

STEREO VISION-BASED VEHICULAR PROXIMITY ESTIMATION

BY ARJUN KRISHNA

A thesis submitted to the
Graduate School—New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Master of Science
Graduate Program in Electrical and Computer Engineering

Written under the direction of
Prof. Janne Lindqvist
and approved by

New Brunswick, New Jersey

October, 2014

ABSTRACT OF THE THESIS

Stereo Vision-based Vehicular Proximity Estimation

by Arjun Krishna

Thesis Director: Prof. Janne Lindqvist

This thesis describes an innovative and cost effective method to develop a low-cost following distance logging algorithm for volunteer participants which will allow quantitative research in driving behavior. Sparse stereo depth estimation methods along with a license plate localization algorithm has been used in order to achieve this. The depth is estimated by processing the video feeds from the stereo camera setup mounted inside a car looking out of the front window. License plate localization is used as a means to localize the position of the car within the image. Depth is estimated from the disparity, which is calculated using the rectified images of the frames from both the video streams.

Acknowledgements

First and foremost, I would like to thank my advisor, Prof. Lindqvist, for giving me the opportunity to be a part of this project and also for his constant guidance and support. I would also like to express my heartfelt gratitude to my parents for their support. I would like to extend a big thank you to the enthusiastic people in my lab for their help in the testing and other aspects of this thesis, without which it would have been extremely difficult to complete the thesis. Also I am really grateful to Prof. Lindqvist and WINLAB, for placing their trust in me and providing me with an assistantship.

This material is based upon work supported by the National Science Foundation under Grant Number SoCS-1211079. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

Abstract	ii
Acknowledgements	iii
List of Tables	vi
List of Figures	vii
1. Introduction	1
1.1. Motivation and goal	1
1.2. Overview	2
1.3. Outline of the document	3
2. Literature Survey	4
3. Technology Used	9
4. Stereo Vision Based Depth Estimation	11
4.1. Camera Calibration	11
4.2. Epipolar Geometry and Rectification	14
4.3. Sparse Stereo Correspondence	16
4.4. Depth Estimation	17
5. License Plate Localization	19
5.1. Preprocessing	19
5.2. Edge detection using the Sobel Operator	19
5.3. Morphological Close Operation	21
5.4. Thresholding	21

5.5. Extracting the Possible Candidate Regions	22
5.6. Contour detection	22
5.7. Candidate elimination	23
6. Implementation	27
6.1. Experimental Setup and Equipment Used	27
6.2. Implementation of the Algorithm	28
6.3. Data collection	31
7. Results	33
7.1. Data Collection	34
7.2. Distance estimation	34
7.3. Analysis	36
8. Conclusion and Future Work	40
8.1. Future Work	41
References	43

List of Tables

7.1. The average errors for different distances	35
7.2. Confidence intervals for the various distances estimated	35
7.3. The estimated distances at various time instances	36

List of Figures

4.1. Pinhole camera model	12
4.2. Calibration pattern as seen from both the cameras	12
4.3. Camera calibration	13
4.4. Epipolar Geometry [1]	14
4.5. Rectified Images	14
4.6. SURF Matching	17
4.7. Illustration of disparity	18
5.1. Sobel operators applied to the grayscale input image	20
5.2. Morphological close operator applied to the original input image	21
5.3. First set of candidates extracted	22
5.4. Possible candidate regions	23
5.5. The localized plate region	24
5.6. Correct Localization	24
5.7. Incorrect Localization	25
6.1. The mounts used for the cameras	28
6.2. Flowchart of the algorithm	29
7.1. Error vs. Distance	35
7.2. Distances estimated with both cars in motion	37
7.3. The estimated distances with both cars in motion	37
7.4. The estimated distances with one car stationary	37

Chapter 1

Introduction

1.1 Motivation and goal

The work presented in this thesis represents the use of stereo vision in achieving depth estimation. The main aim of this work is to develop a low-cost following speed logging algorithm for volunteer participants, which will allow quantitative research in driving behavior. The resulting distance estimates from this work, will be used for determining driving behavior related aspects such as the speed at which a vehicle follows another vehicle under various circumstances. Research in driving behavior can be used to quantify various aspects such as factors leading to accidents, driving patterns of drivers in various geographical locations, driver influence on car fuel consumption, and CO_2 emissions [2], among others. Along these lines, a number of driver measurement scales such as Driver Anger Scale [3], Driver Skill Inventory [4], etc. have been devised in order to quantify the driving behavior. This data can also be used to perform driver profiling and be used by driving schools, law-enforcement agencies, and so on. All these quantifications can result in a better and safer driving environment for everyone. The ease of availability and deployability of smartphones, and their underutilization, poses the following question: can they can be used to fuel research interests in serving a wide range of applications? Statistics show that one out of five people in the world own a smartphone and this number will only increase with time. The various sensors present in the smartphones can be taken advantage of by using them in research activities. Depth estimation and 3D reconstruction are among the most researched topics in computer vision. The stereo camera setup acts as a so-called software sensor, which is not an active sensor, as is the case with SONARs and RADARs, these require an additional source of energy and may not always be feasible to implement. Computer

vision, on the other hand, provides a whole new perspective on the given problem; it can also be extended to solve other problems without the need for any extra hardware. In this project, we use depth estimation using a stereo rig to achieve obstacle detection and obstacle avoidance, which would go a step further towards the possibility of having *driverless* cars. Other applications that can be implemented using the setup described here include driving assistance methods such as proximity alerting.

1.2 Overview

This thesis proposes a method to continuously determine the distance between a vehicle and any other vehicle that it is following. There are several ways of doing this: for example, using a laser range finder, sonar, or stereo vision. The former methods mentioned above are not very cost effective, hence in this thesis *stereo vision* is used to that end. Stereo vision involves the use of two cameras: the basic theory behind it is that the information about a scene, as seen from two vantage points, can be used to extract 3D information about the objects that are present in both the views. This is similar to the biological process of stereopsis.

To obtain an *Euclidean reconstruction* of the scene, that is to get the actual depth, the two cameras have to be calibrated. Calibration involves determining the intrinsic and extrinsic matrices of both the cameras. The intrinsic camera matrix gives us the focal length, image format, and principal point, whereas the extrinsic camera matrix specifies the spatial relationship between the two cameras. The two images are then *rectified*. Rectification is the process by which the images are rotated and translated such that the image planes of both the cameras are parallel to each other. The extrinsic camera matrix plays a crucial role in rectification. The next step in depth estimation is finding corresponding points of interest in both of the images. Rectification aids this process by making sure that these corresponding points lie along the same horizontal line, thereby limiting the search to a one dimensional line. A *license plate localization* technique is used restrict the image area in which the corresponding points have to be found. This reduces the computational complexity and speeds up the process of depth estimation. Once the corresponding feature points are found, all that remains is

determining the depth. This is done by calculating the disparity between the images. Disparity is the measure by which the corresponding points are shifted from one image to the other. Since the images are rectified, the disparity is the difference between the X-coordinates of the corresponding feature points. Once the disparity of the various interest points is obtained, the depth at each of these points can easily be calculated using the fact that disparity is inversely proportional to the depth. The average depth of all the interest points is the estimate of the distance to the vehicle in front.

1.3 Outline of the document

The various aspects involved in this thesis are highlighted in each of the chapters. Chapter 4 explains the depth estimation process, which consists of camera calibration using a checkerboard pattern, aspects of epipolar geometry and rectification of the input images, sparse stereo correspondence detection and matching, and finally the estimation of depth from the sparse disparity values obtained. Chapter 5 sheds light on the process of determining the location of the car in the image space through the use of a license plate localization algorithm. Other intermediate processes are described in smaller detail, but with the proper references, as and when required throughout this document.

Chapter 6 offers an in-depth view of the implementation of the algorithm and the experimental setup used to gather the results. This thesis, then concludes with the results obtained from the data collected along with a discussion of these results.

Chapter 2

Literature Survey

Much research is being done in order to quantify driver behavior in order to improve road safety, improve fuel efficiency [2], and in general improve the overall driving experience [5]. Alessandrini et al. [2] look into the influence of the driver behavior and driving style on the rate of CO_2 emissions and the overall fuel efficiency of the vehicle. They determined that at typical urban driving speeds, if all the drivers among the volunteer participants had adhered to the *eco-driving* driving style proposed by them, CO_2 emissions could have been reduced by up to 30%. Li et al. [5] propose the use of multiple noninvasive sensors for the purposes of the modeling of driver behavior in various real world scenarios. They focus their study on the effects of distractions caused by the various in-car technologies that are being developed.

Depth estimation has been one of the main focal points of computer vision research from the early 1980's. R. A. Jarvis [6] has explored the various approaches to range finding in computer vision. He sheds light on the advantages and shortcomings of each of these approaches. The various systems which are used for range estimation can be classified into two main types. The first classification is direct or active systems. These are systems such as ultrasonic range estimators, light time-of-flight estimation and triangulation systems, all of which involve emission of a controlled energy beam and reflected energy detection. The other type is passive systems, including monocular based range finding methods including texture gradient analysis, photometric methods, occlusion effects, size constancy, and focusing methods. Passive methods have a wider range of applications, since they do not need an external source of energy as is the case with ultrasonic and LASER range finding systems.

Strecha, von Hanses et al. [7] have studied in depth as to whether image based 3D

modeling techniques can be used to replace LIDAR and other such systems for outdoor 3D data acquisition. Although the use of LIDAR has its advantages, such as the ability to produce 3D point clouds with an accuracy of up to 1 cm, its major drawbacks are the high costs and time-consuming data acquisition. They use laser scanning (LIDAR) to obtain the ground truth and then compare it with the 3D reconstruction obtained from the stereo reconstruction. Their structure and motion pipeline consists of three steps. Firstly, sparse-feature based matching which is done using invariant feature detectors and descriptors which are applied to the input images. Secondly, from these descriptors the position, orientation, and the internal camera parameters are obtained using camera calibration techniques. Once this is done, the last step is to find dense correspondences between the two input images which have been rectified having taken into account the radial and tangential distortions in order to obtain the complete 3D model. A variance weighted evaluation shows that the best results for camera calibration deviate by σ and by around 3σ for the dense depth estimation, where σ is the standard deviation. In other words, within the horopter, the 3D modeling technique based on stereo vision has accuracy comparable to that of the LIDAR, has almost no energy requirements as is the case with an active range estimation system, and can be implemented at a fraction of the cost.

Scharstein and Szeliski [8] provide a taxonomy of existing stereo algorithms which enables their comparison, stating the pros and cons of each of them. Their analysis focuses on calibrated two-frame methods. They also propose a test bed for the quantitative evaluation of these algorithms. While their study is limited to just dense reconstruction methods, it can be used to get an idea about the performance of the sparse reconstruction methods as well since the only difference between the two is that dense reconstruction methods provide a depth estimate of the entire scene, whereas the latter provides a depth estimate at only certain points of interest in the scene. Bansal et al. [9] aim to develop a practical stereo vision sensor that is impervious to the variability of the high volume production processes and the impact of the unknown environmental conditions during its operations. They provide a framework that can perform experimental analysis that can provide an estimate of the physical bounds within which the

system is supposed to produce sufficiently accurate results for range estimation using a stereo vision sensor.

Bertozzi et al. [10] propose a different approach to the same problem being tackled in this thesis. They propose a way of locating the vehicle in front based on the considerations that a vehicle is generally symmetric, and hence can be detected by a bounding box of a certain aspect ratio size placed in a specific region of the image. The next step in their approach is to find gray level, and horizontal and vertical edge symmetries within this bounding box in order to increase the detection robustness. Once this is done they detect the bottom two corners of the a rectangular bounding box and then the top horizontal limit is searched for, localizing the vehicle. They however propose a monocular system which exploits specific features or patterns, such as texture, shape, symmetry and optical flow. As observed by Lee et al. [11], this is inappropriate for an inter-vehicle distance measurement system. Another major drawback of this approach is that there might be certain scenarios where a grouping of distant vehicles might lead to false detections. Apart from this their algorithm also suffers from high computation costs.

Lowe [12] has proposed a scale-invariant feature detection and tracking algorithm also known as SIFT, wherein a new class of object recognition system has been developed using a new class of features. These features are unaffected by changes in image scaling, translations and rotations and are also partially invariant to illumination changes and affine or 3D projection. This method works using a staged filtering approach. A Difference-of-Gaussian function is applied to the image. The first stage of filtering involves locating the maxima and minima which represent key locations in the image space. A feature vector is generated for each of these key locations such that it describes the local image region relative to its scale-space coordinate frame. The main advantage of the SIFT approach, apart from it being invariant to scaling, rotation, and translation, is that it is very fast. The current implementation of SIFT, in which each image generates around 1000 SIFT keys, takes less than a second of computation time.

Another robust feature detector, proposed by Bay et al. [13], called SURF (Speeded Up Robust Features) is claimed to be more robust, accurate, and faster than the other

interest point detection algorithms that had been previously proposed. The most important property of SURF is its repeatability, that is the property that the algorithm repeatedly finds the same interest points under different viewing conditions reliably.

Another feature point detector worth mentioning is the Harris corner detector [14], which finds the feature points based on the eigenvalues of the second-moment matrix. But the main drawback of the Harris corner detector is that it is not scale-invariant. Several other feature detectors are proposed like the one proposed by Kadir and Brady [15], which focuses on low level approaches to solving the feature detection problem, and the region detector proposed by Jurrie et al. [16], which is based on the shape rather than the texture of the objects in the scene. The downside of most of these algorithms is that they do not really focus on the speed aspect of the detection and matching of the feature points.

Juan et al. [17] do a comparative study of the three main robust feature detection methods, SIFT, PCA-SIFT and SURF. K Nearest Neighbors (KNN) and RANSAC (RANdom SAmple Consensus) are used to analyze the effectiveness of each of the above algorithms in recognition. The repeatability measurements and the number of correct matches produced are used as parameters of their comparative study. From their study it is evident that for the purposes of this thesis work, SURF was the ideal approach to be pursued. It is the fastest of the three, good at scale changes and illumination changes and is also invariant to affine transformations.

Kyto et al. [18] have devised a method to measure the accuracy of the depth obtained using a stereo camera based on stereoscopic vision. The accuracy of depth, as obtained by a stereo algorithm, is affected by several factors such as errors in calibration and matching of the corresponding point pairs. The positioning and mounting of the stereo cameras should be such that there is no relative motion between them and even if there is, it should be traceable [19]. They propose that the accuracy of a stereo camera must be evaluated in terms of both the absolute depth measurement accuracy and the depth resolution.

Kwaśnicka and Wawrzyniak [20] propose an algorithm for license plate localization and recognition which works under various environmental conditions and is not

restricted to a certain type of plate. In this algorithm, license plate localization is achieved by determining the presence of what is called the *license plate signature*, which is a characteristic sequence of minimums and maximums in the brightness which is observed when a horizontal scan line, passing through the entire image, passes through the license plate region. Once all the possible *signatures* are found, further grouping and eliminating algorithms are applied in order to narrow the list of possible candidates, until the plate is successfully located.

Bulugu [21] proposes a method which takes advantage of the standard colors of the Tanzanian license plates by simply trying to locate regions in the image with black writing on a white or yellow background. Shapiro, Dimov et al. [22] propose an adaptive method for license plate localization purposes. Their method involves edge detection, rank filtering of the aforementioned edges, plate candidate segmentation and plate candidate verification, all done in order to localize the plate region in the image.

Mahini et al. [23] propose another robust approach to solving the plate localization problem. Their approach is based on the different features of the license plates in order to deal with the complex situations, like varying illumination, shadows, scale, rotation and weather conditions, that are faced in the real world. The candidates are determined based on edge detection, morphological operations, and color analysis of the images. The incorrect candidates are then eliminated based on the image features and the correct license plate features are obtained. The algorithm used in this thesis is similar to the one proposed by them and makes use of the edge detection and morphological operations as suggested by them.

Chapter 3

Technology Used

OpenCV [24] is used in order to implement and test the algorithm that is being proposed. OpenCV is a computer vision library of programming functions developed by Intel mainly for real-time computer vision. The primary interface of OpenCV is in C++ with additional interfaces in Matlab, Java and Python. Some of the areas in which OpenCV [25] is widely used are:

1. 2D and 3D feature toolkits,
2. Facial recognition,
3. Gesture recognition,
4. Human-computer interaction (HCI),
5. Object identification,
6. Segmentation and recognition,
7. Stereopsis stereo vision: depth perception from two cameras,
8. Structure from motion (SFM),
9. Motion tracking, and
10. Kalman filtering

The complete documentation and support for OpenCV can be found online [26]. The features of OpenCV have been summarized and depicted by Hewitt [27]. OpenCV was first released in 1999. The initial requirement included the use of Intel's Image Processing Library which was later removed, OpenCV can now be used as a standalone

library. OpenCV has support for many platforms like Windows, Linux and recently Mac OS X. Its interfaces are platform independent. A brief summary of the major functionalities of OpenCV are given below.

1. General computer vision algorithms

Includes implementation of many standard computer vision algorithms without having to code them yourself. These include edge, line, and corner detection, ellipse fitting, image pyramids for multiscale processing, template matching, Fourier transform and more.

2. High-level computer vision modules

Includes several high-level capabilities like facial recognition, and tracking, optical flow (using camera motion to determine 3D structure), camera calibration, and stereo vision.

3. Machine learning modules

Computer vision applications often require machine learning or other AI methods. Some of these are available in OpenCV's Machine Learning package.

4. Image sampling and view transformations

OpenCV includes interfaces to process a group of pixels as a subunit like extracting image subregions, random sampling, resizing, warping, rotating, and applying perspective effects.

5. Math routines

Includes support for many algorithms from linear algebra.

6. Data structures and algorithms

Includes support to manipulate large data sets like lists, collections and trees.

Because of the plethora of computer vision related functions readily available in the OpenCV libraries and also the ease of implementation, OpenCV is an ideal tool for all computer vision related applications.

Chapter 4

Stereo Vision Based Depth Estimation

Depth estimation is the most important aspect of this thesis. It is through depth estimation, along with the license plate localization algorithm explained in Chapter 5, that the distance to the vehicle in front is estimated. In this chapter we look at the various factors that are involved in the actual depth estimation process that is being proposed. The process of depth estimation is done over several stages as elaborated in the following sections of this chapter.

4.1 Camera Calibration

Camera calibration is the process of estimating the spatial relationship between the two cameras: that is, determining the positions of the cameras that make up the stereo set up with respect to one another. This is usually done by using a calibration pattern, such as a checkerboard pattern where the size of the checks is known, and taking pictures of this pattern in different orientations and distances from the cameras. A simple contrast based algorithm can be used to detect the black-white intersections on the checkerboard. Using the disparities between these intersection points in both images the rotation and translation matrices between the two cameras can be determined and hence the coordinate systems of each of the cameras can be transformed into a common coordinate system. The algorithm used here is based on the OpenCV [24] implementation based on [28] and [29].

The functions in OpenCV use the pinhole camera model. A pinhole camera model, as seen in Figure 4.1, describes the mathematical relationship between the coordinates of any 3D point in the world and the 2D coordinates of the image plane onto which this 3D point is projected. The name *pinhole* camera is due to the fact that the aperture

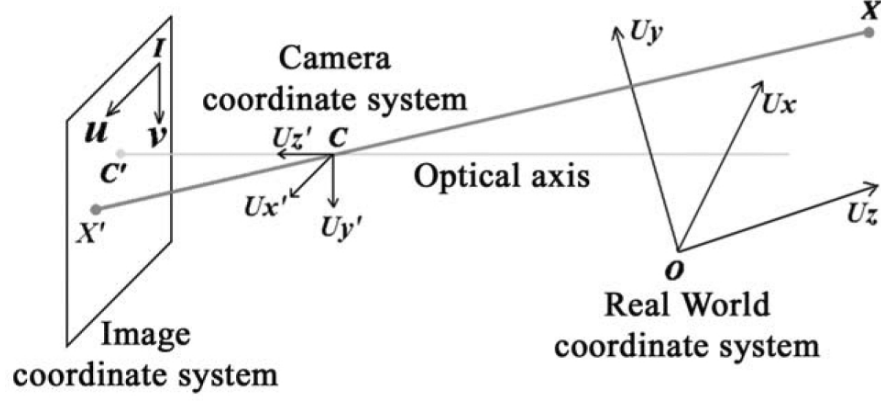


Figure 4.1: Pinhole camera model



Figure 4.2: Calibration pattern as seen from both the cameras

of the camera is assumed to be a point and no lenses are used to focus light [30]. The 3D point is projected onto the 2D image plane using a perspective transform:

$$s \times m' = M_{int} \times [R|t] \times M' \quad (4.1)$$

$$s \times \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \times \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad (4.2)$$

where, (X, Y, Z) are the coordinates of a 3D point in the world coordinate space, (u, v) are the coordinates of the projection point in pixels, A is the camera matrix, or a matrix of intrinsic parameters, (c_x, c_y) is a principal point that is usually at the image center, (f_x, f_y) are the focal lengths expressed in pixel units. The intrinsic matrix parameters,



Figure 4.3: Camera calibration

as specified by M_{int} , do not depend on the scene viewed. So once it has been estimated, it can be reused as and when necessary as long as the focal length remains the same and the stereo setup is not disturbed. The $[R|t]$ matrix describes the rotation and translation of the camera around a static scene, or vice versa. That is, $[R|t]$ is used in order to translate a point (X, Y, Z) into a coordinate system that is with respect to the camera.

In the real world, camera lenses usually have some form of distortion, mostly radial and in some cases tangential. Distortion parameters are incorporated into the above model in order to account for this. As explained in the OpenCV documentation [24], the transformation given by Equation 4.2, taking into account the distortion parameters, as given in the OpenCV documentation [24], results in

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \times \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + t, \quad (4.3)$$

which implies

$$x' = x/z \quad (4.4a)$$

$$y' = y/z \quad (4.4b)$$

$$x'' = x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2 (r^2 + 2x'^2) \quad (4.4c)$$

$$y'' = y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 y' y' + p_2 (r^2 + 2y'^2), \quad (4.4d)$$

where

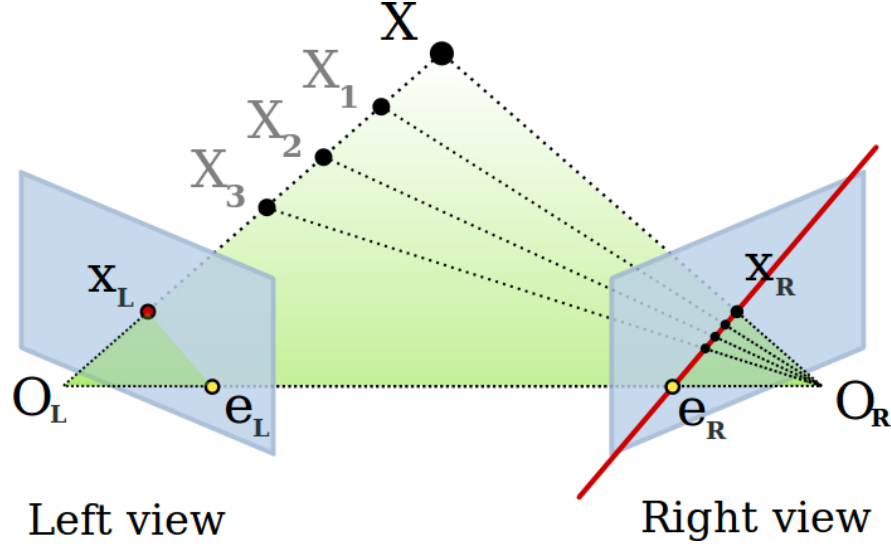


Figure 4.4: Epipolar Geometry [1]

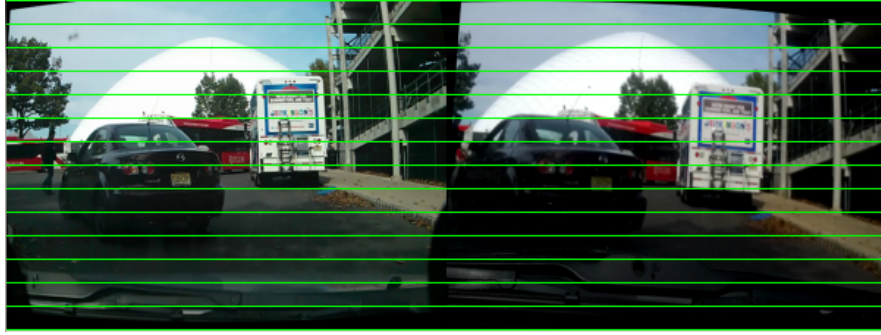


Figure 4.5: Rectified Images

- $r^2 = x'^2 + y'^2$,
- $u = f_x * x'' + c_x$,
- $v = f_y * y'' + c_y$,
- k_1, k_2, \dots, k_6 are the radial distortion components, and
- p_1 and p_2 are the tangential distortion components.

4.2 Epipolar Geometry and Rectification

By assuming the *pinhole* camera model (as described in Section 4.1) for both the cameras, we can arrive at a number of geometric relations between the 3D point and the

2D images from the two cameras. The geometry that deals with this is called *epipolar geometry* and is extensively used in stereo vision. The epipolar geometry can also be described as the intrinsic projective geometry between the two views. It is independent of scene structure, and only depends on the cameras' internal parameters and relative pose. Figure 4.4 depicts two cameras, with camera centers O_L and O_R , *looking* at a real world 3D point \mathbf{X} . The points \mathbf{x}_L and \mathbf{x}_R represent the projections of the point \mathbf{X} onto the image planes of the left and the right cameras respectively. The points \mathbf{X} , \mathbf{x}_L and \mathbf{x}_R all lie on the same plane. This plane is called the *epipolar plane*. This geometry is usually motivated by considering the search for corresponding points in stereo matching. The process of determining the corresponding point pairs in both the images can computationally be very intensive as, for each point in any one of the images, called the source image, the algorithm has to search for a similar point in the other. This point in the second image, called the search image, can be located anywhere, hence a two dimensional search has to be carried out. Suppose we know the point \mathbf{x}_L , and want to figure out how its corresponding point \mathbf{x}_R is constrained. We know that the ray corresponding to \mathbf{x}_R lies in the epipolar plane, hence the point \mathbf{x}_R lies on the line of intersection of the epipolar plane and the right image plane. This line of intersection is known as the epipolar line. Given the camera internal and external matrices and the point, we can estimate the epipolar line in the right image, thereby constraining the search space to this 1D line.

The process can be further simplified by using a process called rectification. Image rectification is the process by which two or more images are projected onto a common image plane. This process rectifies the distortion by transforming the images into a standard coordinate system in which the images are rotated and translated such that the image planes of both the cameras are parallel to each other, thereby resulting in the epipolar lines being parallel to the baseline connecting the two cameras. This is done by moving the epipoles to infinity. As a result of this, the image planes of both the cameras are made parallel to each other. A set of rectified images of the calibration pattern is as shown in Figure 4.5.

4.3 Sparse Stereo Correspondence

Stereo correspondence is the problem of discovering the closest possible matches between two images captured simultaneously from two spatially separated cameras. There are two main ways of doing this:

1. Dense stereo matching, and
2. Sparse stereo matching.

Dense stereo matching involves trying to match as many pixels as possible. This is useful if detailed information about the scene is required or if the entire scene is being reconstructed. This method is computationally very expensive. Some slightly efficient algorithms such as *Block matching*, *Semi Global Block Matching* etc. have been incorporated into OpenCV [31]. Another way of working around this would be to have dedicated hardware which runs a brute-force algorithm. However, for the requirements of this thesis, a *Sparse stereo matching* algorithm suffices since the complete 3D reconstruction of the scene is not the objective. As we are just interested in the vehicle that is directly in front, the sparse stereo matching algorithm is run on the Region of Interest (ROI) as returned by the license plate localization algorithm, explained in Chapter 5, thereby improving the efficiency and also automatically localizing the location of the vehicle in the image scene.

Sparse stereo correspondence involves searching for, and locating feature points of interest, that is highly distinctive points, in one of the images. Let us call this image the search image. The goal is to find the corresponding points in the other image. SURF (Speeded Up Robust Features) method is being used here to find the point matches between the left and the right image as explained in Section 4.4. It is a robust local feature detector. It is similar to SIFT (Scale Invariant Feature Transform) extractor, but is faster and more robust. SURF is an algorithm that extracts some unique keypoints and descriptors from the image. The detection and matching process can be done at high speeds, making it a viable option for real time applications. Object detection using SURF is scale and rotation invariant, and as a result the object can

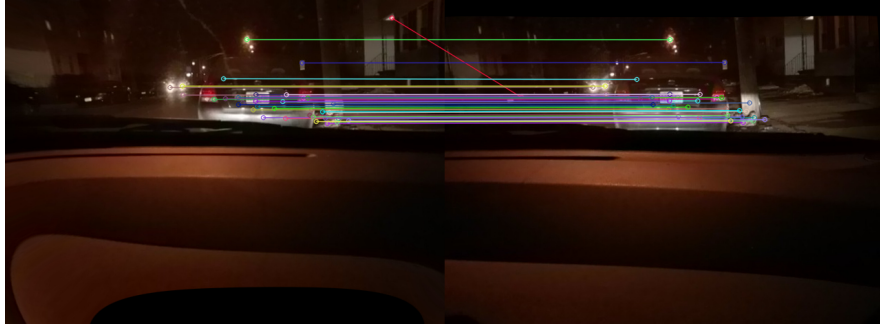


Figure 4.6: SURF Matching

be detected in any orientation. The detector used is based on the Hessian matrix, and as a result the computation time and accuracy is improved. The determinant of the Hessian matrix is used for selecting the location and the scale. Further details of the SURF can be seen in [13]. *SURF*, along with the *brute force* matching algorithm given by the *BFMatcher* class in OpenCV [24] is used to do so. *RANSAC* is then used for the effective removal of the outliers, hence obtaining a set of corresponding matches with high confidence. The output of the *SURF* matching is shown in Figure 4.6.

4.4 Depth Estimation

Depth estimation is the most important step in this project, wherein the actual distance to the vehicle in front is determined. In order to find the depth, we first need to determine the disparity of the corresponding points. The disparity is the apparent shift in the locations of the interest points along the x-axis in a rectified pair of stereo images. The disparity can be seen in Figure 4.7, which is obtained by superimposing the image from one of the cameras on top of the other. For the purposes of this thesis, a sparse disparity map would suffice as we are just looking to find the depth of the car and do not require the entire depth field. Hence we use the *SURF* feature detector along with the *brute force* feature matcher in order to find the corresponding points in both the rectified images. Once we get the corresponding features from both the rectified images, we find the difference between their x-coordinates thereby obtaining disparity at that point. In order to make the feature detection more robust, we use *RANSAC* (*RANdom SAMpling and Consensus*) in order to filter out the outliers, thereby obtaining better



Figure 4.7: Illustration of disparity

point correspondences.

Once the disparities are determined, the corresponding distances at each set of obtained point correspondences is calculated using the formula,

$$distance = \mathbf{f} \cdot \mathbf{B} / \mathbf{d}, \quad (4.5)$$

where \mathbf{f} is the focal length in pixels, \mathbf{B} is the baseline length, that is the distance between the two camera centers and \mathbf{d} is the disparity at the corresponding set of points being considered.

Summarizing the above points, the individual frames from each of the video streams are extracted and then the frames are rectified in order to remove tangential and radial distortions that might exist due to imperfections of the lens. The camera intrinsic and extrinsic matrices are used in order to perform rectification. Then a SURF detector is used along with the brute force matcher which has been implemented in OpenCV [24] in order to find the feature points and their corresponding point matches. From the point matches thus obtained, the disparity is calculated and thus the distance is estimated from the disparity.

Chapter 5

License Plate Localization

In this thesis, we propose a novel license plate localization algorithm, which is used in order to localize the region of the image in which the car is located. The algorithm is designed so as to take advantage of the various characteristics of the license plate such as its rectangular shape, the fact that it usually has dark writing on a lighter background, the fact that in the video feed it is usually seen near the center of the frame, and others. A feature-based license plate localization algorithm that is robust and effective in different image capturing conditions, similar to the ones used by Mahini et al. [23] and Chhabada et al. [32] is implemented. The algorithm is robust and can overcome various undesired conditions such as out of focus images, different illumination conditions, and variations in the orientations of the license plate with respect to the cameras. The various steps involved in plate localization are as described in the following sections.

5.1 Preprocessing

Before carrying out the license plate localization, the incoming video feed has to be preprocessed, in order to improve the quality of the image feed. Firstly, the frames are grabbed from the video feed and then converted into grayscale. This is followed by *histogram equalization*, thereby enhancing the contrast of the image.

5.2 Edge detection using the Sobel Operator

The Sobel operator is used in edge detection algorithms [33]. It is a discrete differentiation operator that computes an approximation of the gradient of the image intensity function. The Sobel operator involves two kernels, one for vertical changes and one for



Figure 5.1: Sobel operators applied to the grayscale input image

horizontal changes. It is given by the following formula:

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} * A \quad (5.1)$$

$$G_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} * A \quad (5.2)$$

where G_x and G_y are two images that contain the horizontal and vertical derivative approximations respectively, that is the horizontal and vertical edge maps. Once the individual components are obtained, at each point in the image we calculate the approximation of the gradient at that point using the formula:

$$G = \sqrt{G_x^2 + G_y^2}. \quad (5.3)$$

Using the Sobel operator, we obtain the edges present in the input image, and the obtained edge map is stored for further processing. The Sobel edge map obtained is shown in Figure 5.1.



Figure 5.2: Morphological close operator applied to the original input image

5.3 Morphological Close Operation

The morphological close operation is a process useful in removing small holes (dark regions) from an image. It is obtained by dilating and then eroding an image file. The effect of this operator is to preserve background regions that have similar a similar shape to the structuring element, or that can completely contain the structuring element, while eliminating all other regions of the background pixels. Here a structuring element of size $[1 \times 2]$ since the aspect ratio of a standard license plate in the USA was found to be 1:2. The closing operator is applied to a grayscale version of the input and then the resulting image is blurred using a median blur filter of size $[7 \times 7]$.

5.4 Thresholding

The output of the closing operation is then subject to a thresholding, yielding a binary image consisting of just the candidate regions. After having run the algorithm over several images under various conditions, a threshold of 150 was found to be ideal. After the thresholding, salt-and-pepper type noise is removed by running a median filter of size $[9 \times 9]$ over the image. The median filtering was done using the `medianBlur`

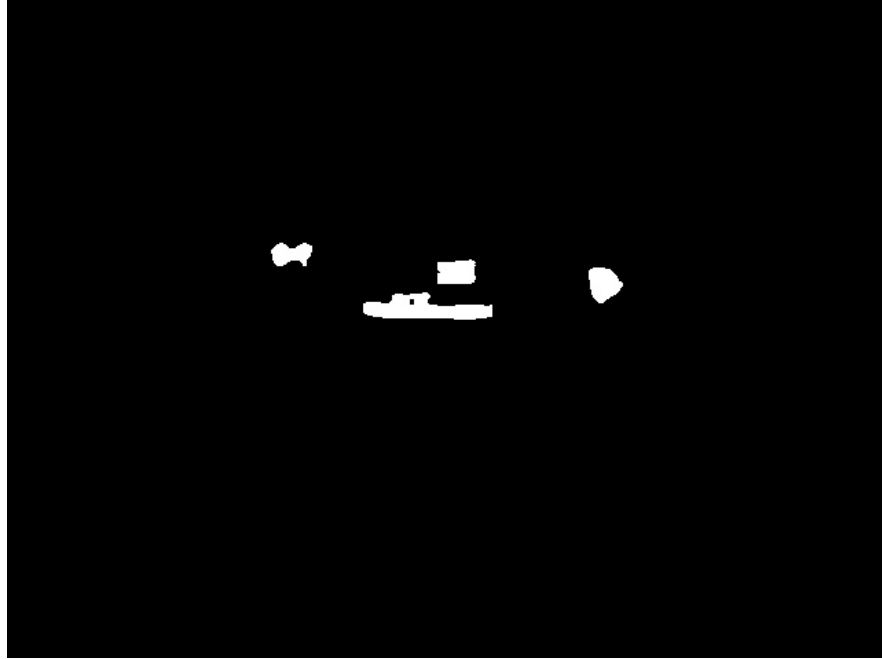


Figure 5.3: First set of candidates extracted

function from OpenCV. The thresholded version of the output of the close operator after blurring is shown in Figure 5.2. This was done using the `threshold` [24] function present in OpenCV.

5.5 Extracting the Possible Candidate Regions

Once the thresholding has been performed, the edge map obtained using the Sobel operator is multiplied by the thresholded image obtained in the previous step. The possible candidates obtained are as shown in Figure 5.3. This was done using the `Mat::mul` operator [24] present in OpenCV.

5.6 Contour detection

The next step was to separate the various candidate regions in the thresholded image. This was done using the `findContours` [24] function in OpenCV. Once all the contours were obtained, the next step was to approximate the contours to polygons, using the function `approxPolyDP`, and only such polygons were selected which have at least four edges. Ideally, only those polygons need to be selected which have just four edges, but it



Figure 5.4: Possible candidate regions

was observed that under different lighting conditions and when there were shadows on the license plate, sometimes the actual license plate region was approximated to polygons consisting of more than four edges. Hence in order to take this into consideration, all polygons with more than four edges were taken into consideration.

Once the polygons were obtained, minimum bounding boxes were drawn around each of the polygons (bounding boxes are rectangles of minimum area required to close in the contours), resulting in a refined set of candidates from the previous set.

5.7 Candidate elimination

The set of candidates obtained from the previous steps were subjected to progressive elimination based on the following criteria:

- The plate region must neither be too small nor too big. For this the sizes of the plate in the images, for the smallest and the largest distances that could be detected, were used as constraints.
- The width of the detected region should be approximately twice that of its



Figure 5.5: The localized plate region



Figure 5.6: Correct Localization



Figure 5.7: Incorrect Localization

height [34].

- The average intensity of the plate region must be light enough.
- The intensity of a line passing through the middle of each of the bounding boxes, must have the highest number of fluctuations, as the license plate has dark characters on a light background.
- The region of the license plate is usually found to be somewhere close to the center of the image and in the lower half. Thus we neglect all candidates that are near the sides and also near the top of the image.

Based on the above criteria, the most eligible candidate area to contain the license plate was selected. The final selected candidate is shown in Figure 5.5.

Once this was done, the Region Of Interest (ROI), the region containing the license plate, was passed to the SURF function. As a result of this, only the distance estimate to the vehicle in front is made.

In summary, the localization algorithm looks for the edges in the image and determines the possible regions of the image where dark characters on a light background is

present. Once this is done, the possible candidates are determined and based on the criteria stated above, one of those candidates is selected to estimate the location of the license plate in the image. As can be seen in Figure 5.6 the license plate was successfully localized using the above stated algorithm. The accuracy of the algorithm varied slightly depending on the environmental conditions such as different illumination levels, angles at which the license plate is viewed by the stereo setup and other factors such as the contrast between the color of the vehicle and that of the license plate. It was found that the accuracy of the algorithm was better for cases where the contrast was high. For example, localization was easier for dark colored cars as opposed to those of lighter colors. Taking into consideration all the factors mentioned above, the algorithm was found to work with an average accuracy of 85.71%.

Chapter 6

Implementation

This chapter deals with the various aspects of implementation, such as the experimental setup, the software functions used for the implementation and the data collection methods incorporated in order to test and verify the proper working of the algorithm. The following sections provide an in-depth view into these aspects.

6.1 Experimental Setup and Equipment Used

Two Google Nexus 4 Android phones were used in the stereo rig. Two phones of the same kind were chosen so that the focal lengths of the cameras were the same, as this makes the distance estimation process simpler. Another reason for using the same kind of phones was to make sure that the video capture was done at the same frames per second in both the phones. The stereo rig was mounted inside a car using phone mounts looking out of the front windscreen. The initial plan was to have the stereo cameras looking out of the rear window, but some states in the USA do not require license plates on the front. Hence the decision was made to place it in the front looking out of the windscreen. It was important to prevent any relative motion between the two phones, as this would ruin the calibration results and lead to erroneous values of estimated distances. This was an especially difficult task since meticulous care had to be taken in order to prevent the vibrations experienced by the moving car from disturbing the calibrated stereo rig. Two phone mounts placed at separation of 10 *cm* was used for this purpose. The distance of 10 *cm* was arrived at by setting an effective range of 1 *m* to 10 *m* and then calculating the baseline length required. This calculation is as elaborated below:

$$\mathbf{d} = \mathbf{b} * \mathbf{f}/\mathbf{z} \tag{6.1}$$



Figure 6.1: The mounts used for the cameras

where,

1. \mathbf{d} is the disparity; the minimum disparity is chosen as 6 pixels and the maximum disparity is chosen as 64 pixels,
2. \mathbf{b} is the baseline length (this has to be calculated),
3. \mathbf{f} is the focal length, which is obtained from the intrinsic camera matrix and was found to be 630 pixels
4. \mathbf{z} is the distance; the range of distance is assumed to be 1 m to 10 m .

Using the above values in the Equation 6.1, it was determined that the baseline length should be 10 cm for the above given parameters. The actual setup of the stereo rig is shown in Figure 6.1.

6.2 Implementation of the Algorithm

The implementation of the algorithm is depicted by the flow chart shown in Figure 6.2. Several functions present in the OpenCV [24] library for C++ were used for the imple-

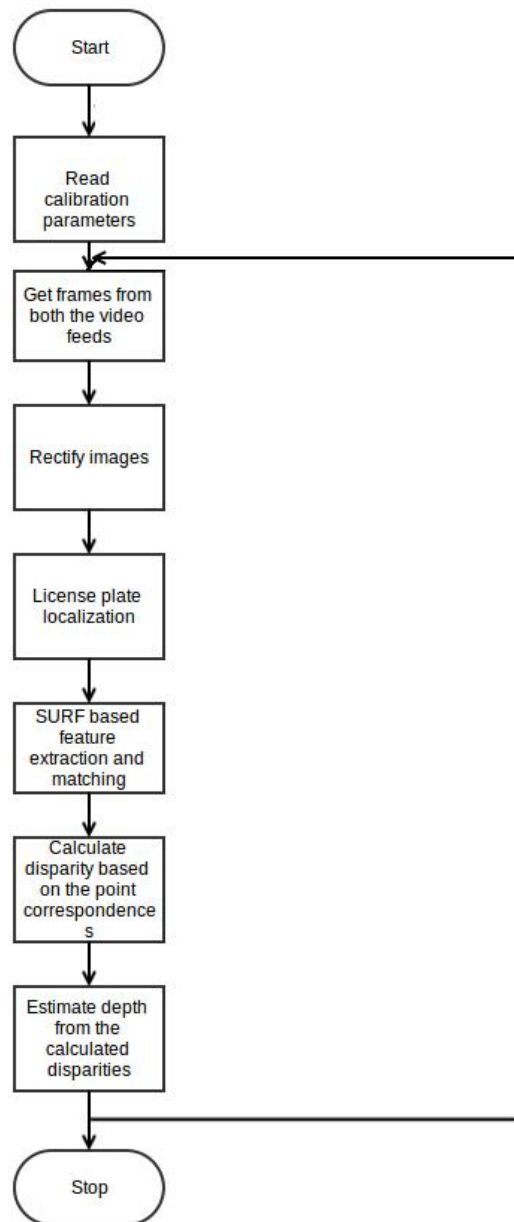


Figure 6.2: Flowchart of the algorithm

mentation of both the depth estimation and the license plate localization stages of this thesis. The functions used for the depth estimation were:

1. The `stereo calibration` tutorial code provided by the OpenCV tutorials [26] and the open source code provided by Martin Peris [35] were used in order to obtain the intrinsic and extrinsic camera calibration parameters.
2. The `filestorage` class, which encapsulates all the information necessary for writing or reading data to/from an XML/YAML file [24], is used to save the camera calibration parameters into, and then read from, an XML file.
3. The functions `initUndistortRectifyMap()` and `remap()` undistort the images and provide the rectification mapping that needs to be applied to the images as explained in Section 4.2. Undistorting removes the effects of radial and tangential distortion that may be present.
4. The `SurfFeatureDetector` class is used in order to detect the feature points in both the image. This is done in order to find the sparse correspondence matches between the images from both the cameras.
5. The `BFMatcher` class, which uses the brute force matching algorithm in order to find the sparse correspondence matches mentioned above.

For the license plate localization algorithm, the OpenCV functions used were:

1. The `Sobel()` function which is used to find the vertical and horizontal derivative approximations, thereby achieving the edge detection.
2. Various morphological transformation operations such as closing, dilation, erosion etc. which were explained in more detail in Chapter 5.
3. The `Canny()` function, in order to perform canny edge detection, the output of which serves as an input to the `findContour` function. The `findContour` function finds the contours which may be possible candidates for the license plate region in the image.

4. The `minRect()` function was used in order to draw minimum bounding rectangles around the detected candidates.

6.3 Data collection

The data collection was carried out under two different scenarios, with the cameras mounted in a Honda and a Toyota during these two phases of data collection respectively. A black colored Ford, silver colored Toyota, and a dark blue Mazda was used for testing.

1. The first scenario involved driving the car towards and away from a parked car, with markers placed on the road at regular intervals. The first set of markers were placed at 1.6764 m or 5.5 feet from the parked car, the next set at 2.4384 m or 8 feet, then 3.048 m or 10 feet and so on. The car with the cameras was made to back up away from the parked car, stopping at each of these markers and images captured from both the cameras simultaneously. The whole process was repeated five times, so as to get an estimate of the confidence levels of the distance estimated, and also calculate the error in the estimation. This set of experiments also gave an indication of the maximum and minimum range of the algorithm. Cars of different colors such as black, dark blue and silver were used for data collection. The tests were run during both day, and night. This was done to test the robustness of the algorithm with varying illumination levels.
2. The second scenario involved situations where both the cars are in motion. This included instances when the car in front is not directly in front, which can happen on curved roads or while making turns. This was done in order to determine if the algorithm works as efficiently for moving vehicles as it does for stationary ones. This set of experiments was conducted with the different conditions which involved dark-colored and light-colored cars and also varying levels of illumination experienced at different times of the day. This was done to test the robustness of the proposed algorithm. Apart from that several other conditions such as one car following the other in a straight line, along turns, when the car rapidly accelerates

from the other car or when it rapidly decelerates, were all taken into consideration during the testing process. The tests were conducted in a parking lot, where in the cars were driven around one behind the other. The path taken by the cars included two long straight lines, along with two turns.

The various tests proposed above were conducted, the results obtained were recorded and analyzed as explained in detail in the next chapter.

Chapter 7

Results

In this chapter we record the distance estimates obtained from the two data collection scenarios described in the previous chapter. The results obtained are recorded and then analyzed in a detailed manner as explained in this chapter.

In the first stage mentioned in the Chapter 6 involving one stationary car, data was collected in order to determine the error in the estimated distances. Markers were placed at various points along the road in between the two cars. The mounts were used to fix the cameras in one of the cars, thereby forming the stereo camera. This was done such that the baseline distance, the distance between the two camera centers, was 10 *cm* as was calculated in Chapter 6. The first set of markers were placed at 1.6764 m or 5.5 feet from the parked car, the next set at 2.4384 m or 8 feet, then 3.048 m or 10 feet and so on. The images from the stereo rig are fed as input to the algorithm. The algorithm, using the predetermined camera calibration parameters, rectifies the images and then applies a sparse descriptor matching algorithm to the images. SURF is used to find the descriptors in the source image and then a brute force matcher is used in order to find the corresponding points in the other image. The ROI for the SURF algorithm is determined by the license plate localization algorithm as explained in Chapter 5. The disparity at these intervals was calculated using the formula given by

$$\mathbf{d} = P_{Lx} - P_{Rx}, \quad (7.1)$$

where, \mathbf{d} is the disparity that is given by the difference of P_{Lx} and P_{Rx} , which are the X-coordinates of the interest point in the left image and its corresponding point in the right image. The disparity thus obtained from Equation 7.1 is then used in the depth calculation formula as given by Equation 4.5. The readings obtained are

tabulated and as shown in Table 7.1 along with the actual distances and the error percentages respectively. Figure 7.1 depicts the error calculated above with respect to the distances. Here the actual distance estimation is done post hoc, hence the whole experiment process is divided into two stages, the *data collection* stage and the *distance estimation* stage.

7.1 Data Collection

The steps involved in the data collection stage are as follows:

1. The first step was to calibrate the cameras and save the calibration parameters thus obtained.
2. Then the car with the stereo rig mounted inside, was positioned at 5.5 feet behind the parked car.
3. An Android application was used in order to synchronize the picture taking process. The images are taken at the exact same instance.
4. The car is then moved to the next marker and step 3 is repeated again.
5. This process is repeated several times at each of the markers.

7.2 Distance estimation

Having collected the data, we move on to the second stage of the experimentation, namely the distance estimation stage. The steps involved in this stage are as follows:

1. The images are fed as input to the algorithm along with the camera calibration parameters obtained during step 1 of the previous stage.
2. The average of the distances at all the feature points found inside the ROI, obtained by the license plate localization, is calculated.
3. The above step is repeated for all the sets of images that were taken at each of the markers. The average value of the distance estimated at each of them is calculated and tabulated.

Table 7.1: The average errors for different distances

Actual Distance (in m)	Estimated Distance (in m)	Error
1.6764	1.599445	0.0459
2.4384	2.40532	0.0135
3.048	2.808225	0.078666
4.2672	3.52961	0.17514
4.8768	3.93870	0.19236
5.4864	4.34157	0.208666
6.096	4.6659	0.2345987

Table 7.2: Confidence intervals for the various distances estimated

Actual Distance (in m)	1.6764	2.4384	3.048	4.2672	4.8768
Trial 1	1.62988	2.52902	2.80822	3.52961	3.93870
Trial 2	1.56901	2.33658	2.82546	3.84975	4.88456
Trial 3	1.42315	2.56821	2.97146	3.68974	4.99854
Trial 4	1.53014	2.48929	3.12458	4.02548	4.25698
Trial 5	1.69058	2.28162	3.25648	4.15897	4.22580
Std. Dev	0.086902	0.101386	0.147553	0.212764	0.506291
Mean	1.56855	2.44094	2.99724	3.85071	4.46091
Confidence Interval	0.076172	0.088868	0.129335	0.186496	0.443784

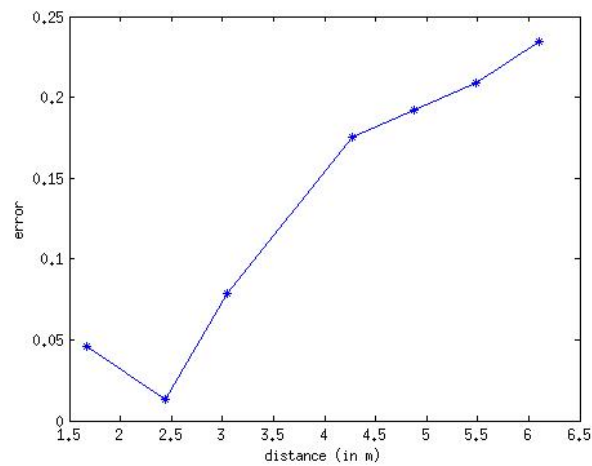


Figure 7.1: Error vs. Distance

Table 7.3: The estimated distances at various time instances

Time (s)	Distance (m)
70	2.3697
80	4.8890
90	3.3965
100	5.1664
110	5.2589
120	5.4007
130	3.8795
140	7.3131
150	5.0511
160	1.6764

For the second method, the *calculation* stage of the experiment remains the same, but there is a slight difference in the *data collection* stage. For the data collection stage, both the cars were driven around a parking lot, with one following the other and the cameras. This time they were used to record a continuous video stream of the car in front. The frames from the video captured were fed as input to the algorithm along with the calibration parameters. In order to reduce the computational load, we process every 100th frame from each of the cameras. With the 60 *fps* video recording capabilities of the Nexus 4 phones used, this works out to one distance estimate every 1.667 *s*. As before, the license plate ROI is determined, the sparse point correspondences found using SURF within the ROI, and the depth estimated by using Equation 4.5, where the disparity is determined by using Equation 7.1. The values for the distance between the cars at different time instances are shown in the Table 7.3. Here, the frames are sampled at 10 second intervals and the distance estimated at each of those instances. The first sample is made at 70 *s*, the next sample at 80 *s* and so on for the remaining length of the video. The distance estimated at each of these time instances is plotted against the time and is shown in Figure 7.2

7.3 Analysis

The distance estimates thus obtained as compared to the actual distances are as shown in Table 7.1. Up to 12 feet, the algorithm output has an error percentage of less than 10% and from 12 feet onwards, this error is seen to increase to about 17 % to 25 %.

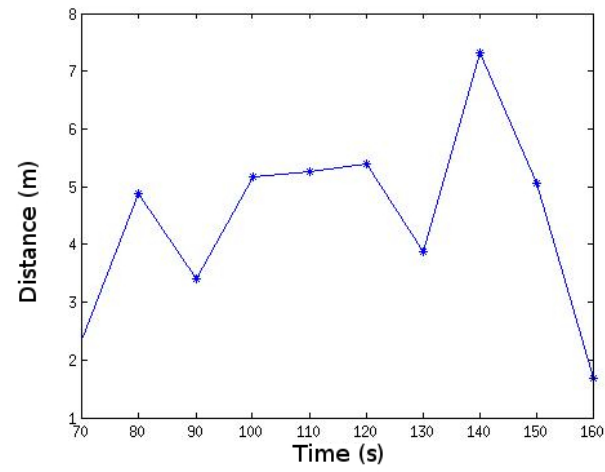


Figure 7.2: Distances estimated with both cars in motion



Figure 7.3: The estimated distances with both cars in motion



Figure 7.4: The estimated distances with one car stationary

This is because the range of effective distance calculation is influenced by the baseline distance. To obtain distances beyond the above given range, the cameras have to be placed further apart, but by doing this we sacrifice the distance calculation capabilities of the nearby objects since the field of views intersect at a greater distance from the cameras as the baseline length increases. The ideal baseline length which was calculated to be 10 *cm*, was used.

The first set of experiments was performed with one stationary car and then taking pictures from the stereo camera, which was mounted in the other car, at a known set of distances. This was done in order to get the ground truth values of the distances and to measure the accuracy of the algorithm. The estimated values, shown in the Table 7.1, show the most effective range to be up to 4.5 *m* or 12 *feet*. The error rates here are less than 10 %. Beyond this 12 *feet* range, the average error is seen to increase to about 25 %. In order to calculate the confidence interval of the distance estimates, we make use of five sets of data for the same distance from the vehicle in front. Initially we start off at the first marker, and then go all the way back to the last marker, stopping at each marker in order to record the data. Once this is done we repeat the process four more times, tabulating the distance estimates obtained during each of the trials. Once this is done, the mean and the standard deviation is calculated and then by setting a confidence level of 95%, we calculate the confidence interval of the estimates of each of the distances from the vehicle in front. The confidence intervals, for the distances estimated is shown in Table 7.2.

In the second set of experiments, with both the cars in motion, the same set of experiments as before were conducted and the results obtained were recorder. Variations in the distances at various instances of time were observed, owing to various conditions, such as turns, straight lanes, disturbances such as other vehicles, etc. The license plate localization algorithm tended to produce erroneous values when there were some disturbances in the frames, such as when other vehicles were present in the scene. This was taken care of by comparing the distance values with the previously obtained value, and if the current distance estimate showed a variation of greater than 3 *m* from the previous estimate, then the current estimate was ignored, and the next set

of frames were read and the distance estimated again. It should be kept in mind that the algorithm is most efficient up until a distance of around 4.5 *m* or 12 *feet* as was determined by the previous set of experiments. Beyond this the error percentage tends to increase significantly and was found to be around 25 % for distances of around 20 *feet*.

Chapter 8

Conclusion and Future Work

This thesis provides a way of measuring the distance to any vehicle that is in front of it by means of depth estimation using a calibrated stereo camera setup. This is then used as an effective data collection tool from volunteer participants, in order to create a following distance logger. The distance estimates obtained will serve as data for research in quantifying driver behavior. The effective range of the algorithm is limited by the horopter, which is dependent on the baseline distance of the stereo setup. By using a larger baseline length, the effective range can be increased. However, this is not always feasible because a larger baseline implies smaller overlap between the two image scenes, thus the stereo matching process is more difficult. Not only this, but the usage of a larger baseline also results in completely missing the estimation of depth of objects close to the cameras. The individual frames grabbed from the video feeds of both the cameras are first rectified and then SURF is used to find feature points in the frames from one of the cameras and then find the corresponding point matches in the other. RANSAC was used to filter out the outliers from the point matches thus obtained. Once the point correspondences are determined, the disparity at each of these points is determined and then from this the depth at each of these points is determined. Since all these points lie within the region containing the license plate of the vehicle in front, an average of all the distance estimates at the individual points is taken and is output as the distance to the vehicle in front. The stereo depth estimation process used results in a computationally less intense algorithm.

The ROI of the image which is fed to the SURF feature detector algorithm is determined by the license plate localization algorithm that was developed. This ensures that the location of the points needs to be determined in just a small portion of the

image as opposed to the entire image and hence reduces the computational load of the algorithm to a certain extent. The license plate localization proposed is a robust, feature-based algorithm, which relies on the fact that the license plate is rectangular and also that it usually consists of dark characters present on a lighter background. This makes this implementation almost universal, which implies that the algorithm is not restricted to any one particular kind of license plates. It was observed that the license plate localization algorithm works exceptionally well at night, when there is not much ambient light and when the headlights of the vehicle in which the cameras are mounted are focused on the license plate of the vehicle in front. This can be attributed to the fact that the reduced amount of ambient light makes other objects, that may interfere with the proper functioning of the algorithm, not visible and the license plate and the vehicle in front is the main object in focus of the headlights. Hence detecting it becomes less error-prone.

8.1 Future Work

In order to further improve the accuracy of the depth estimation process, one could use the accelerometer data already available from the Android phones and look to negate the effects of the minute vibrations that may occur and result in the displacement of the cameras with respect to each other. Another area where the accuracy might be improved is the license plate localization algorithm which can be made more robust to changing illumination levels, for example when there are shadows across the face of the license plate and for video feeds taken during different times of the day with varying levels of ambient lights. It was observed that the presence of shadows severely deteriorated the accuracy of the algorithm and as a result the ROI which localized the car in the image frame would not be detected or the detected region would be incorrect. Another area of development could be making it possible for the license plate localization algorithm to differentiate between other regions in the image that have dark characters on a light background, for example billboards, from the license plates. This is taken care of, to a certain extent, by confining our search for the license plate in the bottom two-thirds of the image and also by neglecting the candidates that occur closer to the left and right

edges of the image. Further work would include making this whole process real-time. This can be achieved by using the OpenCV library for Android and using one of the two phones used in the stereo setup to process of the image frames from the video feeds of both the cameras.

References

- [1] Wikipedia. Epipolar geometry — Wikipedia, The Free Encyclopedia, 2013.
- [2] A. Alessandrini, A. Cattivera, F. Filippi, and F. Ortenzi. Driving style influence on car CO_2 emissions. In *In proceeding of: 20th International Emission Inventory Conference - "Emission Inventories - Meeting the Challenges Posed by Emerging Global, National, and Regional and Local Air Quality Issues"*, August 2012.
- [3] J. Deffenbacher, E. Getting, and R. Lynch. Development of a driving anger scale. In *Psychological Reports: Volume 74*, pages 83–91, 1994.
- [4] T. Lajunen and H. Summala. Driver experience, personality, and skill and safety motive dimensions in drivers self-assessments. In *Personality and Individual Differences, Vol. 19, pp*, 1995.
- [5] N. Li, J.J. Jain, and C. Busso. Modeling of driver behavior in real world scenarios using multiple noninvasive sensors. *Multimedia, IEEE Transactions on*, 15(5):1213–1225, Aug 2013.
- [6] Ray A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 5(2):122–139, 1983.
- [7] E. Tola, C. Strecha, and P. Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Mach. Vis. Appl.*, 23(5):903–920, 2012.
- [8] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV'01)*, SMBV '01, pages 131–, Washington, DC, USA, 2001. IEEE Computer Society.
- [9] M. Bansal, A. Jain, T. Camus, and A. Das. Towards a practical stereo vision sensor. In *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 63–63, June 2005.
- [10] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele. Stereo vision-based vehicle detection. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 39–44, 2000.
- [11] K. Y. Lee, J. W. Lee, and M. R. Cho. Detection of road obstacles using dynamic programming for remapped stereo images to a top-view. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 765–770, June 2005.
- [12] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

- [13] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.
- [14] C. Harris and M. Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.
- [15] T. Kadir and M. Brady. Saliency, scale and image description. *Int. J. Comput. Vision*, 45(2):83–105, November 2001.
- [16] F. Jurie and C. Schmid. Scale-invariant shape features for recognition of object categories. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–90–II–96 Vol.2, June 2004.
- [17] L. Juan and O. Gwon. A Comparison of SIFT, PCA-SIFT and SURF. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009.
- [18] M. Kyto, M. Nuutinen, and P. Oittinen. Method for measuring stereo camera depth accuracy based on stereoscopic vision. volume 7864, pages 78640I–78640I–9, 2011.
- [19] W. Zhao and N. Nandhakumar. Effects of camera alignment errors on stereoscopic depth estimates. *Pattern Recognition*, 29(12):2115 – 2126, 1996.
- [20] H. Kwaśnicka and B. Wawrzyniak. License plate localization and recognition in camera pictures. In *AI-METH 2002*, 2002.
- [21] I. Bulugu. Algorithm for license plate localization and recognition for tanzania car plate numbers. *International Journal of Science and Research (IJSR), India*, 2013.
- [22] V. Shapiro, D. Dimov, S. Bonchev, V. Velichkov, and G. Gluhchev. Adaptive license plate image extraction. In *Proceedings of the 5th International Conference on Computer Systems and Technologies, CompSysTech '04*, pages 1–7, New York, NY, USA, 2004. ACM.
- [23] H. Mahini, S. Kasaei, F. Dorri, and F. Dorri. An efficient features - based license plate localization method. In *Proceedings of the 18th International Conference on Pattern Recognition - Volume 02*, ICPR '06, pages 841–844, Washington, DC, USA, 2006. IEEE Computer Society.
- [24] A. Bradski. *Learning OpenCV, [Computer Vision with OpenCV Library ; software that sees]*. O'Reilly Media, 1. ed. edition, 2008. Gary Bradski and Adrian Kaehler.
- [25] Wikipedia. OpenCV — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 19-March-2014].
- [26] G. Bradski. OpenCV documentation. *Dr. Dobb's Journal of Software Tools*, 2000.
- [27] R. Hewitt. Seeing with OpenCV, January 2007.
- [28] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

- [29] J. Y. Bouguet. Camera calibration toolbox for Matlab, 2008.
- [30] Wikipedia. Pinhole camera model — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 26-February-2014].
- [31] OpenCV Documentation. Camera calibration and 3D reconstruction.
- [32] S. Chhabada, R. Singh, and A. Negi. Heuristics for license plate detection and extraction. *World Journal of Science and Technology*, 1(12), 2012.
- [33] Wikipedia. Sobel operator — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18-March-2014].
- [34] Wikipedia. Vehicle registration plate — Wikipedia, The Free Encyclopedia, 2013.
- [35] M. Peris. OpenCV: Stereo camera calibration, January 2011.