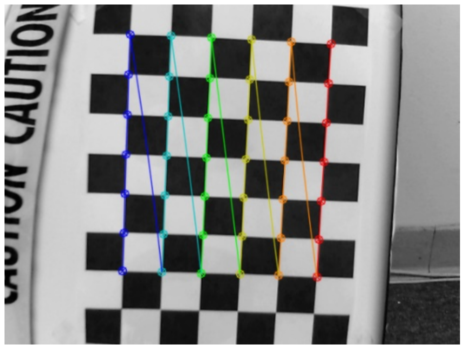


Figure 7 Calibrated Image

Now that the intrinsic parameters of the camera are known (focal length *f*, the pixel size *s*, and the image center O_x and O_y), setup and testing for field collection are performed. These parameters are then implemented when video or image collection is performed.

One of the main limitations of the algorithm being used for this study is that the input image height and width must be equal. Using OpenCV, two methods are available to solve this issue. The first option is to resize the image. This method performs a geometric image transformation and has increased the time for processing over the image crop method. The cropping method is just that; it crops off the image using the predefined values defined by the user without slowing down the overall process.

Upon successful video feed and record test in the lab, the cameras were then mounted to the vehicle for field collection. During the field collection or video recording, permission was granted from the university police department. It was conducted over winter break, so that traffic interference was minimal. Using a 3D printer, custom mounts were created for the cameras to be mounted to a piece of steel channel. Using strong magnets, the steal channel was held in place in the front bumper of a vehicle. Long USB cables routed from the cameras to the interior, where the laptop was located. All video capture was performed also using OpenCV.

Although there are video and image processing techniques to remove environmental occlusions and distortions, careful thought was applied for selecting the time of day when the video was recorded for training. The objective of this research is not to process these types of distortions. Mid-day was chosen to eliminate any introduction of lighting difficulties. After all field capture was completed, a total of 22 minutes of road-related video was recorded.

2.3 Data Preparation and Augmentation

A fundamental part of all machine learning is preparing the data. In this section, the data preprocessing methods that were incorporated are discussed. There are many datasets they are designed to contribute for computer vision purposes. Cityscapes is a benchmark suite and large-scale dataset to train and test approaches for pixel-level and instance-level semantic labeling (Marius Cordts, 2016).

Once images are downloaded, they are then uploaded to the web-based software Supervisely, an image processing tool (Supervisely, 2017-2018). Supervisely provides various tools for image data preparation for machine learning purposes. Each image needs to be resized, tagged for training or validation, and the classes determined. Data augmentation is essential to train the network the desired invariance and robustness properties when only few training samples are available (Li, et al., 2017). Finally, with the Supervisely tool, a .json file was used to organize, export and create the download package. The output file consists of the segmented images in a .png format, a .json file that contains the descriptions for each color in the segmented images, the original image, and finally a ground truth image that is substantially three images in one. The ground truth contains from left to right, the original image, the segmented image, and the original image with a transparent overlay of the segmentation.

2.4 Image Segmentation

Convolutional Networks also known as CNN employ a specialized linear math operation known as a convolution. In a less complicated form, CNN's are neural networks that use a convolution in at least one of the layers in place of general multiplication (Goodfellow, et al., 2016). In each one of the CNN layers, there are usually three stages. The first stage consisting of the input, where multiple convolutions are performed in parallel in-turn produces a set of linear operations. The second stage is occasionally referred to as the detector stage, runs the set of linear operations through a non-linear activation function, for example, a rectified linear (ReLu). The third stage acts a method of grouping or pooling stage. In the pooling stage, the output is further modified by using a series of nearby statistical values to produce the results in preparation for being the next layers input. These layers will repeat as much as the data scientist sees fit until there will be a final output layer for classification and show in Figure 8.

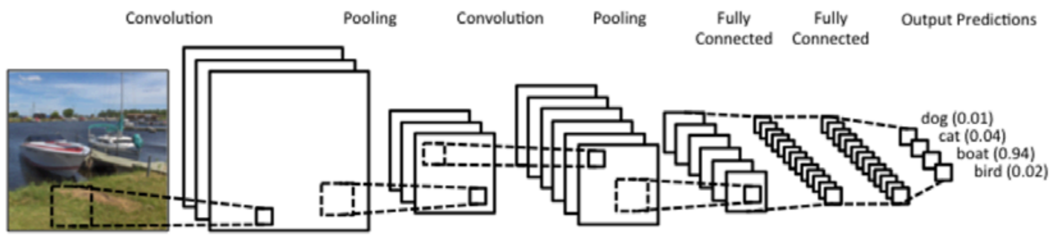


Figure 8 CNN Overview, Source: Deep Learning

Although the general basis of a CNN is usually similar, there are various other frameworks. A U-NET style architecture was first introduced in (Ronneberger, et al., 2015). It is primarily agreed on that for successful training of deep learning networks requires many thousand annotated training samples (Ronneberger, et al., 2015). They presented a novel method that relies heavily on the data augmentation to use the annotated training data more efficiently using fewer data. Their motivation is due to the limited amount of biomedical data, as it is challenging to acquire. This type of network proved to be successful. Only using 30 images with size 512x512, they were able to achieve 92.03% accuracy. Unlike a standard CNN being Feedforward, with U-NET each layer input segmentation mask is then concatenated to its corresponding mirrored output layer as shown in Figure 9.

The framework that was used for the video/image classification and segmentation portion of this manuscript uses a U-NET style CNN. This type of network was selected due to its performance and efficiency. When first introduced by Ronneberger, et al. (2015), only a small number of images were used. The type of network overcomes the difficulties of large datasets.

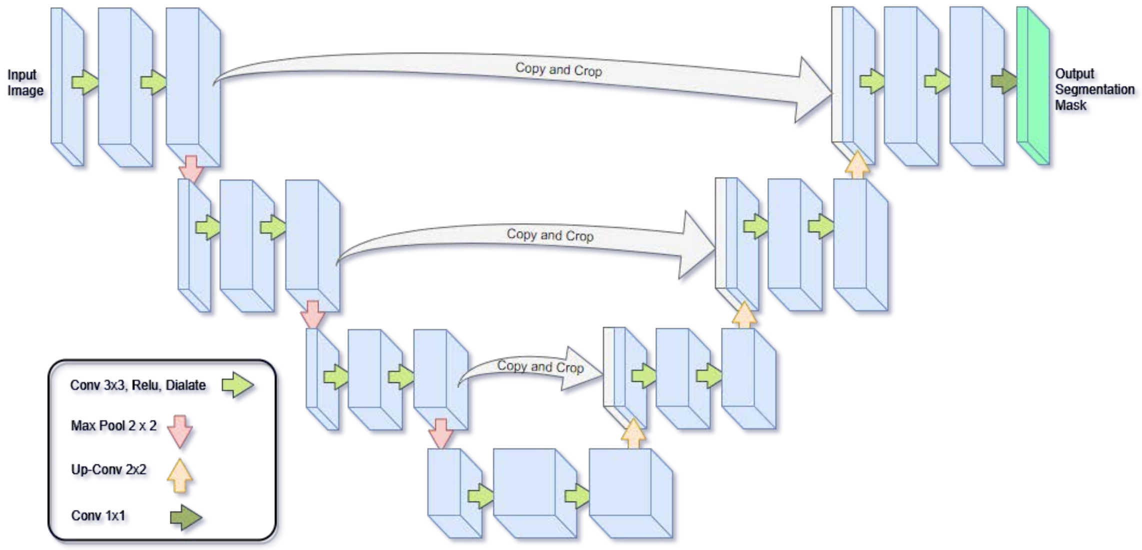


Figure 9 Simplified U-NET Architecture

The training of deep learning segmentation typically takes more time than conventional CNN's trained for object identification. The extended training time is due to the comparison of each convolution between the images and the masks annotations. Another limitation of the U-NET is the requirement of square images. However, even with these limitations, the advantage of the accuracy outweighs for this research.

Much like (Ronneberger, et al., 2015) and (Erdan, et al., 2016), a UNET style CNN was used for this research. However, modifications were made to increase the performance outcomes. In this section, a modified version of U-NET is presented as DilatedU-NET. Some of the other modifications include adding padding and creating a deeper network. However, the most significant contribution to deep learning is the implementation of dilated convolution layers for the downsampling portion.

During the convolutional portion of the network, padding can be disabled. However, if padding is disabled, the output image will not be the same size as the input image. Permanently padding adds a placeholder type border around the image to ensure the output size equals the input. As the final product needs as much data to reconstruct the 3D scene, the choice of not using the padding was not an option. Figure 10

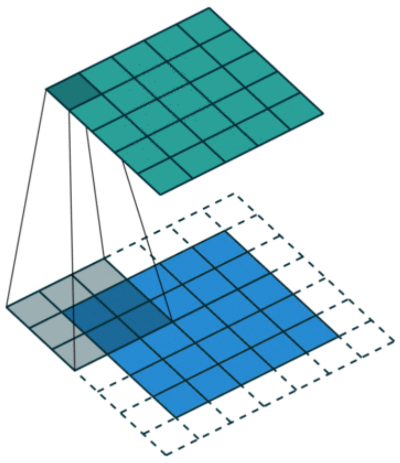


Figure 10 CNN with Padding

As the motivation is to increase the speed for the estimation of the visual odometry. Therefore acute accuracy is desired. It was for this reason that more layers were added. DeepU-NET was presented for segmentation of remote sensing images (Li, et al., 2017). They used a total of 42 layers to classify land and water. Using their dataset, they compared their method with that of SegNet (Badrinarayanan, et al., 2015) and the original U-NET (Ronneberger, et al., 2015). The DeepU-NET outperformed both.

Using Atrous Spatial Pyramid Pooling or Dilated Convolution for deep learning based image segmentation has shown to decrease noise when edges are created for segmentation (Badrinarayanan, et al., 2015) and (Chen, et al., 2017). Atrous Spatial

Pyramid Pooling selects the next nearest pixel or the next-next most adjacent pixel during the downsampling portion of the Convolutional Neural Network. Figure 11

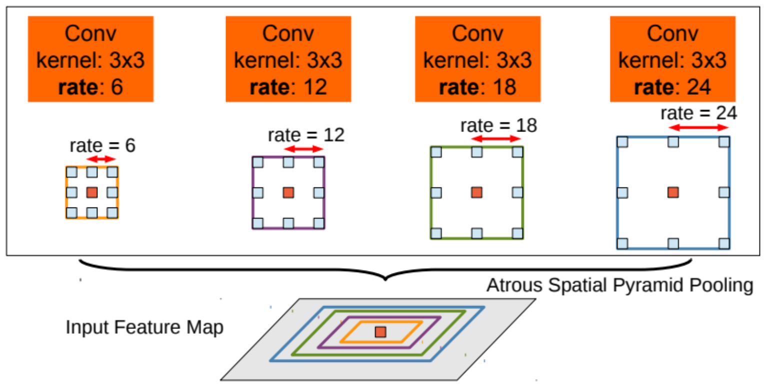


Figure 11 Dilated Examples

The overall network framework illustrated in Figure 9 consists of a downsample paths on the left and upsampling paths on the right. The downsampling paths follow the stages of a CNN composed of two 3x3 dilated convolutions with padding, detector stage uses a rectified linear (ReLu), and finally, the grouping stage uses a 2x2 max pooling operation. Figure 12 The number of feature channels doubles at each downsampling step. Each step of the upsampling consist of a 2x2 up-convolution, halving the number of feature channels, the concatenation with the corresponding copy and cropped feature, two 3x3 convolutions, followed by a ReLu. Once at the final layer, a 1x1 convolution is used to map each component to the final output using a SoftMax activation.

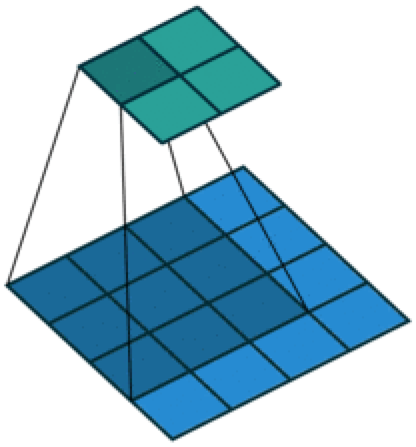


Figure 12 CNN Max Pooling, Source: Deep Learning

Deep learning algorithms need various methods of optimization. The job of either minimizing or maximizing some function *f(x)* by altering *x* is known as an optimization (Goodfellow, et al., 2016). Another requirement for the algorithm combined with this type of data is to have a type loss function where the error is zero when there is no error and get larger the as the error increases. Binary cross entropy functions are able to fulfill this need. The binary cross entropy loss function is a type of logistic regression model. This type of loss functional was also used with the U-NETs used by Li, et al. (2017) and Erdan, et al. (2016).

Only one of the two terms will matter at a time; therefore, it is referred to as a binary. The target is either one or zero If the objective is one only the first term is imperative, whereas if the goal is zero then only the second term matters. Since *log(y)* will never be positive, the whole is multiplied by a negative.

$$t = target$$

$$y = output\ of\ logistic$$

$$\log(y) = -\infty\ to\ 0$$

$$Total\ Loss = -\sum_{n=1}^{N} t_n \log(y_n) + (1 - t_n) \log(1 - y_n)$$

Finally, the total error needs to be calculated to optimize over all the training data simultaneously. This is accomplished by summing over all the individual errors that give the sum over all the individual cross entropy.

Probabilities that are associated with a multinomial distribution is often performed using a SoftMax prediction. Fundamentally, multinomial distribution involves the prediction of two or more outcomes, i.e., this research

$$softmax(x) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_i)}$$

2.5 Extract 2D Vectors

In this section, the method for extracting the edges from the semantic images is presented. In many practices, the classification of the edges is lost during edge extraction. The technique being presented preserves the classification and carries it through to the visual odometry portion. Starting with the classified segmented images, each class is represented by a unique color and is explicitly assigned these colors by the DilatedU-NET model. For each category, a threshold is created. All classified segments are selected in parallel. Once selected, each is converted to greyscale using OpenCV (OpenCV Dev Team, 2014). Each class is temporally stored in the temporary memory. The edges are then extracted from each segment using a Canny Edge (Canny, 1986) detection algorithm and converted into 2D polylines utilizing the find contours command in OpenCV.

2.6 Feature Detection and Visual Odometry

Similar to how humans can recognize edges or corners is second nature, feature detection needs to be accurate and speedy. Visual odometry and feature detection work in conjunction with each other. Unlike a bundle adjustment, visual odometry is used primarily for advancing moving image sequences, i.e., video to obtain the position and rotation of the camera. Visual odometry and a bundle adjustment are similar in the sense that they both solve for the location of the camera positions. However, a bundle adjustment is designed for more significant differences in camera positions and can compensate for using different types of cameras. Visual odometry was created when there are only slight changes in camera positions, i.e., video image sequences can be used for real-time applications (Hartley, et al., 2003).

The KITTI dataset was used to maintain a standard point of reference for the estimated visual odometry. The KITTI dataset is benchmark data that contains twenty-two stereo sequences. It also provides ground truth data for eleven of the sequences. For our purposes, only the left view of the was used.

Much consideration was taken when selecting the algorithm for feature extraction. Based on the literature reviewed, using the ORB algorithm give the impression to be the preferred method. However, the evaluation of both the ORB and the FAST were performed using the KITTI dataset (Vision meets Robotics: The KITTI Dataset, 2013).

During testing a visual way to locate the error is to view a ground truth map and look for drift; i.e., Figure 19 and Figure 20. Drift occurs when the feature detector selects points or landmarks in frame one and the predicted corresponding points in frame two.

There can be some small errors in the difference of location estimation. As the frames advance, so does the error propagate and gives the appearance of drifting off course.

Principally the methodology for estimating the visual presented in (Garc´ıa, et al., 2012) is what was used for this portion. However, ORB and FAST feature detectors were used instead of SURF and SIFT. In this research, we compared the original images, the color segmented images, and the 2D lines extracted from the segmented images to one another using both ORB and FAST.

3. Experiment Implementation and Results

3.1 Hardware and Software

Prior to performing any experiments, consideration for the types of hardware and software must be taken. Majority cameras sold today are digital and are also incorporated into many devices ranging from smartphones to automobiles. Before camera selection begins, some criteria were set and are listed,

- ✓ Fixed Focal Length
- ✓ 5 Megapixels or Greater
- ✓ Each Camera Cost less than $50.00USD
- ✓ Linux/Windows Fully Compatible
- ✓ Transfer Data/Powered by USB

Based on this criterion we selected the "ELP 5 Megapixel" sensor with fixed focus lens of 2.8mm, costing $45.00USD. Two cameras were purchased for redundancy. Though not labeled as such, these are a pinhole style of camera. Pinhole cameras use a small sensor. However, due to the size, only a limited amount of light can be sensed. To alleviate this issue, a highly curved lens is added, this is not as extreme as a fisheye style of lens. Though a significant amount of radial distortion is introduced, this was accounted for in 2.2 Camera Calibration

Also, a powerful computer with a robust GPU was necessary for training the algorithms. A custom-built lab computer containing an Intel Xeon processor, 128gb of memory, 256gb SSD, and an NVidia GeForce Titan Black GPU proved to be more than enough for the training. A Dell laptop with an Intel I7 CPU, 16gb of memory, 1TB SSD, and an NVidia GeForce GTX970 was used for the mobile applications. This laptop was provided by other sources.

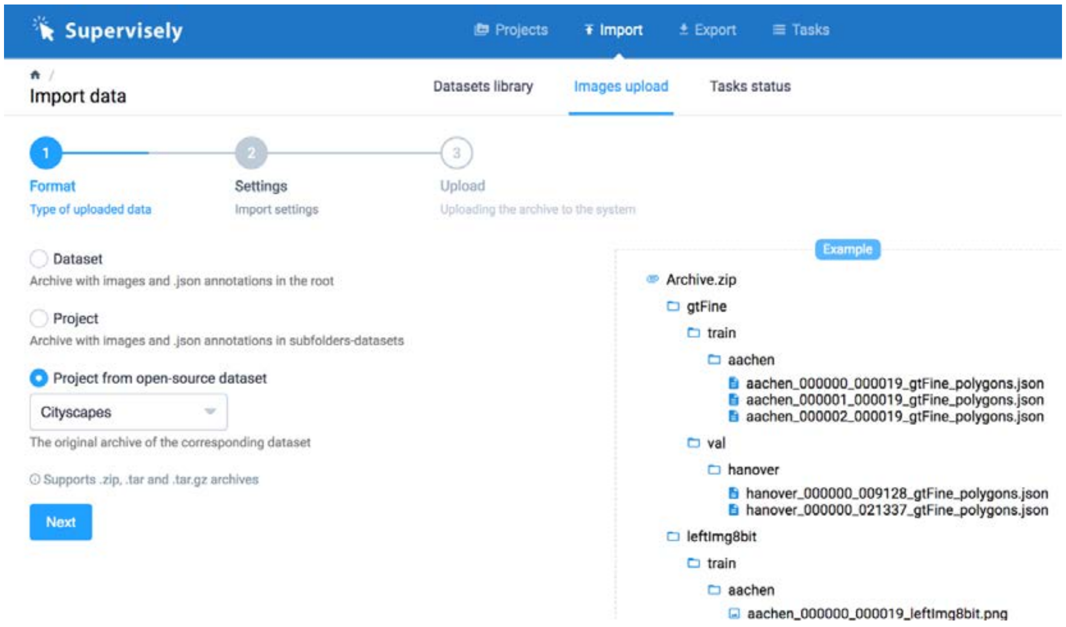
Ubuntu 16.04LTS is the operating system used. The crucial software and libraries used were Python 3.5 (Python Software Foundation), OpenCV 3.3.0 (Itseez, 2017), Keras 2.1.4 (Chollet, 2015), TensorFlow 1.4 (Martín Abadi, 2015), and Numpy 1.14 (Anon, 2018).

3.2 Image Preprocessing

In this section, the setup and results of the experiments are discussed. Prior to downloading the Cityscapes dataset, an account needed to be approved by the Cityscapes team. This took four days after registration to obtain approval. Upon approval, “gtFine_trainvaltest.zip” and “leftImg8bit_trainvaltest.zip” were downloaded. The segmented annotations are contained in “gtFine_trainvaltest.zip” and the images in “leftImg8bit_trainvaltest.zip”.

Before uploading to Supervisely, initial preparation steps were taken, so that the Supervisely web tool would recognize the data accordingly. Both files must be extracted, then, the folder “leftImg8bit” within the “leftImg8bit_trainvaltest.zip” and the “gtFine” folder within “gtFine_trainvaltest.zip” must be compressed into a new folder. Once the new compress file is created, it can then be uploaded to the Supervisely web tool as shown in Figure 14.

Upon files being uploaded, the images were then resized, tagged for training and validation as shown in Figure 15. Due to the limitations of the purchased cameras, the image size was resized to 416 X 416. Then using these settings, they were exported and downloaded for training purposes. To simplify the experiments, the classes were reduced from 31 to 5 to include road, vegetation, sky, lamp post, and vehicle classes. Only 3 of the cities were used for the training, consisting of 230 for training, 100 for validation, totaling 330 images comprised of the Cityscapes dataset.



Import data

1 Format — Type of uploaded data
2 Settings — Import settings
3 Upload — Uploading the archive to the system

Dataset
Archive with images and .json annotations in the root

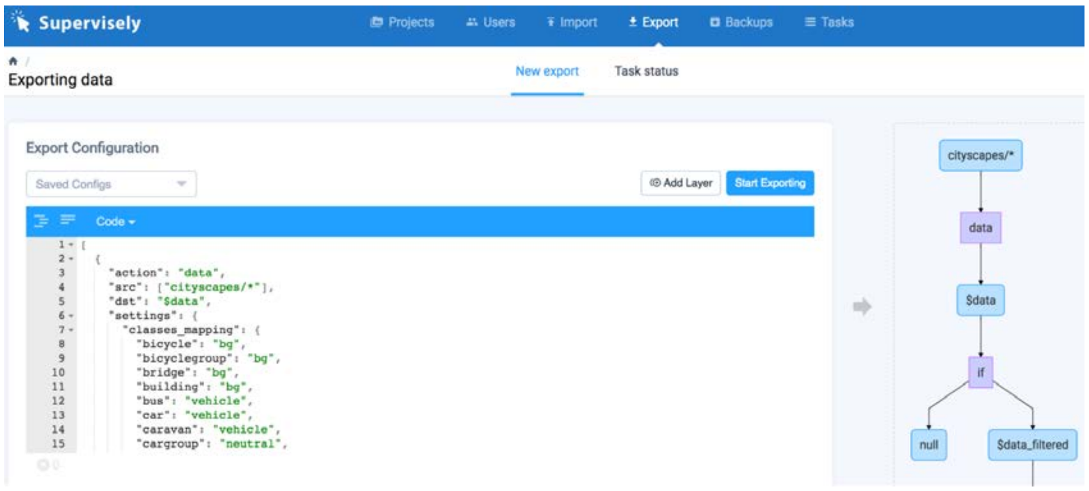
Project
Archive with images and .json annotations in subfolders-datasets

Project from open-source dataset
Cityscapes
The original archive of the corresponding dataset

Supports .zip, .tar and .tar.gz archives

```
Archive.zip
  gtFine
    train
      aachen
        aachen_000000_000019_gtFine_polygons.json
        aachen_000001_000019_gtFine_polygons.json
        aachen_000002_000019_gtFine_polygons.json
    val
      hanover
        hanover_000000_009128_gtFine_polygons.json
        hanover_000000_021337_gtFine_polygons.json
  leftImg8bit
    train
      aachen
        aachen_000000_000019_leftImg8bit.png
```

Figure 13 Supervisely Project Creation



Exporting data

Export Configuration

```
[
  {
    "action": "data",
    "src": ["cityscapes/*"],
    "dst": "$data",
    "settings": {
      "classes_mapping": {
        "bicycle": "bg",
        "bicyclegroup": "bg",
        "bridge": "bg",
        "building": "bg",
        "bus": "vehicle",
        "car": "vehicle",
        "caravan": "vehicle",
        "cargroup": "neutral",
```

Figure 14 Supervisely Image Preprocessing

3.3 Deep Learning Results

Using KERAS with TensorFlow on the backend, the DilatedU-NET script was created. When training the network, two different methods were evaluated. Type 1, non-dilated and Type 2 containing the dilated layers. The separate models were created to evaluate if adding dilated layer would increase the accuracy while decreasing the processing time within the DilatedU-NET.

Other than the dilated layer, both Type 1 and Type 2 settings remained constant. The learning rate was set 0.00001, batch size set to eight and a total of 1000 epochs. Five models from each of the compared algorithms were saved during the training at 100, 250, 500, 750, and 1000 epoch intervals. The entire time for each training session was a little over 26 hours.

Accuracy for Type 1, the standard U-NET was 81.6420% whereas the accuracy for DilatedU-NET was 84.3839% as shown in Table 1, Figure 16, and Figure 17. An example DilatedU-NET processed frame is shown in Figure 18.

Table 1 Model Output Metrics

	Validation Accuracy		Frame Rate(fps)	
Epochs	DilatedU-NET	U-NET	DilatedU-NET	U-NET
100	76.4898%	74.8880%	7	5
250	81.9801%	78.9968%	7	5
500	83.9027%	80.3077%	7	5
750	84.3839%	81.6420%	7	5
1000	83.8285%	80.8482%	7	5

When testing for the framerate comparison, the DilatedU-NET also outperformed the U-NET with seven frames per second and five frames per second respectively. Framerate testing of the model was performed using the Dell laptop.

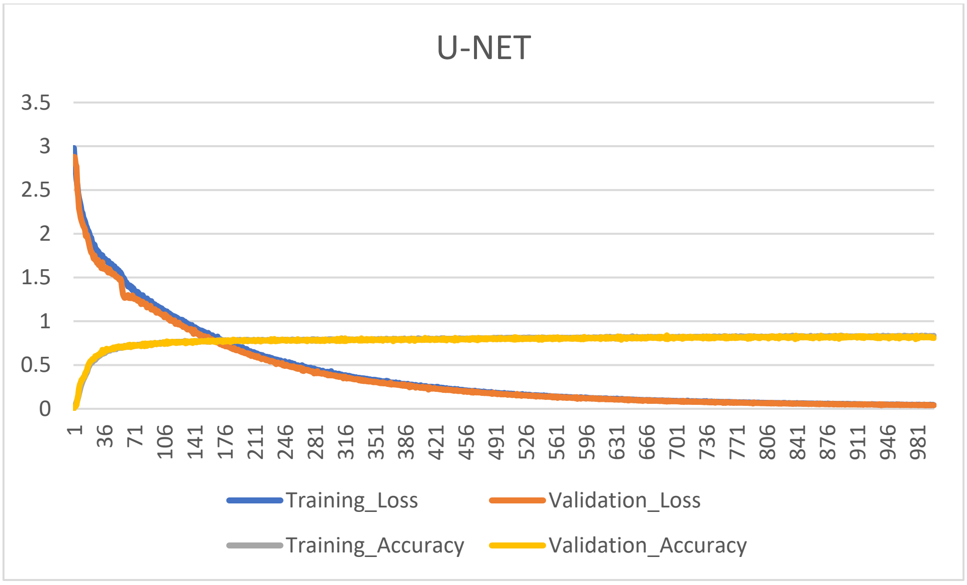


Figure 15, U-NET Training Curves

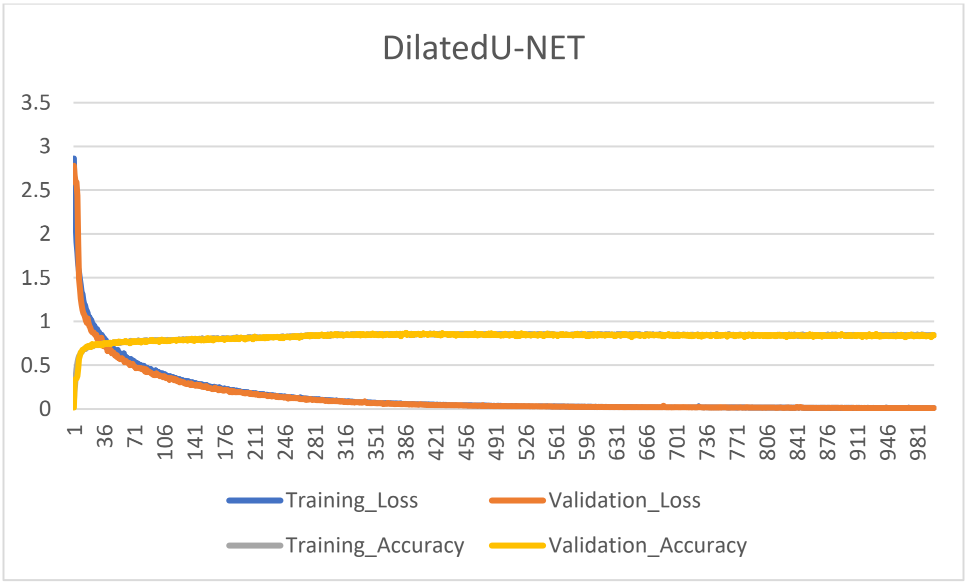


Figure 16, DilatedU-NET Training Curves

3.4 Visual Odometry Results

In this section, the implementation of the 2D edge extraction along with results of the estimated Visual Odometry are presented. Many libraries are available for use with the Python programming language. With OpenCV, canny edge detection is one of the tools available. Each one of the segmented classes has a predefined unique color assigned to them. Knowing these colors, a threshold was used to extract only the road edges. A Gaussian blur was also tested, however, the results of the output proved better not use the Gaussian blur before the canny edge detection. The edges produced by the canny edge detection do not produce an actual line, but a series of points. Therefore, find contours also from OpenCV was used to convert the edges into lines for 2D extraction.

The baseline was set using the KITTI benchmark. The odometry color dataset contains the images in sequences; the ground truth poses provide the coordinates of the location when the images were acquired, and the calibration files include the parameters of the camera that was used for obtaining the images.

When testing the proposed method of using the edges to estimate the visual odometry, the Root Mean Square Error (RMSE) of Euclidean distance between the ground truth and the estimated location was used to determine the blunder. Firstly, we compared FAST and ORB, to each other to determine which would be the best baseline. The frame rate for FAST was eight frames per second, whereas the ORB outperformed with nine frames per second. However, they both performed identically with regards to accuracy as shown in Table 2. Using “sequence 00” and the ORB feature detector, the standard baseline was determined. Secondly, we used the segmented edges and the extracted lines using both FAST and ORB. The speed of the segmented edges was noticeably faster at 43 frames per second, however also very evident was the error of 14.412m. Figure 19 and Figure 20 are plots of the estimated path location in red versus the ground truth location in green. More specifically Figure 19 is a plot of the existing method described in (Garc´ıa, et al., 2012) using the ORB as feature extractor. Whereas in Figure 20, displayed is the plot of the ground truth versus using the segmented images for the source of the ORB feature extractor.

Table 2 Visual Odometry Results

	ORB	FAST	Segmented Edge(ORB)	Line(ORB)	Segmented Edge(FAST)	Line(FAST)
RMSE(m)	2.444	2.444	14.412	26.654	14.412	26.654
Framerate(fps)	9	8	15	4	10	4

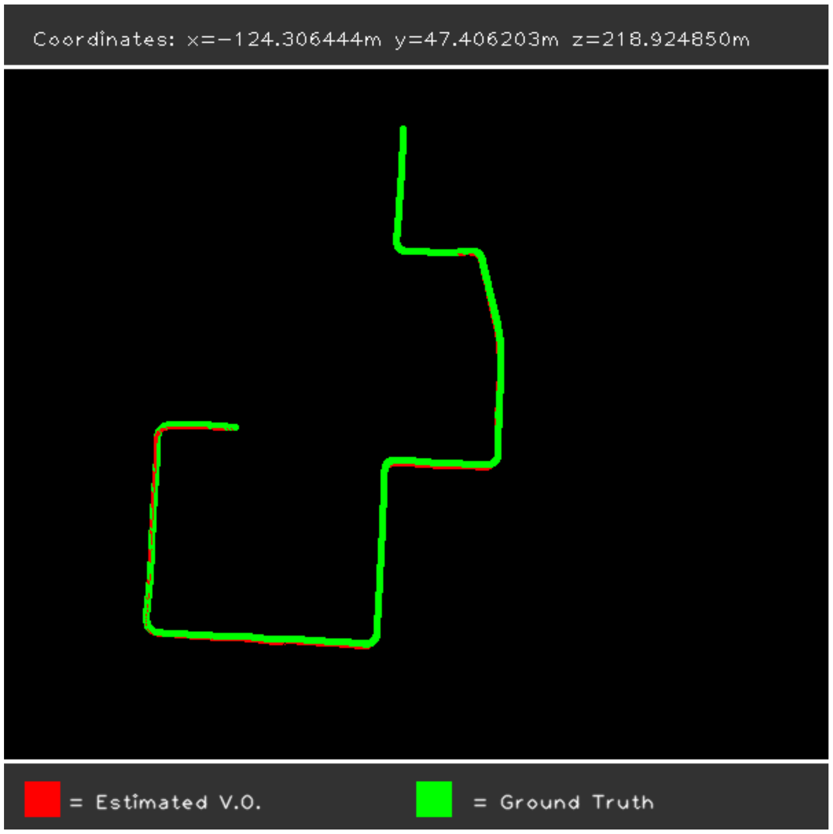


Figure 17 ORB & FAST using existing methods compared to ground truth

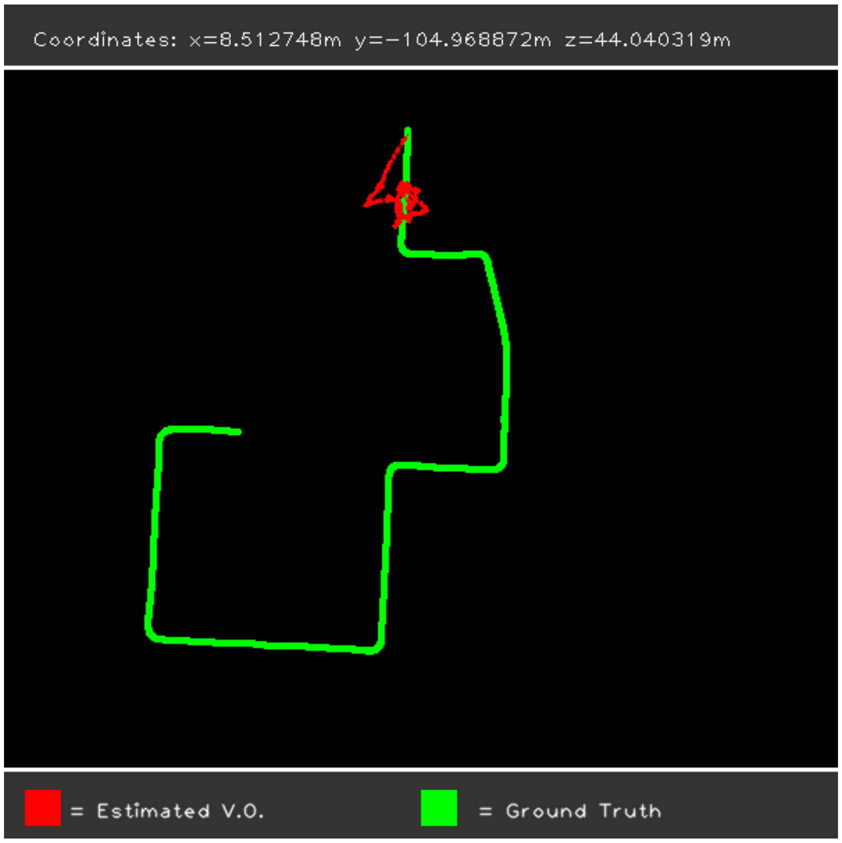


Figure 18 ORB and Segmented Image compared to ground truth

3.5 Field Collection Results

In this section, the results of processing the field collected data are presented. The process for capturing and calibrating the video frame in real-time was discussed in 2.2 Camera Calibration and 3.2 Image Preprocessing. The pre-calibrated field video was processed using the DilatedU-NET. The overall results of the DilatedU-NET segmentation and classification algorithm are displayed Figure 18. There are many variances between the type of data the DilatedU-NET was trained on and the field collected data; one being the color of the road appears. When processing the field collected video, it was noticed that large portions of the road were misclassified as vehicle class. Since no ground truth data is available to use a reference, the field collected video was not processed through the visual odometry method presented in this research.

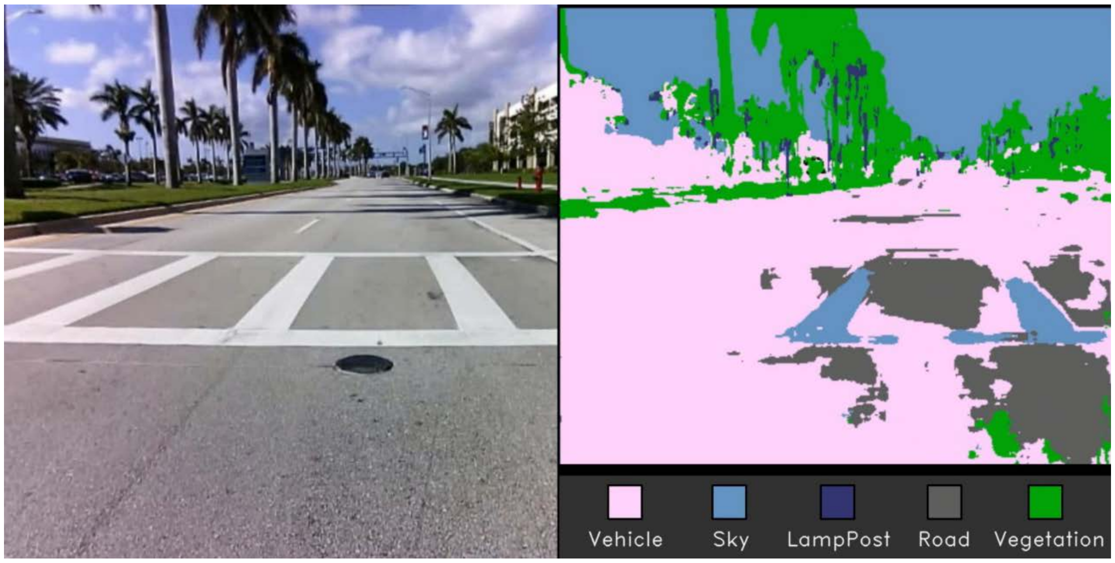


Figure 19 Field Capture Classification, (Left Image Calibrated Video Frame, Right Image: Process Framed Results via DilatedU-NET)

4. CONCLUSIONS AND RECOMMENDATIONS

We presented a novel method to estimate the visual odometry of a monocular camera setup that uses segmented images. This method could process 15 frames per second using the segmented images. However, the accuracy was subpar. Though the accuracy performance of the visual odometry estimation was not adequate, it is encouraging to research this method further due to the speed achieved. Due to the framerate achieved, this provides a significant advantage to future computer vision research. Also, a new image segmentation algorithm was introduced as DilatedU-NET. This algorithm was used to prepare the segmented images that were used for the visual odometry portion. DilatedU-NET provides a more accurate segmentation over the existing U-NET 84.3839% and 81.6420%, respectively. DilatedU-NET also processes images 1.4x faster than U-NET, proving to be more robust when performing image segmentation and classification. Although DilatedU-NET was primarily used for creating the data used for a visual odometry estimation, it displays excellent potential to be used in many other areas of the computer vision field.

Bibliography

A Comparative Study on Extraction and Recognition Method of CAD Data from CAD Drawings. **Jabal, M. F. A., Rahim, M. S. M., Othman, N. Z. S., & Jupri, Z. 2009.** 2009. 2009 International Conference on Information Management and Engineering.

Aboelmagd N, Karmat TB, Georgy J. 2013. *Fundamentals of inertial navigation, satellite-based positioning and their integration.* 2013.

—. 2013. *Fundamentals of inertial navigation, satellite-based positioning and their integration.* 2013.

American Society of Photogrammetry. 1980. *Manual of Photogrammetry.* Falls Church, Va : American Society of Photogrammetry, 1980.

Anon. 2018. NumPy: a numerical extension for the computer language Python. Version 1.14. s.l. : The Regents of the University of California, 2018.

Badrinarayanan, Vijay, Handa, Ankur and Cipolla, Roberto. 2015. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling.* 2015.

Bailey, Tim and Durrant-Whyte, Hugh. 2006. Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics and Automation Society.* 2006, pp. 108-117.

Ballardini, Augusto Luis, Cattaneo, Daniele and Fontana, Simone. 2016. *Leveraging the OSM building data to enhance the localization of an urban vehicle.* Rio de Janeiro, Brazil : IEEE, 2016.

Bay, Herbert, et al. 2007. *Speeded-Up Robust Features (SURF).* 2007.

Bhardwaj, Romil, et al. 2017. *AutoCalib: Automatic Traffic Camera Calibration at Scale.* 2017.

Bláha, Maroš, et al. 2016. *Large-Scale Semantic 3D Reconstruction: an Adaptive Multi-Resolution Model for Multi-Class Volumetric Labeling.* s.l. : IEEE, 2016. pp. 3176-3184.

Borenstein J, Everett H, Feng L. 1996. *Where am I? Sensors and methods for mobile robot positioning.* 1996.

Calonder, Michael, Lepetit, Vincent and Fua, Pascal. 2010. *BRIEF: Binary Robust Independent Elementary Features.* Berlin, Heidelberg : s.n., 2010.

Canny, John. 1986. *A Computational Approach to Edge Detection.* 1986. pp. 679-698.

Certainty3D. 2018. TopoDOT. s.l. : Certainty3d, 2018.

Chen, Liang-Chieh, et al. 2017. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs.* 2017.

Chollet, Fran. 2015. Keras. s.l. : GitHub, 2015.

E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. 2006. *Real time localization and 3D reconstruction.* 2006.

Egodagamage, Ruwan and Tuceryan, Mihran. 2018. *Distributed monocular visual SLAM as a basis for a collaborative augmented reality framework.* 2018.

Erdan, Bora, Gamboa, Noah and Wood, Sam. 2016. *3D Convolutional Neural Network for Brain Tumor Segmentation.* 2016.

Fischler, M.A. and Bolles, R.C. 1981. *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography.* 1981. pp. 381–395.

Garc´ıa, David Valiente, et al. 2012. *Visual Odometry through Appearance-and Feature-Based Method with Omnidirectional Images.* 2012.

Gomez-Ojeda, Ruben, Briales, Jesus and Gonzalez-Jimenez, Javier. 2016. *PL-SVO: Semi-Direct Monocular Visual Odometry by Combining Points and Line Segments.* Daejeon, Korea : s.n., 2016.

Goodfellow, Ian, Bengio, Yoshua and Courville, Aaron. 2016. *Deep Learning.* s.l. : MIT Press, 2016.

Harris, Chris and Stephens, Mike. 1988. *A COMBINED CORNER AND EDGE DETECTOR.* 1988.

Hartley, Richard and Zisserman, Andrew. 2003. *Multiple View Geometry in Computer Vision Second Edition.* New York : Cambridge University Press , 2003.

Heikkila, J. 2000. *Geometric camera calibration using circular control points.* 2000.

Heikkila, Janne and SilvCn, Olli. 1997. *A Four-step Camera Calibration Procedure with Implicit Image Correction.* San Juan, Puerto Rico : IEEE, 1997.

Helmick received, Daniel, et al. 2004. *Path following using visual odometry for a Mars rover in high-slip environments.* Big Sky, MT, USA : s.n., 2004.

Huang, Albert S, et al. 2017. *Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera.* 2017. pp. 235-252.

Huang, Yen-Kai and Yang, Vivian. 2017. *Street View Segmentation using FCN models.* s.l. : Stanford University, 2017.

Itseez. 2017. Open Source Computer Vision Library. s.l. : Itseez, 2017.

Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. **Kundu, Abhijit, et al. 2014.** Zurich, Switzerland : Springer International Publishing Switzerland, 2014. ECCV 2014. pp. 703-718.

Kendall, Alex, Badrinarayanan, Vijay and Cipolla, Roberto. 2016. *Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding.* 2016.

Kundu, Abhijit, et al. 2014. *Joint Semantic Segmentation and 3D Reconstruction from Monocular Video.* s.l. : Springer, 2014. pp. 703-718.

Lhuillier, M. 2005. *Automatic structure and motion using a catadioptric camera.* 2005.

Li, Ruirui, et al. 2017. *DeepUNet: A Deep Fully Convolutional Network for Pixel-level Sea-Land Segmentation.* 2017.

Lowe, David G. 2004. *Distinctive Image Features from Scale-Invariant Keypoints.* 2004.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele. 2016. *The Cityscapes Dataset for Semantic Urban Scene Understanding.* 2016.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, L. 2015. *TensorFlow: Large-scale machine learning on heterogeneous systems.* 2015.

Nist´er, David, Naroditsky, Oleg and Bergen, James. 2004. *Visual Odometry.* 2004.

Niste´r, David. 2004. *An Efficient Solution to the Five-Point Relative Pose Problem.* 2004. pp. 756-770.

OpenCV Dev Team. 2014. Camera calibration With OpenCV. [Online] 2014. http://docs.opencv.org/3.0-beta/doc/tutorials/calib3d/camera_calibration/camera_calibration.html.

Poelman, Conrad J. and Kanade, Takeo. 1997. *A Paraperspective Factorization Method for Shape and Motion Recovery.* 1997. pp. 206-218.

Python Software Foundation. Python Language Reference, version 3.5. s.l. : Available at http://www.python.org.

Rone W, Ben-Tzvi P. 2013. *Mapping, localization and motion planning in mobile multi-robotic systems.* 2013.

Ronneberger, Olaf, Fischer, Philipp and Brox, Thomas. 2015. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* s.l. : Springer, 2015. pp. 234-241.

Rosten, Edward and Drummond, Tom. 2006. *Machine Learning for High-Speed Corner Detection.* Berlin, Heidelberg : Springer, 2006. pp. 430-443.

Rublee, Ethan, et al. 2011. *ORB: An efficient alternative to SIFT or SURF.* Barcelona, Spain : s.n., 2011.

Sturm, Peter and Triggs, William. 1996. *A Factorization Based Algorithm for multi-Image Projective Structure and Motion.* Cambridge, United Kingdom : Springer-Verlag, 1996. pp. 709-720.

Supervisely. 2017-2018. Supervisely. [Online] 2017-2018. https://supervise.ly/.

Taketomi, Takafumi, Uchiyama, Hideaki and Ikeda, Sei. 2017. *Visual SLAM algorithms: a survey from 2010 to 2016.* 2017.

Tomasi, Carlo and Kanade, Takeo. 1992. *Shape and motion from image streams under orthography: a factorization method.* 1992. pp. 137-154.

Urban 3D Semantic Modelling Using Stereo Vision. **Sengupta, Sunando, et al. 2013.** s.l. : IEEE, 2013. In Robotics and Automation (ICRA). pp. 580-585.

Vision meets Robotics: The KITTI Dataset. **Geiger, Andreas, et al. 2013.** 2013, International Journal of Robotics Research (IJRR).

Wang D, Liang H, Zhu H, Zhang S. 2014. *A bionic camera-based polarization navigation sensor.* 2014.

Witten , Ian H, et al. 2017. *Data Mining Practical Machine Learning Tools and Techniques Fourth Edition.* Cambridge : Elsevier Inc, 2017.

Wolf, Paul R, Dewitt, Bon A and Wilkinson, Benjamin E. 2014. *Elements of Photogrammetry with applications in GIS.* New York : McGraw-Hill Education, 2014.

Y. M. Wang, Y. Li, J. B. Zheng. 2010. *A camera calibration technique based on OpenCV.* Chengdu, China : IEEE, 2010.

Yu, Fisher and Koltun, Vladlen. 2016. *MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS.* 2016.

Zhang, Z. 2000. *A flexible new technique for camera calibration.* 2000. pp. 1330-1334. Vol. 22.