A Parking Assistant System

by

Duangmanee Putthividhya

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2002

Author .....
Department of Electrical Engineering and Computer Science
June 26, 2002

Certified by .....
Dr. Ichiro Masaki
Principal Research Associate
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

A Parking Assistant System

by

Duangmanee Putthividhya

Submitted to the Department of Electrical Engineering and Computer Science
on June 26, 2002, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering

Abstract

Parking a car can be very cumbersome and difficult in many circumstances, especially when the rear view of the car is blocked. This thesis designs a prototype for an efficient parking assistant system that aims to facilitate the parking process by providing the drivers with distance information of obstacles in the scene. A video camera mounted on the rear window of a car is used to supply a sequence of input images to the system which then generates a 3-D depth map of the environment. The algorithm proposed in this thesis to reconstruct 3-D distance information from an image sequence involves several steps. First, feature points are extracted in each image using a Curvature Scale Space (CSS) based corner detector. Corresponding feature points are then matched between consecutive frames and tracked through the entire sequence. Using the motion parameters of the vehicle obtained from measuring the angular moment of the steering wheel and the distance information provided by the wheel encoders attached to the rear wheels of the vehicle, the distance of objects in the scene can be reasonably estimated. The algorithm is tested on several sets of real image sequences captured inside a parking lot. Some experimental results are presented and analyzed in detail.

Thesis Supervisor: Dr. Ichiro Masaki
Title: Principal Research Associate

Acknowledgments

I would like to express my deepest gratitude towards Prof. Berthold Horn and Dr. Ichiro Masaki for giving invaluable advice for this thesis. I also would like to thank Nicole S. Love and Yajun Fang for being so patient in helping me with all the work.

This thesis cannot be completed without all the help from Toyota Motor Corp. in Japan for taking and supplying data and being so helpful in answering all my questions. I am in debt to Ashish Koul Chanqing Zheng and Kevin Choi for laboriously proof-reading my thesis.

Thanks to all my friends who have been so supportive to me through all the hard times. I would like to give speical thanks to Rattapoom Tuchinda, Piyada Phanaphat, Samerkhae Jongthammanurak, Wanida Pongsaksawad, and Ratchatee Techapiesancharoenkij.

Above all, I would like to devote this thesis to Mom, Dad, and my two sisters who have been giving me their unconditional love and emotional support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The past few decades have seen a tremendous growth in the field of three-dimensional image analysis. This rapid expansion is due in part to the emerging popularity of the more recent applications of machine vision in the Intelligent Transportation System (ITS) domain. Increasing demands from a number of companies in the automobile industry and government agencies alike have spawned a large amount of research that has led to many new developments of more accurate and more reliable intelligent vehicle systems, e.g., passive navigation, automatic surveillance systems, etc.

1.1 Motivation

Several car companies have expressed their interests in manufacturing a new generation of road vehicles with intelligent functionalities that are specifically designed to suit aged drivers. These smart vehicles will have several additional features built into the system to alleviate, if not eliminate, the burden of driving. Aged drivers, therefore, will benefit a great deal from this new type of vehicle and will serve as the first group of target customers.

In this thesis, we focus on one intelligent system that could serve as an added feature in smart cars–a parking assistant system. The goal of this thesis is to develop a real-time system that aids aged drivers in various difficult parking scenarios. Specifically, two scenarios–parallel and right-angle parking–will be addressed in this

research. Figure 1-1 and Figure 1-2 illustrate the two cases. Next, we present the problem statement and the background research of various systems that have been proposed.

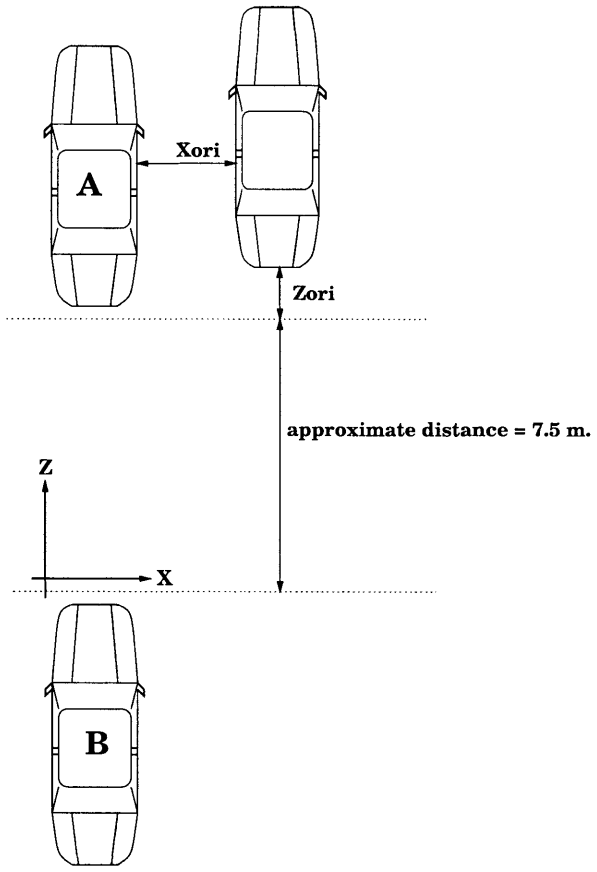


Figure 1-1: Bird's eye view of parallel parking

1.2 Problem Statement

In a conventional parking situation, drivers must approximate how much parking space is available and estimate the distance from the car and obstacles behind by looking back through the rear window. Many problems arise when small obstacles on the road, e.g., small children, are not visible through the rear window of minivans and SUV's and other vehicles of similar shape. The situation becomes even worse when parking is attempted under extreme weather conditions, e.g., rain, snow, fog,

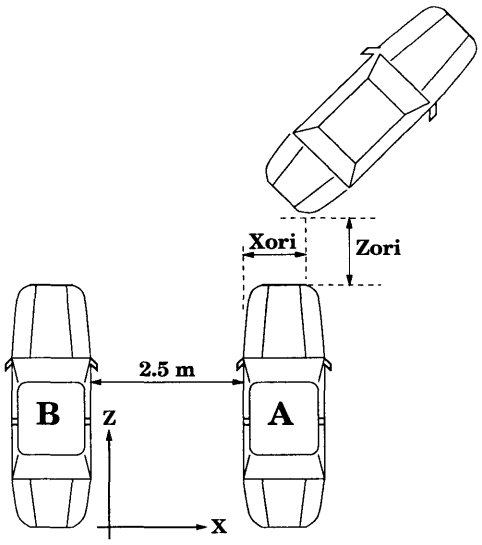


Figure 1-2: Bird's eye view of right-angle parking

etc. Under those circumstances, the driver's ability to estimate the proximity of obstacles in the scene severely deteriorates. The parking assistant system developed in this thesis proposes a solution to this and other similar problems in parking by providing the drivers with a 3-D distance information of obstacles in the scene. To achieve this goal, a smart vehicle must be equipped with one or more types of sensors through which the vehicle perceives the environment. The problem then becomes one of designing such a system that allows a vehicle to perceive the distance of objects in the scene and efficiently reconstructs a 3-D depth map of the environment. The next section provides the background and literature review of the research work that has been carried out.

1.3 Background

For several decades, the study and development of efficient depth recovery systems has been the focus of many research topics in different areas. Particularly in the field of autonomous robot applications where robots must rely entirely on their depth estimation system to safely guide and navigate them through the environment, the issue concerning the accuracy and reliability of the computed depth becomes greatly

emphasized. Various types of sensors with different strengths and weaknesses have been extensively explored in many depth recovery applications. This section presents a brief overview of some commonly used sensors.

Depending on their mechanisms to recover depth, range sensors can be categorized into two broad classes–active and passive sensors. Active sensory systems make use of probing signals to compute distance of objects in the scene. The concept behind depth computation using active sensory systems is based on the principle of triangulation. First, some types of signals are emitted into the environment in all directions. Upon hitting some obstacles in the scene, the signals are reflected back to the sensors. By measuring the time of flight of the returned signals, the distance of objects from the sensors can be calculated. Passive sensory systems, on the other hand, do not require additional signals in depth calculation. A light-based camera system is used to capture images of the scene from different angles. Based on the discrepancy of those images, depth can be recovered.

For different applications, different types of range sensors are preferred. Active range finders that use sonar for example, are commonly found in many autonomous robot systems that do not require a precise localization of objects in the scene. The long wavelength of sonars greatly limits the spatial resolution of the computed depth maps [17]. In other applications where the positions of obstacles in the environment need to be more localized, the use of higher frequency signals in detecting range is preferred. Millimeter-wave radar, for example, is gaining more popularity in ITS applications due to the high detection range and high signal penetration level that allows for its robust usage under various weather conditions, i.e., fog, rain, snow, etc. One problem, however, is the degraded system performance under the presence of signal interference. In addition, the high maintenance and operational costs associated with the use of millimeter-wave radar systems impose a major drawback to the practical implementation of such range finder systems [17].

Passive sensor systems, i.e., light-based sensors, on the contrary, do not bear the extra costs presented in their active system counterparts. Instead of emitting signals into the environment and measuring the time of flight of the returned signals,

depths are inferred from the differences in imaging the same objects in different views. In ITS applications, passive sensor systems have many more advantages over active sensors besides being more economical. Among those, we found that 3-D reconstructed maps of the environment obtained using passive sensors have a much higher spatial resolution. Therefore, in this study we will limit ourselves to the design and implementation of a vision-based (camera-based) navigation system. The next section reviews the literature on many existing vision-based systems.

1.3.1 Vision-based Navigation System

Vision-based navigation systems use cameras as their means of retrieving information from the outside world. Using a perspective projection model, i.e., a pinhole camera model, light reflecting off a three-dimensional object must pass through the *center of projection* (COP) of a camera. Figure 1-3 illustrates the mechanism of perspective projection. As seen in Figure 1-3, point (x', y') on the image plane can be a projection of point (X_1, Y_1, Z_1), or (X_2, Y_2, Z_2), or (X_3, Y_3, Z_3), or any (X_i, Y_i, Z_i) that lies along the perspective ray that intersects the image plane at point (x', y'). This observation implies that the mapping between points (X, Y, Z) in the environment and points (x', y') on the image plane is not unique. Specifically, in projecting a 3D object onto a 2D image plane, the depth, Z, is lost. The perspective projection equations relate x', y', X, Y, Z, and the focal length of the camera, f, as follows:

$$x' = \frac{X \cdot f}{Z} \quad and \quad y' = \frac{Y \cdot f}{Z}.$$

Since only the measurements of x', y', and the camera geometry, f are available, this perspective projection relationship implies that without *a priori* information of X or Y, the depth, Z, cannot be recovered from an analysis of one image alone.

Marr and Poggio proposed their theories in [16] on the mechanisms of human stereopsis that led to the development of depth recovery systems using binocular stereo vision (two-camera setup). Stereo vision has since then become the dominant topic receiving a wide array of attention from many groups of researchers. Consider a

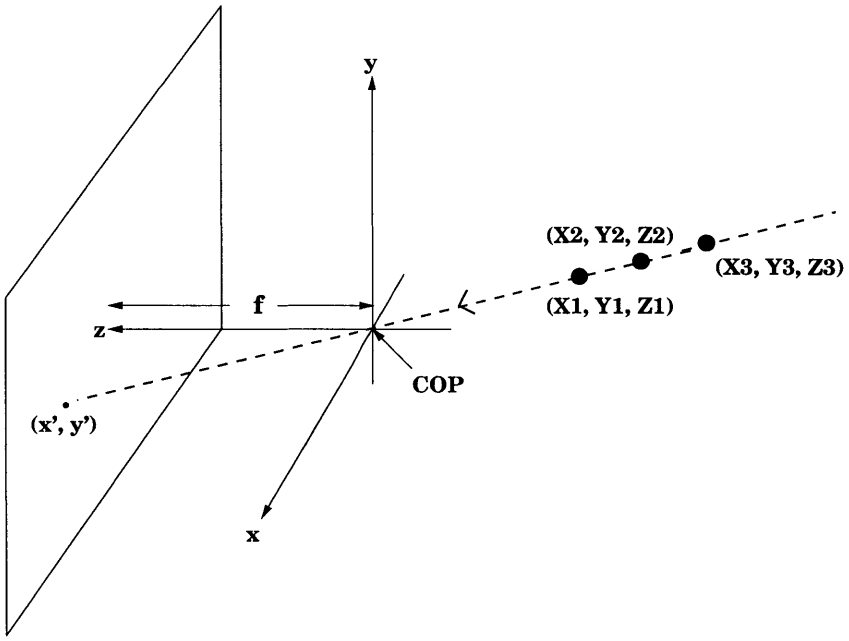


Figure 1-3: A pinhole camera model illustrating perspective projection. Points (X_1, Y_1, Z_1), (X_2, Y_2, Z_2), and (X_3, Y_3, Z_3) are imaged onto the same location (x', y') on the image plane

basic setup as shown in Figure 1-4 where the optical axes of the two camera stations are parallel and separated by a distance B. The concept behind a depth recovery process of binocular stereo vision lies in the displacement of the locations of (x'_l, y'_l) and (x'_r, y'_r), the corresponding projections of the same point $p = (X, Y, Z)$ on the two image planes. This difference in pixel locations, $x'_l - x'_r$, is referred to as the disparity between the conjugate points on the two images. By using the disparity measured in the two images together with the known camera focal length, the depth, Z, can be computed using the following relationship:

$$Z = \frac{B \cdot f}{d}$$

where B denotes the length of the baseline (the line connecting the two camera centers), f denotes the focal lengths of the cameras (assuming the two cameras have the same focal length), and d denotes the disparity value.

The key problem in binocular stereo is how to accurately locate the corresponding

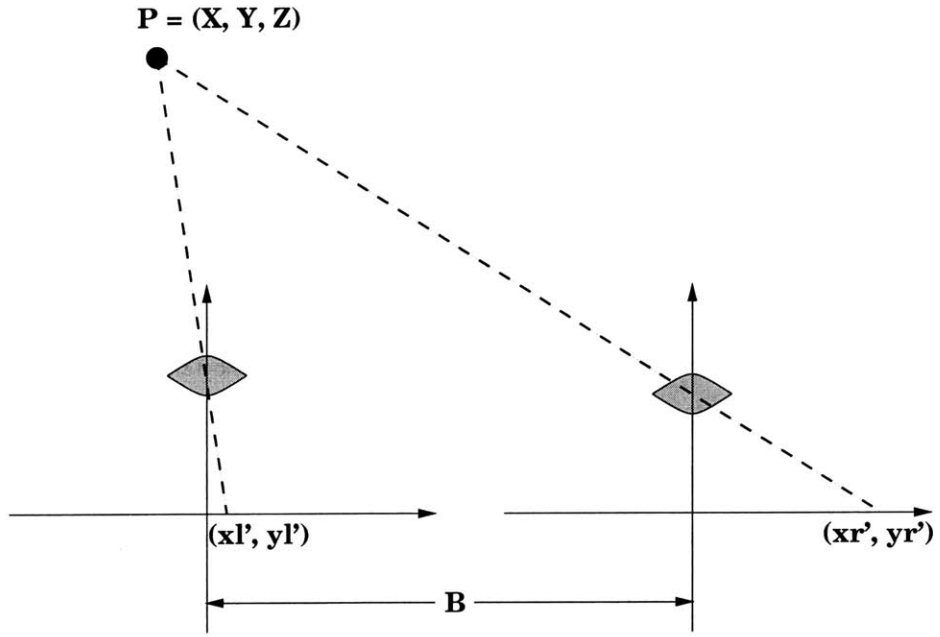


Figure 1-4: Binocular stereo setup. The optical axes of the two cameras are parallel and perpendicular to the line connecting the two camera centers (baseline)

points between a stereo pair of images. Many groups of researchers have studied this correspondence problem over the years and have proposed different algorithms to solve the problem. In general, we can broadly classify these methods into two primary divisions, area-based and feature-based correspondence. It is observed that in the neighborhood of the conjugate points, we should expect to see similar brightness intensity patterns. Area-based approaches, therefore, make use of some forms of cross-correlation functions to measure the similarity between intensity patterns within the local windows of the two images. Given a patch in one image, we compute the correlation with all possible patches in the other image. The point with the largest correlation value is then selected to be the match. The result of area-based stereo correspondence is a dense depth map of the environment. In [1], Baker shows that area-based correspondence has been successfully applied to the stereo analysis of rolling terrain scenes. However, many problems arise when the scene is not smoothly varying and continuous, e.g., scenes in urban settings. In images from such domains many windows to be matched will not have correspondences in the other image due

to the occlusion of one object behind another.

In feature-based correspondence, however, different approaches are taken to tackle the correspondence problem. Feature-based methods make use of the observation that disparity should only be computed over the patches that have more or less unique intensity fluctuations. Two steps are usually involved in feature-based correspondence. First, the left and right images are separately analyzed to select the area with distinctive gray-level patterns to perform the matching. This process corresponds to picking the "*feature points*" in the images that have high brightness intensity gradients as on the edges or corner points. The second step is to find the feature points in the right image that correspond to the matching feature points in the left image. Many algorithms define some auxiliary information, e.g., the difference in intensity values across the edge, to help constrain the matching. Two edges are declared a match when their corresponding auxiliary information is closely similar.

In our simple binocular stereo system–two cameras with parallel optical axes as shown in Figure 1-4, one additional important constraint can be incorporated to assist the matching. First, define an epipolar plane as a plane formed by the line connecting the two camera centers, COP_1 and COP_2, and the line connecting point P of interest to one of the centers of projection. The intersection of the epipolar plane, i.e., the plane $P - (x'_l, y'_l) - (x'_r, y'_r)$ as seen in Figure 1-5, with the two image planes results in two corresponding epipolar lines. In this camera setting where the two camera stations have parallel optical axes, the epipolar lines correspond to the two horizontal lines as seen in Figure 1-5. This observation allows us to limit the search range for corresponding points to be along the matching epipolar lines, thus reducing the possibility of false matches. In practice, however, even for the simple case where disparity is restricted to a horizontal displacement (the epipolar lines correspond to horizontal lines), mismatches still occur when gray-level intensity patterns are inherently similar. The occurrence of false correspondence can severely affect the accuracy of the subsequent depth calculation. In addition, searching for corresponding points can be a computationally intensive task. For real-time applications, several techniques such as dynamic programming and parallel processing must be employed

to speed up the computation. The feature-based correspondence scheme allows for depth to be computed only at the feature locations; hence, a sparse depth map is obtained. This quality could present drawbacks in many applications where dense depth maps are preferred.

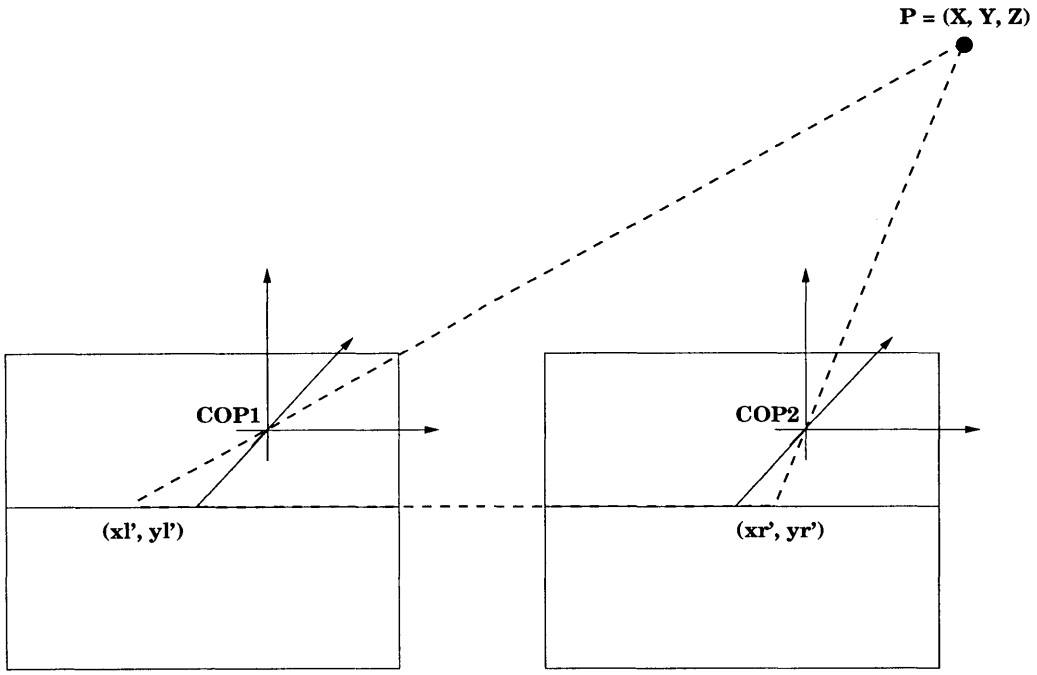


Figure 1-5: An epipolar plane is defined as a plane formed by the line connecting the two centers of projection (COP), and the line connecting point P to one of the COPs. The intersection between the epipolar plane, plane $P-(x'_l, y'_l)-(x'_r, y'_r)$, and the two image planes are the two corresponding epipolar lines. In a binocular stereo setup with parallel optical axes, the two corresponding epipolar lines are simply horizontal.

In fact, it has been shown that the number of correspondence mismatches can be dramatically reduced, and thus performance can be improved, by introducing more camera geometry constraints. As Chellappa and Shekhar pointed out in [27], the trinocular system usually yields more robust 3D depth determination than the traditional binocular stereo system. With a multi-ocular stereo paradigm, the correspondence problem should then become even more constrained and the matching more simplified. Nevertheless, in many practical applications where the cost and maintenance of multiple camera systems are not realizable, the use of monocular vision system for a 3-D reconstruction of the environment becomes necessary. We now

turn ourselves to the study of depth recovery using monocular vision systems. The next section presents a literature review and a thorough discussion of different methods and algorithms used in the problem of 3-D reconstruction of the environment and motion analysis using a monocular stereo system.

1.3.2 Monocular Stereo

Contrary to the use of multiple stationary cameras in the case of multi-ocular systems, in monocular vision systems, depth is recovered from a sequence of images taken from a single moving camera. To simplify the analysis, it is assumed that all objects in the scene remain stationary, i.e., the camera is moving through a stationary environment. Based on very different assumptions, many algorithms have been proposed and developed to suit specific tasks and constraints. In most algorithms for monocular stereo, the first step usually requires that feature points be extracted from each image in the sequence and corresponding feature points be matched. However, if the motion of the camera becomes unknown, the epipolar geometry constraint that helps limit the search range for correspondence similar to the case in binocular stereo no longer exists. Many researchers employ a technique that exploits the inherent topological relationship, e.g., inter-point distance, between the feature points to help constrain the matching process. In [8], [27], [26] and [34], for example, ambiguity in matching has been shown to reduce by imposing topological constraints among feature points to disambiguate false matches. When a long sequence of images are used, which often is the case in monocular stereo vision systems, feature points are first matched between subsequent images and then tracked over multiple frames to improve reliability.

In order to compute depths after correspondence has been found, the precise camera motion between two consecutive frames in the sequence must first be determined. For the case where the camera motion can be safely assumed to be a pure translation, the problem of depth recovery is simplified. In [8] and [22], the camera is assumed to move along a straight line in the direction of its optical axis. With the known feature correspondence, Murphy et al. [22] used the principle of triangulation as shown in

Figure 1-6 to prove the following relationship:

$$R_2 \approx \frac{D_1}{D_2 - D_1}(R_1 - R_2)$$

where R_1 represents the distance of point P from the camera station at time t_1; R_2 represents the distance of point P from the camera station at time t_2; D_1 represents the x-coordinate of the projection of point P on the image plane at time t_1; and D_2 represents the x-coordinate of the projection of point P on the image plane at time t_1. The depth map obtained from this algorithm is only a relative depth map of the scene if $R_1 - R_2$ is unknown. To obtain an absolute depth map, the distance that the camera moves between time t_1 and t_2 must be determined *a priori* or obtained from other types of sensors. Another method to determine depth from an image sequence

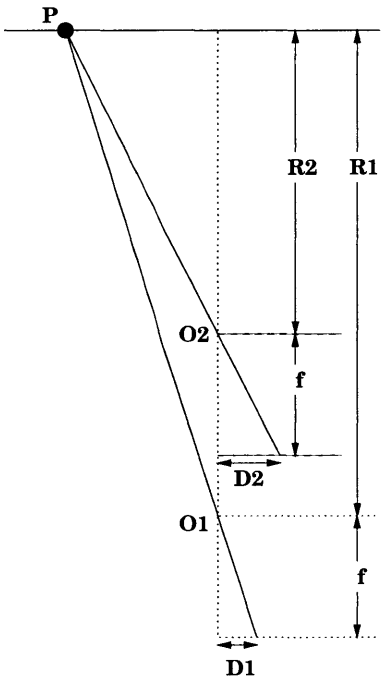


Figure 1-6: When the motion between the two camera stations is restricted to be a pure translation along the optical axis of the camera, the depth, R_2, can be approximated using the relationship: $R_2 \approx \frac{D_1}{D_2 - D_1}(R_1 - R_2)$

when the camera motion is restricted to a pure translation was proposed by Dalmia and Trivedi in [6]. Based on the spatio-temporal gradients (STG) technique, Dalmia

et al. uses the relationship between spatial and temporal gradients of brightness intensity to compute depth. In particular, STG imposes a constraint that the value of a temporal gradient be proportional to the spatial gradient computed at the same location. The STG constraint equation for the case of a pure translation can be formulated as follows. Let $(I_0, I_1, I_2, ..., I_N)$ denote a sequence of images taken at times $(t_0, t_1, t_2, ..., t_N)$. Let $g_0(n)$ be the gray level intensity value at location n in image I_0 at time t_0. Without considering the effect of noise, we should expect the intensity value at location n of image I_0 to be the same as that of location $n + \delta$ in image I_j, that is, $g_0(n) = g_j(n + \delta)$. Subtracting $g_j(n)$ on both sides, we have

$$g_0(n) - g_j(n) = g_j(n + \delta) - g_j(n).$$

The left-hand side corresponds to the temporal gradient and the right-hand side the spatial gradient at location n. Rewriting the above equation, we obtain

$$\delta_t g_0(n) = \delta_s g_j(n)$$

where δ_s denotes the drift in space, and δ_t denotes the drift in time.

One advantage of the STG technique is that the analysis cleverly avoids the time-consuming correspondence problem by calculating depth directly from spatial and temporal gradients. Computing gradients for STG algorithm is computationally simple and can be done in real-time with the help of specialized hardware [6]. However, STG uses the pixel intensity values directly to compute both spatial and temporal gradients. This technique can thus be very susceptible to quantization and brightness measurement noise.

One major drawback common to the three above depth recovery algorithms in [8], [22], and [6] is the restricted camera motion assumption under which these algorithms can be applied. In application areas such as mobile robot navigation systems, where it is possible to physically link a camera to the wheels of the robot to allow for a control of the orientation of the optical axis of the camera to align with the camera motion, Zhang's and Fujii's algorithms have been shown to work rather efficiently

and accurately. However, for the case where the optical axis of the camera is fixed in one particular orientation, the rotational and translational motion off the optical axis direction becomes unavoidable. In fact, the camera rotation and translation off the optical axis direction are claimed to be the primary attribute of large errors in the depth calculation [22].

For the case when the camera motion is unknown and is allowed to be arbitrary, the problem of depth recovery becomes extremely complicated. First, the arbitrary transformation between the camera stations of two consecutive image frames in the sequence must be estimated. The problem of recovering the relative orientation and position of two image planes relative to each other using a set of known corresponding points is the classic relative orientation problem. Let the terms *left* and *right* denote the earlier and later frames in the sequence. Consider a point $p = (X, Y, Z)$ in the world coordinate; its location as described in the *left* and *right* camera coordinates are $p_l = (x_l, y_l, z_l)$ and $p_r = (x_r, y_r, z_r)$ respectively. Let $p'_l = (x'_l, y'_l)$ be the projection of point $p = (X, Y, Z)$ on the *left* image plane and $p'_r = (x'_r, y'_r)$ be the projection of the same point on the *right* image plane. Under the assumption of a rigid body motion between the *left* and the *right* camera coordinates, the transformation can be represented by a rotation followed by a translation. That is, the relationship between p_l and p_r is described as

$$p_r = \mathbf{R}p_l + \mathbf{t}$$

where $\mathbf{R}$ denotes a rotation matrix

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

and $\mathbf{t}$ a translation vector $[X_0, Y_0, Z_0]^T$. Plugging in x'_l, x'_r, y'_l, and y'_r from the perspective projection equations,

$$\frac{x'_l}{f} = \frac{x_l}{z_l} \quad and \quad \frac{y'_l}{f} = \frac{y_l}{z_l},$$

and

$$\frac{x'_r}{f} = \frac{x_r}{z_r} \quad and \quad \frac{y'_r}{f} = \frac{y_r}{z_r},$$

we find that for a given pair of corresponding points, we have the following constraints [12]:

$$r_{11}x'_l + r_{12}y'_l + r_{13}f + X_0\frac{f}{z_l} = x'_r\frac{z_r}{z_l} \tag{1.1}$$

$$r_{21}x'_l + r_{22}y'_l + r_{23}f + Y_0\frac{f}{z_l} = y'_r\frac{z_r}{z_l} \tag{1.2}$$

$$r_{31}x'_l + r_{32}y'_l + r_{33}f + Z_0\frac{f}{z_l} = f\frac{z_r}{z_l}. \tag{1.3}$$

There are 3 equations and 14 unknowns $r_{11}, r_{12}, ..., Y_0, Z_0, z_l, z_r$. Each additional point pair adds 3 more equations but also introduces 2 more unknowns. By additionally constraining the rotation matrix $\mathbf{R}$ to be orthonormal, we introduce 3 more constraints:

$$\begin{aligned} r_{11}^2 + r_{12}^2 + r_{13}^2 &= 1 \\ r_{21}^2 + r_{22}^2 + r_{23}^2 &= 1 \\ r_{31}^2 + r_{32}^2 + r_{33}^2 &= 1. \end{aligned}$$

Given n point pairs, we have $12 + 2n$ unknowns and $7 + 3n$ equations. Finsterwalder established that at least 5 corresponding point positions must be known to obtain a finite number of solutions to the relative orientation problem [12]. In practice, however, more than 5 correspondence matches are usually obtained and the least square method is often used to attain more accurate solutions. Nevertheless, due to a highly nonlinear nature of this system of equations, most iterative schemes proposed suffer a convergence problem and require a good initial guess to arrive at a correct solution. In [13], Horn proposed a least square iterative method that does not require a good initial guess using the quarternion representation. Once the transformation between the *left* and *right* image planes are identified, depth can be computed using Eq.(1.1), (1.2), (1.3). The problem with recovering depth using the relative orientation scheme is that the process can be very computationally intensive. As the focus of

this research is on developing an efficient parking assistant system that can operate in real time, this property can thus be highly undesirable.

Another approach to recover motion and depth that uses Extended Kalman Filtering (EKF) as a prediction-and-correction tool has been extensively explored by many groups of researchers. In [27], features are first extracted and matched using a label-graph matching technique. Under the assumption of smoothness of motion, Shekhar and Chellappa proposed an algorithm based on Iterated Extended Kalman Filtering (IEKF) technique of recursive estimation to predict the motion and range as the state parameters of the filter, which are updated at each iteration. The motion parameters are then used to predict where the feature points will lie in the next frame. The closeness of the matched points to their predicted locations is in turn used to update the state parameters in the next iteration. This recursive technique has also been applied to other applications such as facial expression recognition with satisfactory results [27].

In [7], Graefe and Dickmanns uses a combination of vision and inertial sensors to solve the problem of 3-D reconstruction of the environment for an intelligent vehicle application. Their method integrates the information on angular velocity of the vehicle from the inertial sensors and the translational velocity from the camera (vision sensor). Feature points are first extracted, then the matching and tracking processes are constrained using EKF technique to find predicted locations of the edge points of interest. The search range for matching feature points in the next image frame in the sequence is reduced using this technique and correspondence mismatching problems have been shown to improve.

As with other motion estimation methods, the results from motion estimation are used to compute depth from motion. A small estimation error in motion prediction can cause a large-scale error in depth approximation. In an application that requires a more precise depth estimation, some other depth recovery methods are preferred.

In navigating mobile robot systems, more than one type of sensors are usually incorporated into the system for the purpose of adding redundant information and providing fault-tolerance. Inertial sensors, in particular, are often used to provide the

robot with the information of its own motion (angular velocity, speed, and direction). Some examples of such systems are discussed in more detail in [27] and [7]. In our vision-aided driving assistant application, inertial sensors are used to provide the angular velocity of the vehicle and the wheel encoders are used to provide the distance information that the car travels. When the motion of the camera is known, i.e., the relative orientation of the *left* and *right* image planes is known, the problem of 3-D reconstruction of the environment is then simplified to the recovery of depth from a binocular stereo system with non-parallel axes stereo geometry. The next section gives an overview of the work that has been done for this thesis.

1.4 Overview of this Thesis

This thesis presents an algorithm for a robust depth recovery system used in the parking assistant application. The system takes as its input a sequence of images captured from a video camera mounted on the back of a testing van overlooking the rear view. The goal is to reconstruct a 3-D depth map of the environment in the parking lot from a long sequence of monocular images.

The flow diagram in Figure 1-7 shows an overview of our algorithm. As in many other depth recovery systems described above, the first step in our implementation is to extract feature points to be matched from each image in the sequence. We explored several types of features that have been used in many existing systems and found that by selecting reliable feature detectors, the subsequent step in matching and tracking corresponding feature points through the image sequence can be simplified. Corners are chosen as the implementation of choice. Chapter 2 first presents a brief literature review of many existing corner detection algorithms. Our implementation is based on the Curvature Scale Space (CSS) method, which has been proven to perform well on video images that have been degraded by a motion blur. The implementation details of the CSS based corner detection algorithm are discussed, and experimental results from testing the algorithm on both synthetic and real imagery are presented.

After extracting feature points in each image in the sequence, we proceed to match

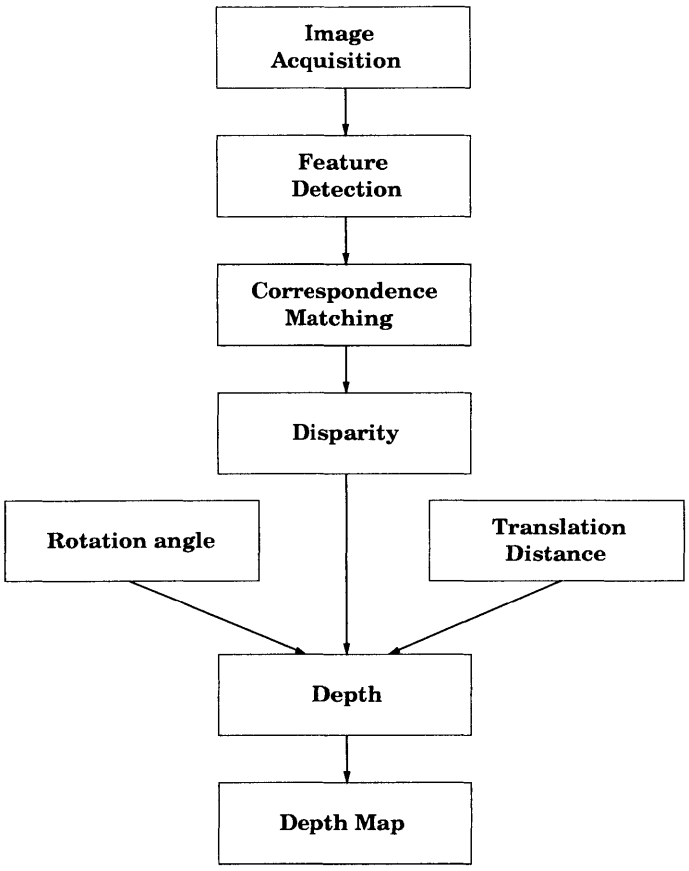


Figure 1-7: Flow chart of depth recovery algorithm for the Parking Assistant System

the corresponding feature points in different frames. Chapter 3 first reviews several techniques to reduce correspondence mismatches in both binocular and monocular stereo settings. The algorithm used in this thesis exploits the densely sampled structure of the video sequence and allows for correspondence matching to be done efficiently between consecutive frames in the image sequence. Feature points are then tracked through the entire sequence using a robust tracking algorithm which performs well even when features points disappear in some part of the sequence.

After feature points are matched and tracked, depth can then be computed using the known camera motion and the difference in locations of corresponding points in any two arbitrary frames in the sequence. Specifically, the precise transformation between the camera coordinates of the two image frames is obtained using the inertial sensors to measure the rotation angle and the wheel encoders to measure the transla-

tion distance of the vehicle. Chapter 4 describes in detail how the motion parameters of the camera are obtained. Some experimental results of the estimated depth are compared with the actual measured distance of objects in the environment in chapter 5. Sparse depth maps of the environment in the parking lot are also generated. In chapter 6, conclusions are drawn from the experimental results and some suggestions for future work are discussed.

Chapter 2

Feature Extraction

The problem of detection and localization of edge contour and feature points has been found to play a central role in a wide array of image analysis systems. Many computer vision applications, e.g., object recognition and motion tracking, require that objects in the scene be tracked over a long sequence of video frames. The problem of detecting and localizing reliable feature points in the input images to match or track thus lies at the heart of those applications.

2.1 Corners

Corner feature points have gained more popularity over the years and have become one of the most widely used features in most machine vision tasks for several reasons. In motion analysis, for example, Horn showed that optical flow velocities computed around the corner points do not suffer from the aperture problem and thus can be determined without ambiguity using the optical flow constraint equation [12]. For motion estimation purpose, the moving corners thus provide very rich and reliable information [33]. Another example is from stereo vision application where disparities are computed by matching corresponding points between a pair of images. Several local characteristics of corners, e.g., brightness intensity variation pattern in the local neighborhood or the absolute curvature value at a corner point, are often used to aid correspondence matching. The accuracy and reliability of these dynamic vi-

sion systems therefore heavily depends on the extraction of corner points from input images.

Section 2.2 first reviews the literature on several existing corner detection algorithms. Section 2.3 describes the details of a Curvature Scale Space based corner detection scheme implemented in this thesis for the parking assistant application. The experimental results from testing our algorithm on both synthetic and real imagery are presented at the end of this chapter in section 2.4.

2.2 Literature Review

Over the past few decades, a great deal of research effort has been invested in developing reliable corner detection algorithms. Depending on different mathematical models used in describing corners, several different techniques have been proposed. Moravec [21] first observed that the average intensity value change around a neighborhood of a corner is significantly high in all directions. Figure 2-1 illustrates Moravec's observation. Across the neighborhood of a flat intensity region (labeled as 1 in Figure 2-1), the intensity change will be small in all directions, while in the neighborhood of an edge (label 2), the change will be large only in the direction of the edge gradient. Harris [10] proposed an improvement to Moravec's idea and used a Gaussian smoothing to aid in the calculation of the first and second derivative of image intensity values to determine a corner strength. His algorithm, called the Plessey corner detector, later became well-known for its detection reliability. Kitchen and Rosenfeld [14] defined a corner as a point where the rate of change of gradient direction is maximum and proposed a corner detector that finds the local maxima of the gradient direction change based on a differential operator that computes the first and second-order partial derivatives of image intensity. Mehrotra [18] observed that most corner detectors were developed based on edge detection algorithms, which performed poorly on corners. Mehrotra then proposed an approach in [18] based on applying the first directional derivative of a Gaussian operator to detect half edges and defined a corner as a junction where two half edges meet at an angle less than 180^o. In [31], Xie pro-

posed a method to detect corners that combines many different cues–edge strength, curvature, and region dissimilarity–in a cost function to be minimized. For a more extensive cover of other existing corner detectors, refer to [20]. All of the above de-

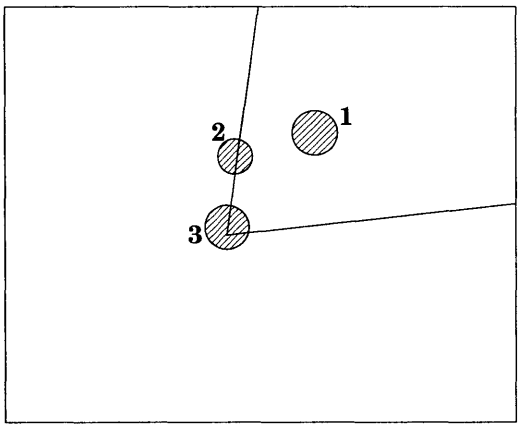


Figure 2-1: The intensity variation over a local neighborhood of an edge point will be high only in the direction perpendicular to the edge direction. For a region corresponding to a corner however, Moravec observed that the variation must be high in all directions

scribed algorithms, however, do not perform well when the image quality is poor. Unfortunately, in most dynamic vision systems, image capturing devices usually undergo some kinds of motion with respect to the objects in the scene; images thus obtained under those circumstances experience some degrees of motion blur. Consequently, none of the approaches to detect corners described above yield a strikingly superior performance.

To make our approach more robust against noise in the input imagery, the corner detection algorithm implemented in this research has been designed to exploit the idea of multi-resolution analysis, i.e., detecting corners at multiple scales. The algorithm starts out by first locating corner points at a coarse resolution; hence, only true corners are extracted. To obtain a better localization, the detected corners are tracked through several finer scales, making the detection and localization scheme of this algorithm more robust against noise in input imagery. In the next section, the outline of the multi-scale corner detection algorithm as well as the implementation details of the individual steps will be presented.

2.3 Multi-scale Corner Detection Algorithm

Research in psychophysics and physiological experiments has shown that the visual cortex of mammals processes signals using some types of multi-scale transform [15]. This evidence implies that multi-scale information processing is inherent to many natural systems. In fact, many researchers such as Burt and Adelson [3], Marr [16], and Witkin [30] had long realized the central importance of extracting information from an image at various levels of details. In recent years, some of these ideas have been formalized into multi-resolution analysis (MRA) scheme of the wavelet analysis [15].

Our implementation of the multi-scale corner detection algorithm is based primarily on the Curvature Scale Space (CSS) technique discussed in more detail in [20]. In the CSS technique, edge contours of objects in the scene are first extracted from an input image using an edge detector. A curvature value is then assigned to each point along the contour based on the instantaneous rate of change of the slope angle at that point [19]. A corner, defined as a point where the maximum rate of change of the slope angle occurs, can be detected by locating the local maxima on the curvature graph. The concept of multi-resolution processing is then applied at this stage by computing curvature at multiple scales. Corners are first extracted at a coarse scale where all the ripple noise along the contour is smoothed out, hence only true corners are allowed to exist and be detected. Then to improve corner localization, the detected corners are tracked through several finer scales, increasing precision in the detected corner positions. The outline of the steps involved in CSS based multi-scale corner detection can be summarized in the flow diagram in Figure 2-2.

This scale-space representation of corners is efficient in its robustness against noise in input imagery. In addition, this representation allows for both the points of local maxima, i.e., corner points, and the zero crossings of curvature graphs to be used as features to be matched or tracked.

In the next section, the implementation details of the individual step will be discussed.

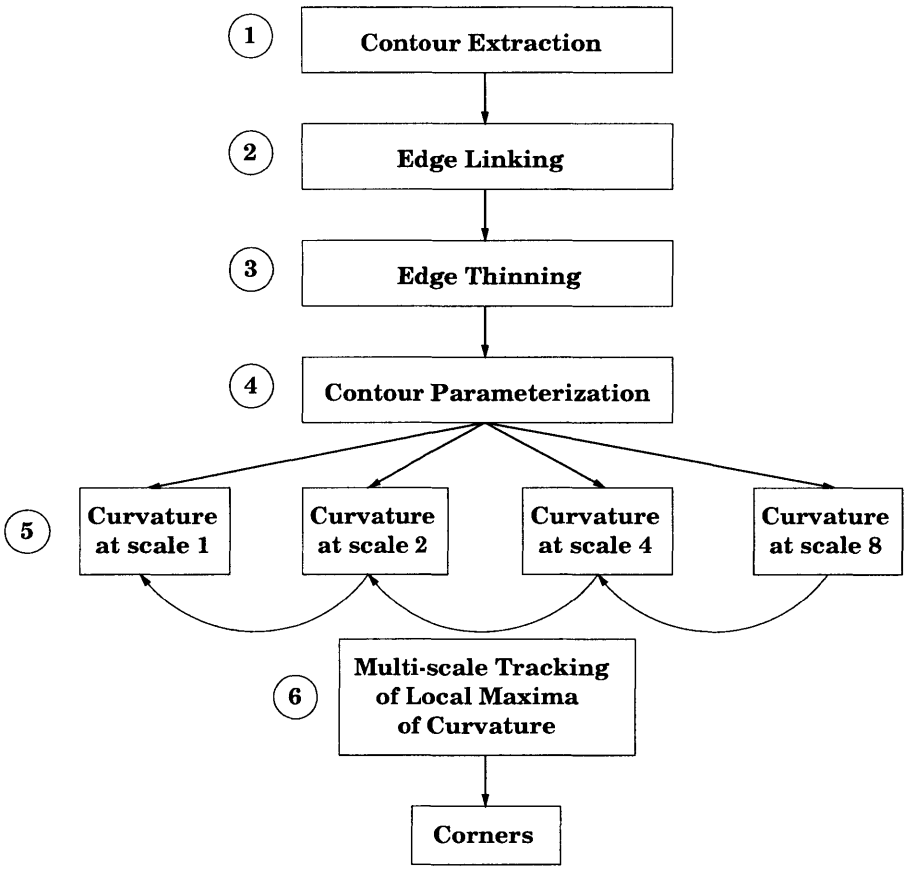


Figure 2-2: Flow diagram of the CSS based corner detection algorithm

2.3.1 Edge Contour Extraction

The first step in the CSS based multi-scale corner detection is the extraction of edge contours from an intensity image. Since the output from this edge detection stage will be used to compute curvature in a later stage, an ideal edge detector should detect all the edges in an input image and output edge contours that are thin and connected. In experimenting with several detectors such as Sobel, Canny [4], SUSAN [28], and half-edge detector [18], we found that the Canny detector consistently outperformed other detectors in its detection reliability. Moreover, Canny has a built-in non-maximum suppression capability. Hence, the edge contour output from Canny is thinned to one pixel wide. This allows for some computational savings in the subsequent thinning stage. Therefore, our implementation will make use of the Canny detector to extract edge contours from input images.

When the input images are very noisy, however, experiments have shown that it is insufficient to detect edges from the gray-scale intensity values alone. Novak et al. concluded in [23] that 10% of the edges detected by a color gradient-based edge detector cannot be identified with analysis on gray-scale images. An experiment was conducted by applying the Canny operator to detect edges separately on each color component, R, G, and B of the input images. The three edge maps are then OR'ed together to obtain a more defined and more connected edge response. Figure 2-3a and Figure 2-3b compare the two binary edge maps. In Figure 2-3a, a color image of a parking lot is converted to a gray-scale image and we apply Canny edge detector to detect edge contours. In Figure 2-3b, Canny operator is used to detect edges in each of R, G, and B components separately. As apparent in Figure 2-3, the edge detection scheme that makes use of color information has the advantage of discovering edges in the area that have similar gray-scale intensity values but different intensity in each color component. A drawback of this method is that the amount of time to extract edge contours is now tripled. In addition, the contours obtained from OR'ing the three binary edge maps appear thick and thus need to be thinned before they can be parameterized.

2.3.2 Edge Linking and Thinning

The edge contours obtained from the first stage must be linked and thinned before proceeding to the parameterization stage. In this step, edge linking is designed to fill in the gap between unconnected edge contours and eliminate short dangling edge responses. Thinning is done after the contours have been linked. The goal is to produce edge contour output with one pixel in thickness to allow for contour parameterization to be simplified.

Edge Linking

By OR'ing the three binary edge responses from the R, G, and B channels, the output edge contours appear more continuous. In fact, most unconnected contours

have automatically been linked due to the slightly different response from each color component to the same edge operator. Therefore, the edge linking scheme used in this stage can be simplified. By looking at the intensity pattern on a 3x3 local window as shown in Figure 2-4, the gap will be filled when the pattern fits into one of the possible patterns to link. For a more detailed discussion of this algorithm, refer to Appendix A.

Edge Thinning

After filling in the gaps to link unconnected pieces of edge contours, the output edge results vary in thickness. We then proceed to thin the contours. The goal is to remove edge pixels until only the core "skeleton" of the contours remain. A pixel in the skeleton can have only two other neighboring edge pixels in the 3x3 window surrounding it, unless it is at the end of an edge, i.e. break point, or at a branch point. The edge thinning scheme used in this thesis is based on a combination of several edge thinning algorithms proposed in literature. One approach presented in [29] is based on an algorithm by Heipke et al. [11] that inspects a 3x3 neighborhood of each edge pixel to determine if the center pixel belongs to the skeleton. Certain conditions on proper edge thinning reduce the number of 3x3 possible combinations to 17 masks. The edge pixel in the center is not considered part of the skeleton and thus will be removed if the 3x3 neighborhood of an edge pixel matches the pattern on any of the 17 masks. A more detailed discussion of this thinning algorithm can be found in Appendix B.

Another edge thinning scheme that is explored in this thesis is based on an algorithm proposed by Chen and Hsu [5]. Chen and Hsu developed a modified version of Zhang-Suen algorithm that is widely used in many optical character recognition (OCR) systems. Chen and Hsu's algorithm looks again at the 3x3 window of an edge pixel and proposed several logical tests based on the 4-neighbor and 8-neighbor connectivities. This method has an apparent advantage of contour noise immunity. The output of this algorithm is thinned edge contours with less noise ripples, when compared to Heipke's method. Refer to Appendix C of this thesis for a more detailed

discussion of this algorithm.

Figure 2-5 shows the results of step 1, 2, and 3, of our corner detection algorithm. Figure 2-5a shows the color image of a car obtained from a moving camera; the image is thus slightly blurry. Figure 2-5b shows the detected edge contours obtained from the contour extraction (step 1). As apparent in Figure 2-5b, the contours are thick and thus need to be thinned before parameterizing. Figure 2-5c shows the result after edge linking and thinning (step 2 and 3).

2.3.3 Contour Parameterization

After edge contours have been linked and thinned, we proceed to calculate the curvature value at each point along the contour. The first step is to label all the points on the contour starting from an end point. Since after the thinning stage, the edge contours have been thinned so that each pixel in the "skeleton" contains 2 neighboring edge pixels in the 3x3 neighborhood surrounding it; the pixels can then be labeled and tracked using the 8-neighbor connectivity. As shown in Figure 2-6, we start parameterizing this contour from an end point P_1 where the center edge pixel has only 1 neighbor. The next point to label is then P_2, which is the neighbor of P_1. The following point to label is then P_3 which is the neighbor of P_2 that has not been labeled. This process continues until another end point is found which terminates the contour. In Figure 2-6, if the starting point of parameterization is P_1, then the contour terminates when either P_5 or P_7 are reached. A problem occurs, however, at the branch points where the center edge pixel has more than 2 neighbors. In this case, the contour must then be split into 2 separate parameterization, $P_1 - P_2 - P_3 - P_4 - P_5$ and $P_1 - P_2 - P_3 - P_6 - P_7$, and curvature must then be computed separately for each contour.

2.3.4 Curvature Computation

Let $i = 1, 2, 3..., N$ be the labels of points along the contour Γ of length N. Many algorithms have been proposed to find the curvature value at each i along the contour.

One method proposed by Rosenfeld and Johnston [25] defines a curvature at point p_i to be the k-cosine of the angle of two vectors of length k that intersect at point p_i. Consider a number of subsequent and previous points from p_i in the sequence as candidates for the arms of a potential corner at p_i. For a positive integer k the forward and backward k-vector at point p_i are defined as

$$a_{ik} = (x_i - x_{i+k}, y_i - y_{i+k}) = (X_{ik}^+, Y_{ik}^+)$$

$$b_{ik} = (x_i - x_{i-k}, y_i - y_{i-k}) = (X_{ik}^-, Y_{ik}^-)$$

where X_{ik}^+, Y_{ik}^+, X_{ik}^-, Y_{ik}^- are the components of a_{ik} and b_{ik}, respectively. Curvature as defined as the k-cosine of the angle between the k-vectors is computed using dot-product relationship,

$$c_{ik} = \frac{(a_{ik} \cdot b_{ik})}{|a_{ik}||b_{ik}|}.$$

This method, however, has proven to be very sensitive to the noisy ripples on the curve. Another method to compute curvature is based on the theory of a multi-scale description of space curves proposed by Mokhtarian [20]. Consider an edge contour as a planar curve, $\Gamma(x, y)$. By parameterizing a contour with respect to the arc-length parameter t we express it in terms of 2 functions:

$$\Gamma(t) = (X(t), Y(t)).$$

Curvature can then be computed using the following definition [20]:

$$K(t) = \frac{\dot{X}(t)\,\ddot{Y}(t) - \ddot{X}(t)\,\dot{Y}(t)}{(\dot{X}(t)^2 + \dot{Y}(t)^2)^{1.5}}.$$

We can make this method of computing curvature more robust against noisy ripples on the curve by first smoothing out the contour with a Gaussian kernel, $g(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}}$. That is, $\dot{X}(t)$ can be computed by taking the first derivative of the smooth contour, $X(t) * g(t)$, with respect to t, i.e. $\dot{X}(t) \approx \frac{d(X(t)*g(t))}{dt}$. Since convolution and derivative are both linear operators, their orders can be interchanged.

We therefore obtain,

$$\dot{X}(t) \approx \frac{d(X(t) * g(t))}{dt} = X(t) * \frac{dg(t)}{dt}.$$

$\ddot{X}(t)$, $\dot{Y}(t)$, and $\ddot{Y}(t)$ can be computed in a similar manner. Thus we obtain,

$$\ddot{X}(t) \approx \frac{d^2(X(t) * g(t))}{dt^2} = X(t) * \frac{d^2g(t)}{dt^2},$$

$$\dot{Y}(t) \approx \frac{d(Y(t) * g(t))}{dt} = Y(t) * \frac{dg(t)}{dt},$$

$$\ddot{Y}(t) \approx \frac{d^2(Y(t) * g(t))}{dt^2} = Y(t) * \frac{d^2g(t)}{dt^2}.$$

2.3.5 Multi-scale Curvature Computation

The Gaussian kernel, $g(t)$, is in fact a function of both the arc-length parameter t and the scale parameter σ. By varying the scale σ of a Gaussian kernel, we control the level of the contour noise smoothing of the Gaussian kernel. The planar curves and the curvature of all the points along the curves can be described at different scales as follows,

$$\Gamma(t, \sigma) = (X(t, \sigma), Y(t, \sigma)),$$

and

$$K(t, \sigma) = \frac{\dot{X}(t, \sigma)\, \ddot{Y}(t, \sigma) - \ddot{X}(t, \sigma)\, \dot{Y}(t, \sigma)}{(\dot{X}(t, \sigma)^2 + \dot{Y}(t, \sigma)^2)^{1.5}},$$

where

$$\dot{X}(t, \sigma) \approx \frac{\partial(X(t, \sigma) * g(t, \sigma))}{\partial t} = X(t, \sigma) * \frac{\partial g(t, \sigma)}{\partial t},$$

$$\ddot{X}(t, \sigma) \approx \frac{\partial^2(X(t, \sigma) * g(t, \sigma))}{\partial t^2} = X(t, \sigma) * \frac{\partial^2 g(t, \sigma)}{\partial t^2},$$

$$\dot{Y}(t, \sigma) \approx \frac{\partial(Y(t, \sigma) * g(t, \sigma))}{\partial t} = Y(t, \sigma) * \frac{\partial g(t, \sigma)}{\partial t},$$

$$\ddot{Y}(t, \sigma) \approx \frac{\partial^2(Y(t, \sigma) * g(t, \sigma))}{\partial t^2} = Y(t, \sigma) * \frac{\partial^2 g(t, \sigma)}{\partial t^2}.$$

Theoretically, the scale parameter σ can be chosen to take on any positive value

in a continuum range. However, in practice, we tend to choose σ to be in the form 2^j. One advantage of choosing σ to be in a power of 2 is that when the contour is convolved with a Gaussian kernel with $\sigma = 2$, the number of points that need to represent the contour can be reduced by half. This observation implies if the contour at scale 1 is represented using N points, a sufficient representation of the contour at scale 2^j only requires $\frac{N}{2^j}$ points. This process of filtering and downsampling the contour is illustrated using the filterbank diagram in Figure 2-7.

2.3.6 Multi-scale Tracking of Local Maxima of Curvature

After the curvature value of each point along the contour is obtained at different scales, a coarse-to-fine multi-resolution tracking is attempted at this stage. The processing is first done at a coarse resolution, i.e., high scale, where the Gaussian kernel with a large support is convolved with the signal, thus the noise as well as the true signal on the contour are heavily smoothed out. The points that correspond to the local maxima on the smoothed curvature graph are detected to be the initial corner points. However, a single pixel at a resolution 2^j covers 2^j number of pixels at resolution 1, due to the filtering and downsampling process [15]. The corner positions detected at scale 2^j must be tracked and localized at finer scales. By letting the tracking window be of constant size K throughout all scales, the local maxima of the curvature graph is located over the neighborhood of varying sizes with respect to the original scale, 1. Specifically, at scale 2^j, the local maxima is located over a local window of size $2^j K$. This coarse-to-fine tracking allows for most of the searching to be done at a high scale and a minimal amount of information to be processed at lower scales, which speeds up the process. It is apparent that this multi-resolution information processing enables some computational savings to be gained over the conventional tracking/searching method.

2.4 Results

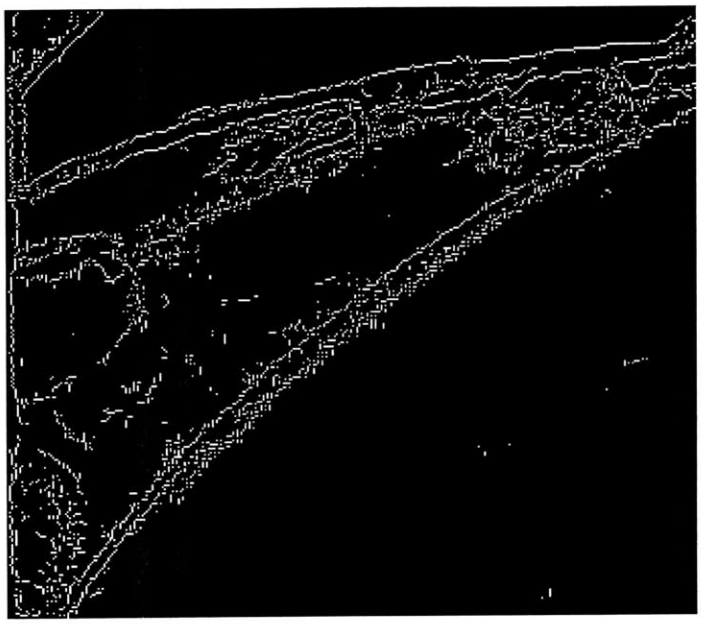
We test our corner detection algorithm on three sets of imagery. The first is a synthetic binary image of a polygon drawn to have 14 corners as shown in Figure 2-8a. The second test image is a real gray-scale image of several polyhedrons shown in 2-17a. To illustrate the differences in the characteristics of the curvature graphs derived from a synthetic image versus a real gray-scale image, we zoomed into one particular polyhedron, a prism, as shown in Figure2-13a. The third set of test images shown in Figure 2-18a is a real color image of a parking lot. This image was captured by a video camera mounted on a moving car. As apparent in the example images shown, real test images are noisier than the synthetic image. We demonstrate the difference in performance of our detector on these three cases.

In the case of a synthetic binary image of a polygon, Figure 2-8b shows that the edge contour detected from the Canny edge detector in step 1 is very well-defined and the contour does not need to be linked or thinned. Since all the corners of the synthetic polygon have very sharp angles, we expect our algorithm to be able to accurately and reliably detect all the corner points. After contour extraction, step 4 of our algorithm parameterizes the contour with respect to the arc-length parameter. In step 5, curvature is computed at varying scales for each point along the contour. The smoothed contours and the absolute curvature graphs are plotted at 4 different scales: when $\sigma = 2$, 4, 8, and 16. Figure 2-9 and Figure 2-10 show the smooth edge contours at different scale parameters. Figure 2-11 and Figure 2-12 plot the absolute curvature of all the points along the contour of the synthetic polygon with respect to the arc-length parameter, again at 4 different scales. As seen in the curvature plots, 14 peaks corresponding to 14 corners of the synthetic polygon can be easily detected at all scales due to the fact that all corner points in the synthetic polygon have sharp angles, i.e., high curvature values. By using the multi-scale tracking/searching scheme in step 6, our algorithm can reliably detect all the corners with great accuracy as shown in Figure 2-16a. False detection does not appear.

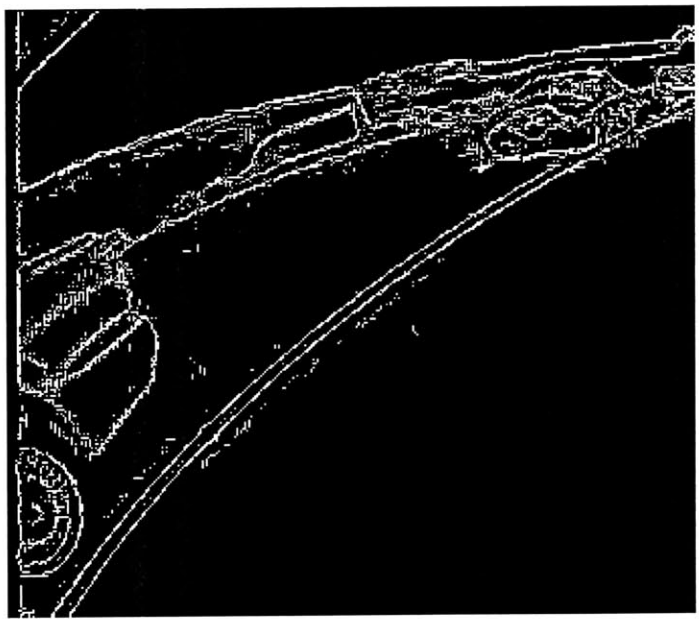
The scenario in the second test image is slightly different due to the noise in

the gray-scale image. First, step 1 applies the Canny edge detector to extract edge contour of the original image of a prism. The binary edge map result in Figure 2-13b illustrates the existence of ripples along the contour, which will result in errors in curvature computation. Linking and thinning edge contours (step 2 and 3) are again not necessary in this case because the detected contour is already thinned and connected. Step 4 parameterizes the contour and curvature is computed at various scales in step 5. The smooth contours of the prism at scale σ = 1, 2, 4, and 8 are shown in Figure 2-13c, d, e, and f, respectively. As mentioned in section 2.3.5 and apparent in Figure 2-13, the sufficient number of points used in representing the contour scale j will be $\frac{1}{2^j}$ of the number of points at scale σ =1. The curvature plot in Figure 2-14a illustrates the appearance of many false local maxima influenced by the ripple noise on the contour at scale $\sigma = 1$. At higher scales, however, a Gaussian kernel with a larger support smoothes out the noise and allows for only the true corners to be detected. As shown in Figure 2-14b and Figure 2-15a, there are 5 peaks on the absolute curvature plot corresponding to the positions of the 5 corners of the prism. Using multi-scale tracking/searching algorithm (step 6), all the 5 corner points are detected and localized accurately as seen in Figure 2-16b. When our corner detection scheme is applied to the entire image that consists of several polyhedrons in Figure 2-17a, the result is shown in Figure 2-17b. Figure 2-17b shows the detection and localization scheme of our algorithm is working satisfactorily.

For the case of a color image of a car in the parking lot in Figure 2-18a, however, the performance of our detection algorithm degrades slightly. The poor quality of the image is caused by the motion blur introduced when the image was captured. The technique of applying the Canny operator to detect edges separately in the R, G, and B components is used. Figure 2-5b shows the edge contour output of step 1 after OR'ing the three binary edge responses from the three channels. The final result of the detected corners are shown in Figure 2-18b. Most corners in the input image are detected. Although the algorithm seems to be missing some corners in the image, all the ones that are detected are well-localized. This example illustrates a robust performance of our coarse-to-fine processing scheme against noise in input images.



(a)



(b)

Figure 2-3: Edge detection scheme that makes use of color information will be able to detect edge contours in the region where gray-scale intensity values are similar but the intensity in each color component is different. In (a), Canny edge detector is applied on the gray-scale intensity value only; (b) shows the binary edge map from applying Canny detector on Red, Green, Blue components separately. The three results of the three color components are OR'ed together to produce a more defined binary edge map.

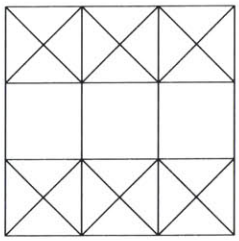


Figure 2-4: Edge linking scheme based on inspection of a local 3x3 neighborhood window. The vertical gap above will be filled when the intensity pattern of the window fits into one of the patterns to link.

(a) (b) (c)

Figure 2-5: (a) Real blurry image of a car in a parking lot; (b) Edge contours detected using the Canny detector ;(c) Contours after linking and thinning

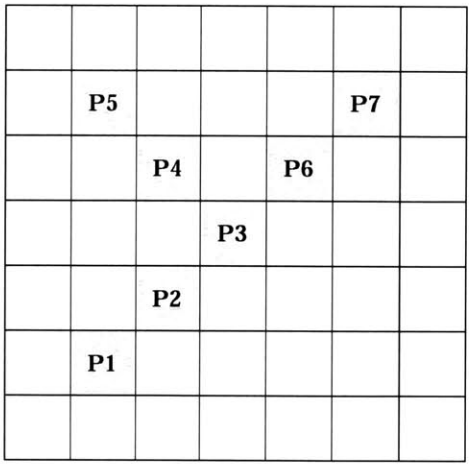


Figure 2-6: Starting parameterizing from an end point P_1, using 8-neighbor connectivity, the next point to label on the contour is P_2, then P_3, etc. When the branch point is reached, the contour is split into 2 separate curves. In this case, the two parameterized curves are $P_1 - P_2 - P_3 - P_4 - P_5$ and $P_1 - P_2 - P_3 - P_6 - P_7$,

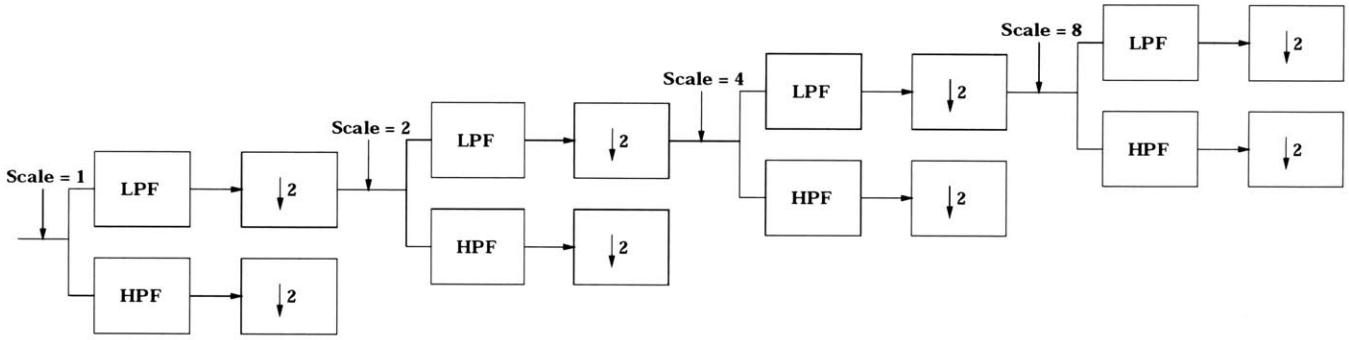
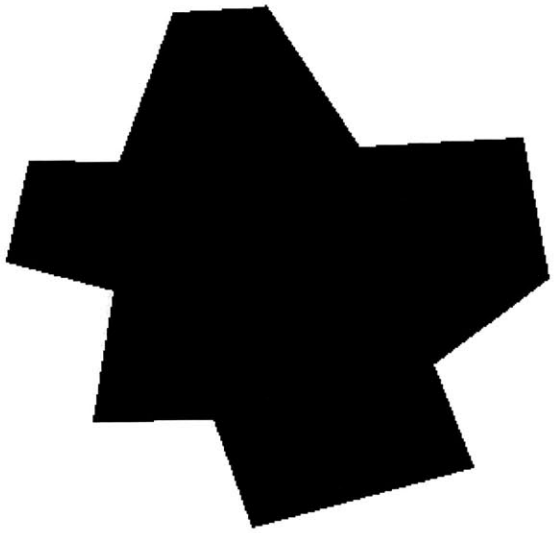
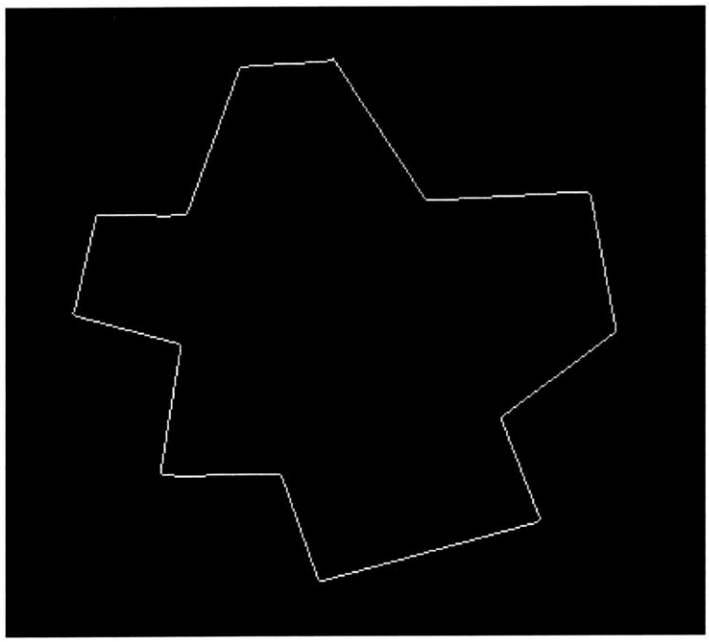


Figure 2-7: Filterbank diagram describing the process of filtering and downsampling. At scale 2^j, the amount of information to be processed is 2^{-j} times that at scale 1.

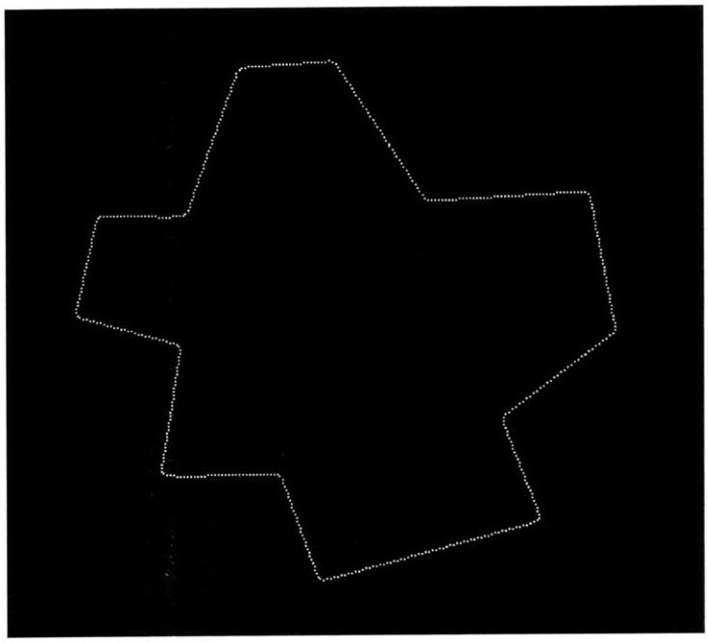


(a)

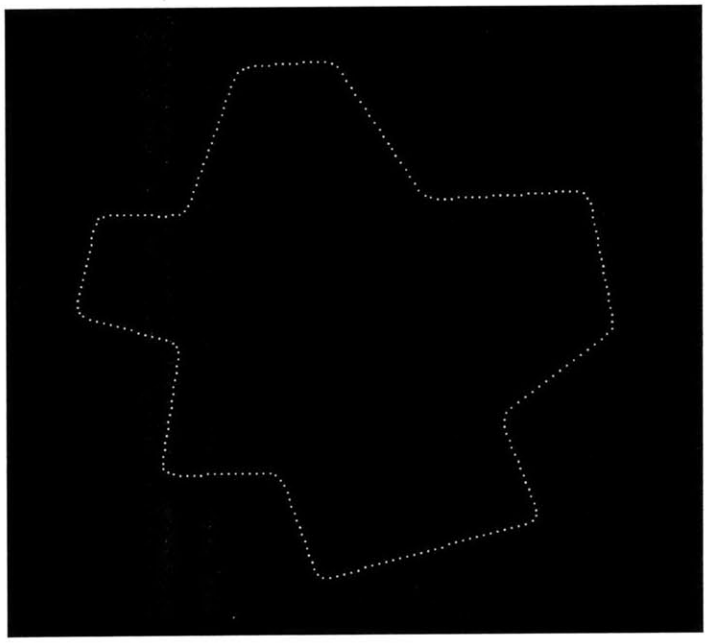


(b)

Figure 2-8: (a) Synthetic binary image of a polygon; (b) Binary edge map output from Canny Edge detector

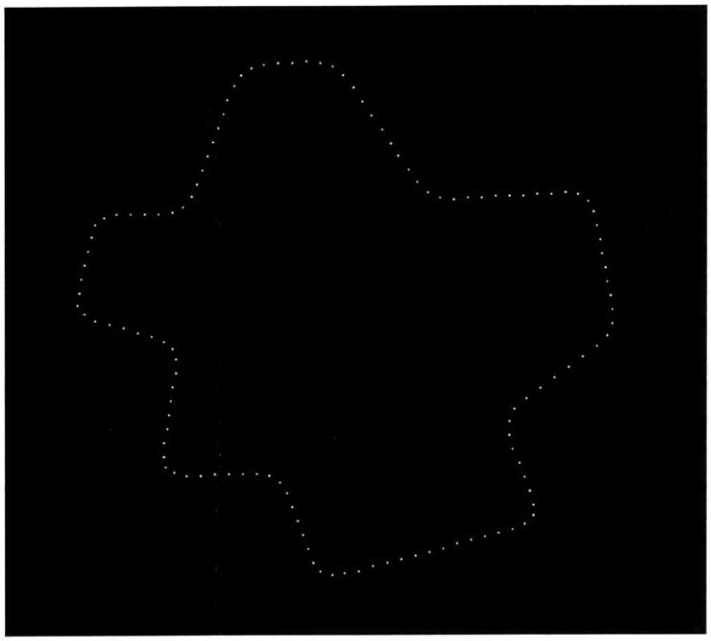


(a)

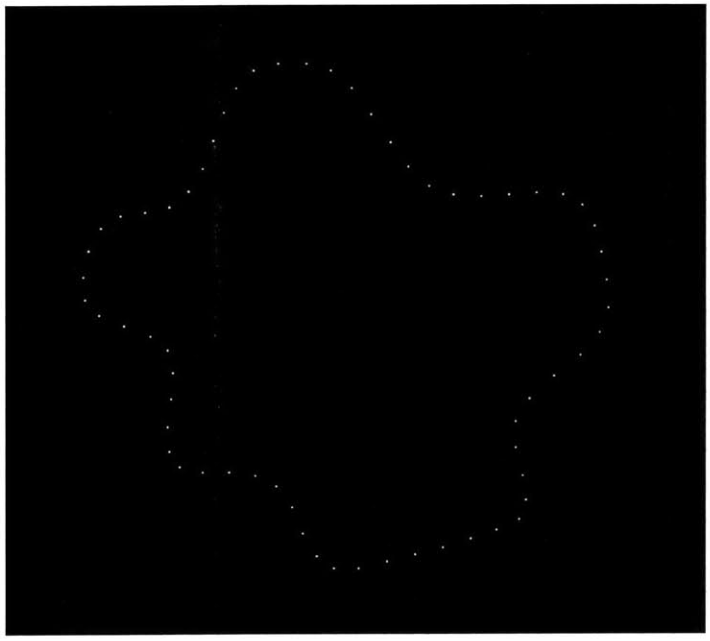


(b)

Figure 2-9: (a) Smooth contour when $\sigma = 2$; (b) Smooth contour when $\sigma = 4$

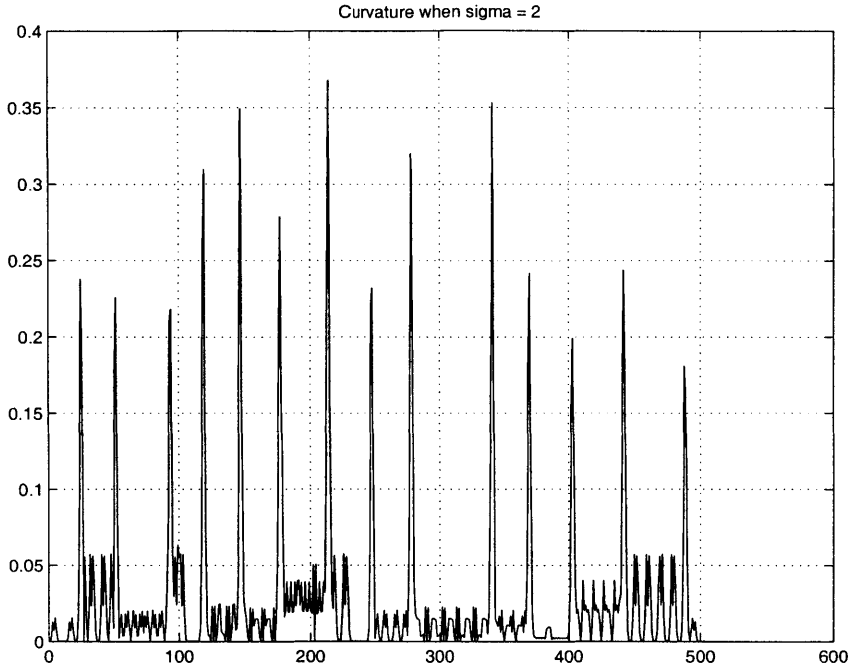


(a)

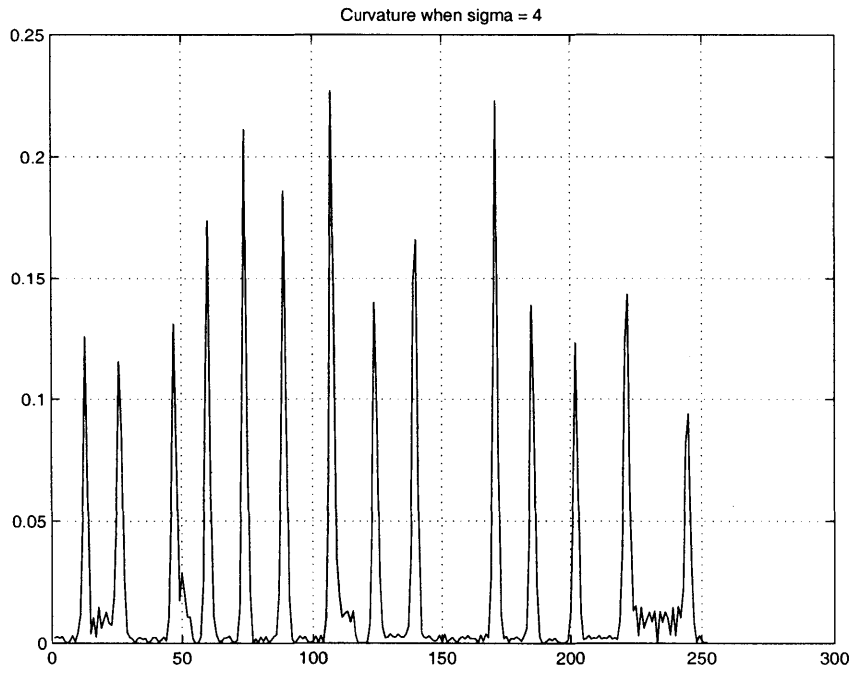


(b)

Figure 2-10: (a) Smooth contour when $\sigma = 8$; (b) Smooth contour when $\sigma = 16$

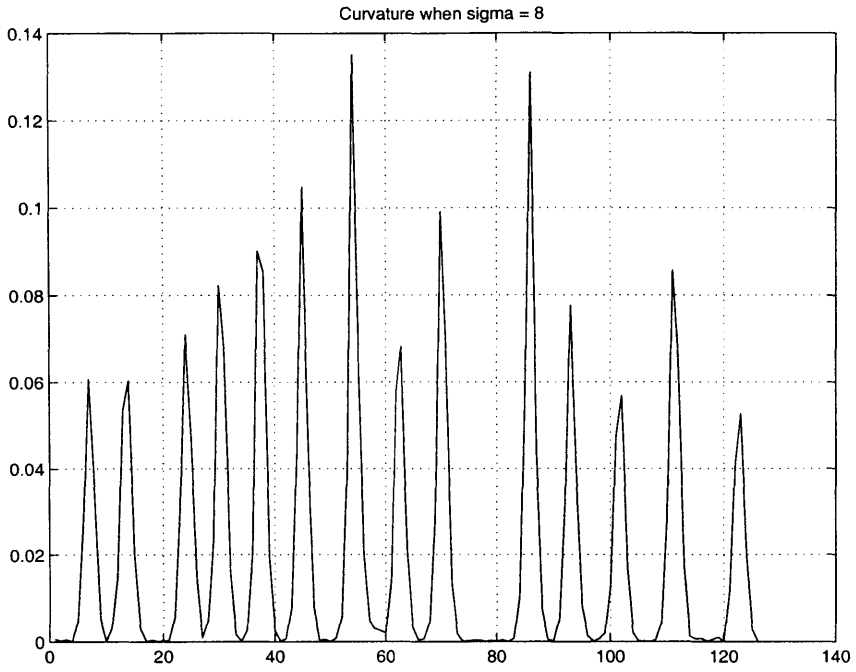


(a)

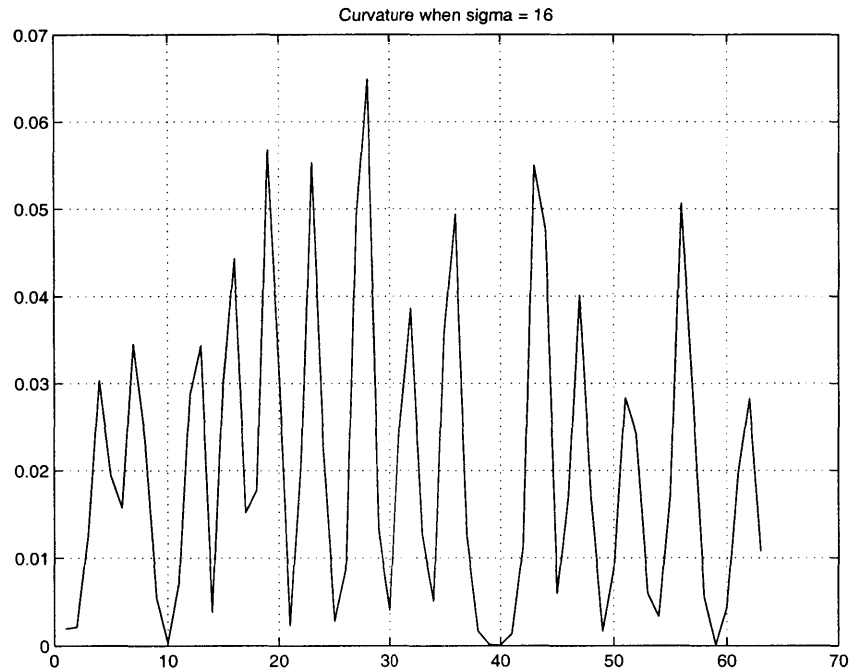


(b)

Figure 2-11: (a) Absolute curvature with respect to arc-length parameter, t, when $\sigma = 2$; (b) Absolute curvature with respect to arc-length parameter, t, when $\sigma = 4$



(a)



(b)

Figure 2-12: (a) Absolute curvature with respect to arc-length parameter, t, when σ = 8; (b) Absolute curvature with respect to arc-length parameter, t, when $\sigma = 16$

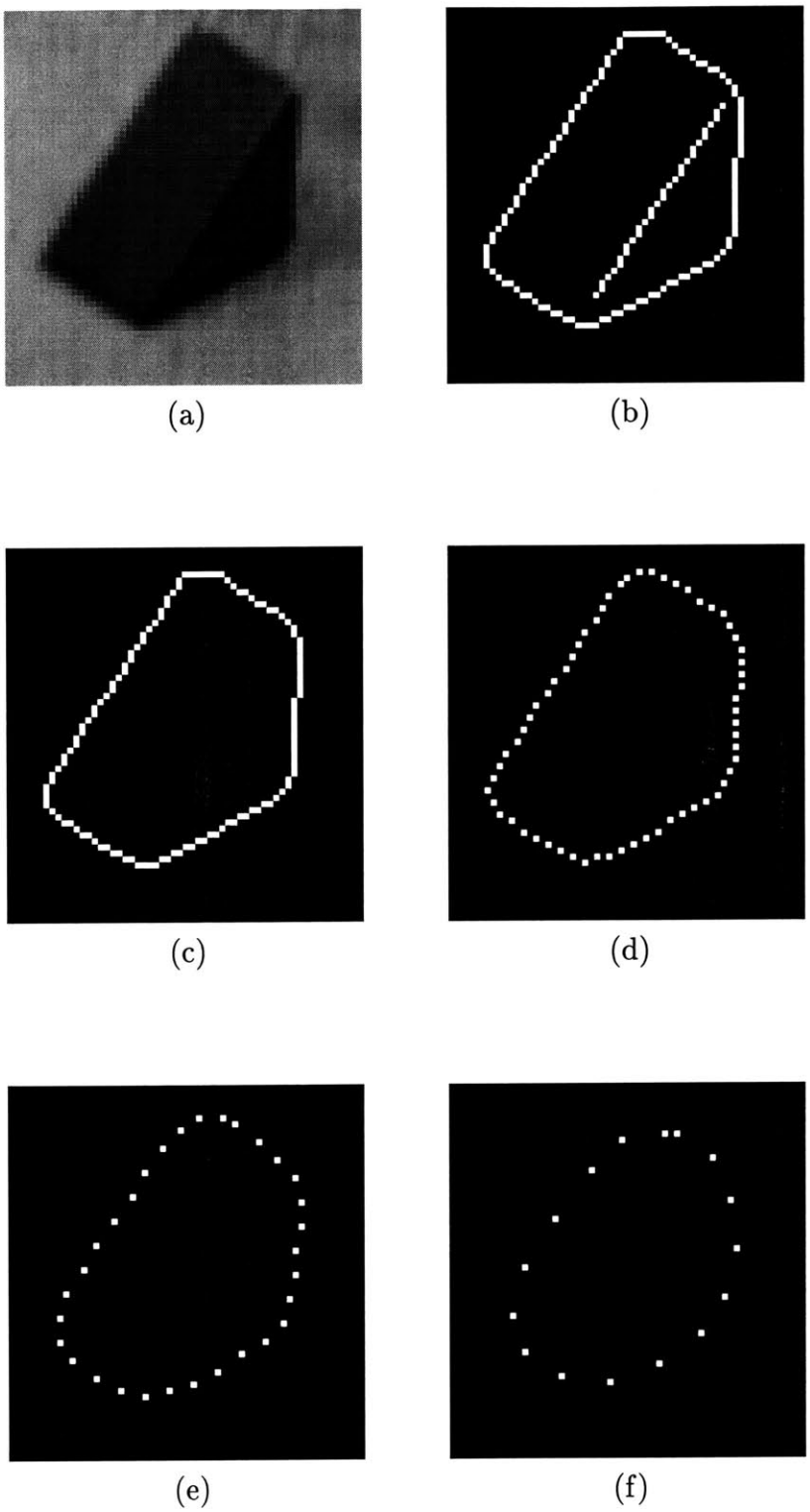
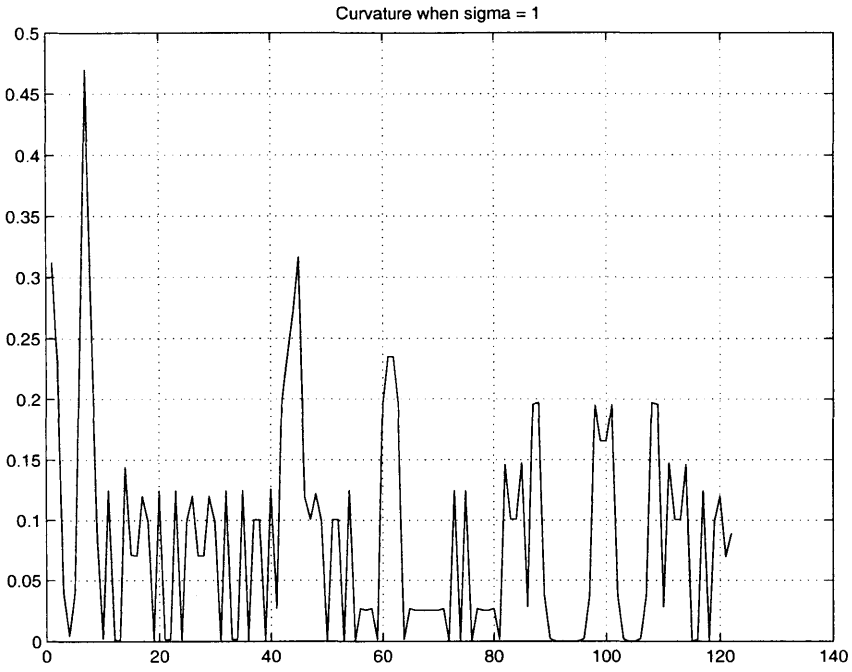
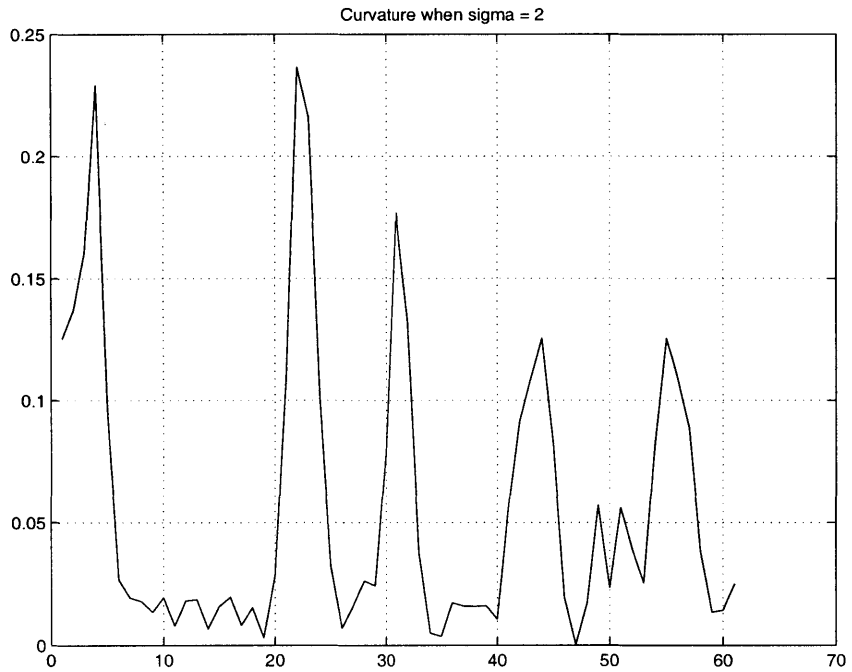


Figure 2-13: (a) Real gray-scale image of a polyhedron; (b) Binary edge map output from Canny Edge detector; (c) Smooth contour when $\sigma = 1$; (d) Smooth contour when $\sigma = 2$; (e) Smooth contour when $\sigma = 4$; (f) Smooth contour when $\sigma = 8$

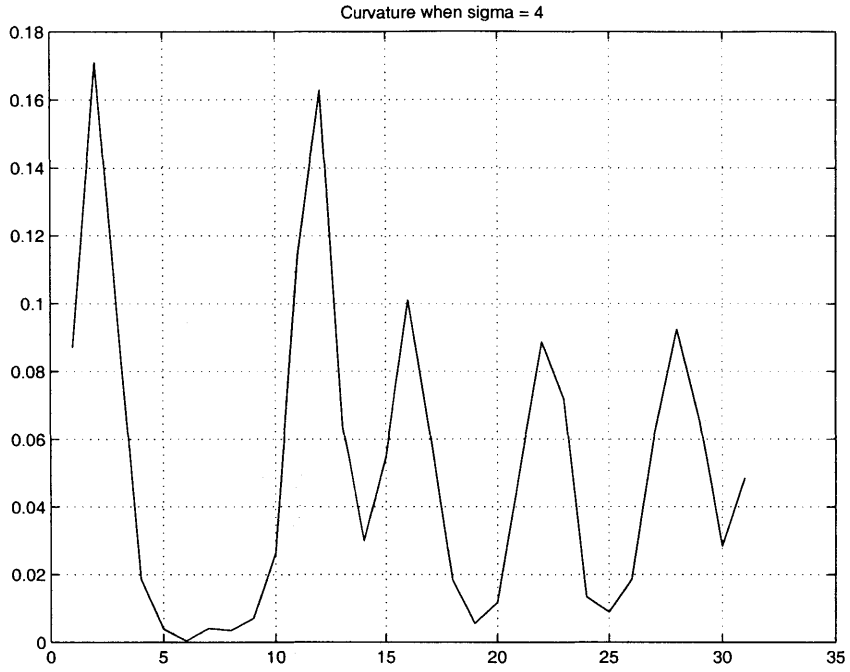


(a)

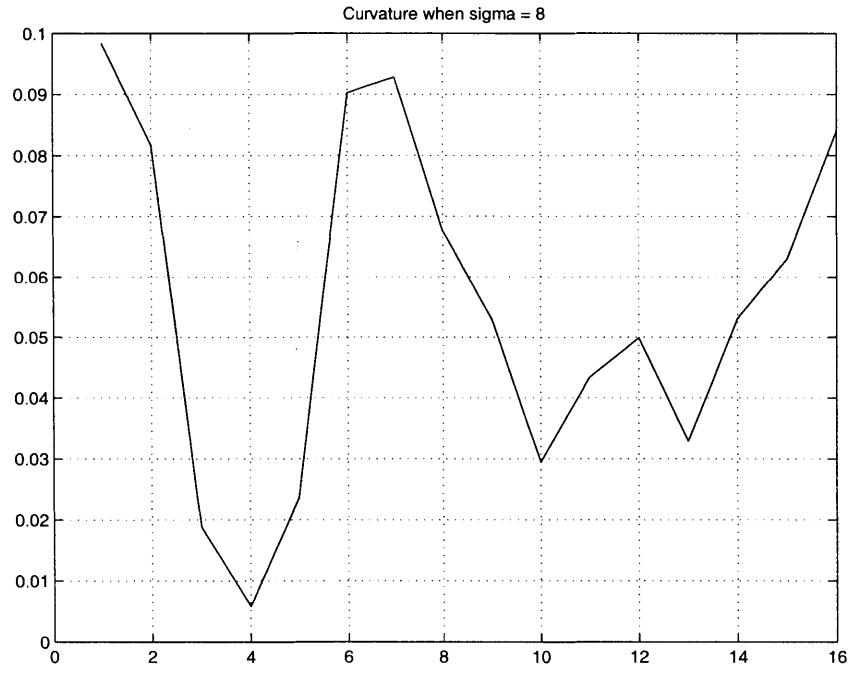


(b)

Figure 2-14: (a) Absolute curvature with respect to arc-length parameter, t, when $\sigma = 1$; (b) Absolute curvature with respect to arc-length parameter, t, when $\sigma = 2$

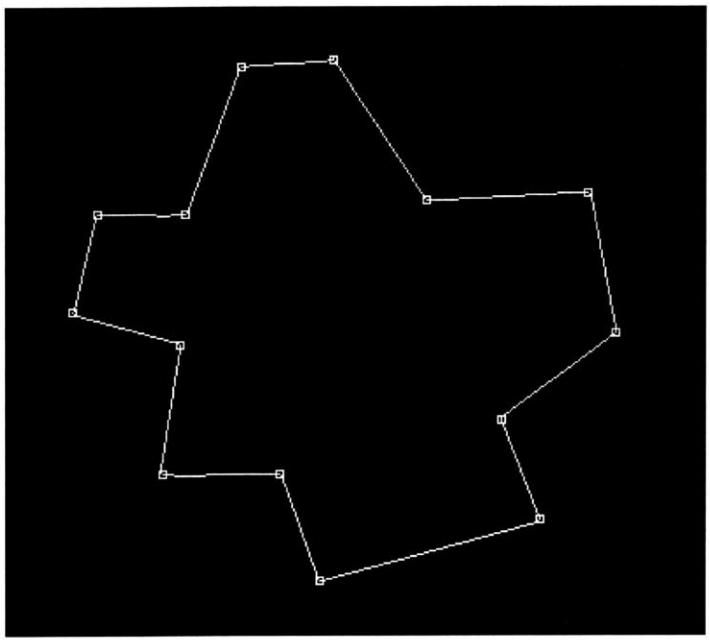


(a)

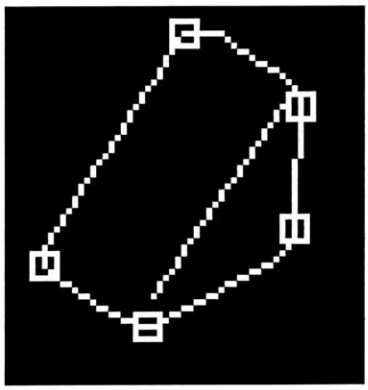


(b)

Figure 2-15: (a) Absolute curvature with respect to arc-length parameter, t, when $\sigma = 4$; (b) Absolute curvature with respect to arc-length parameter, t, when $\sigma = 8$

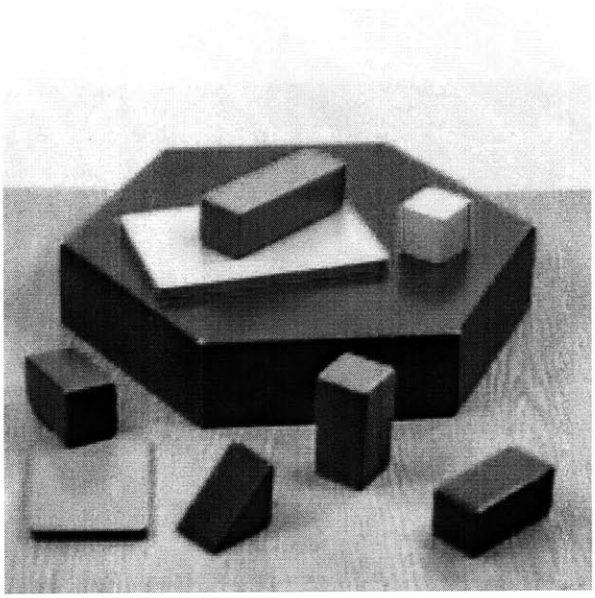


(a)

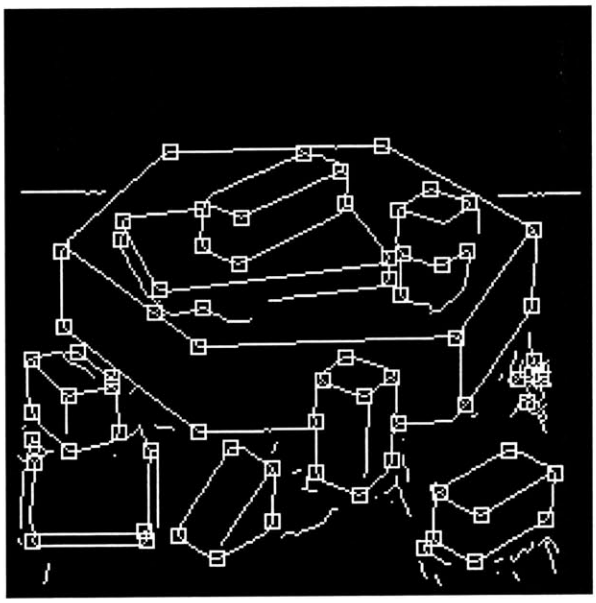


(b)

Figure 2-16: (a) Detected corners in a synthetic image of a polygon; (b) Detected corners in a real image of a polyhedron



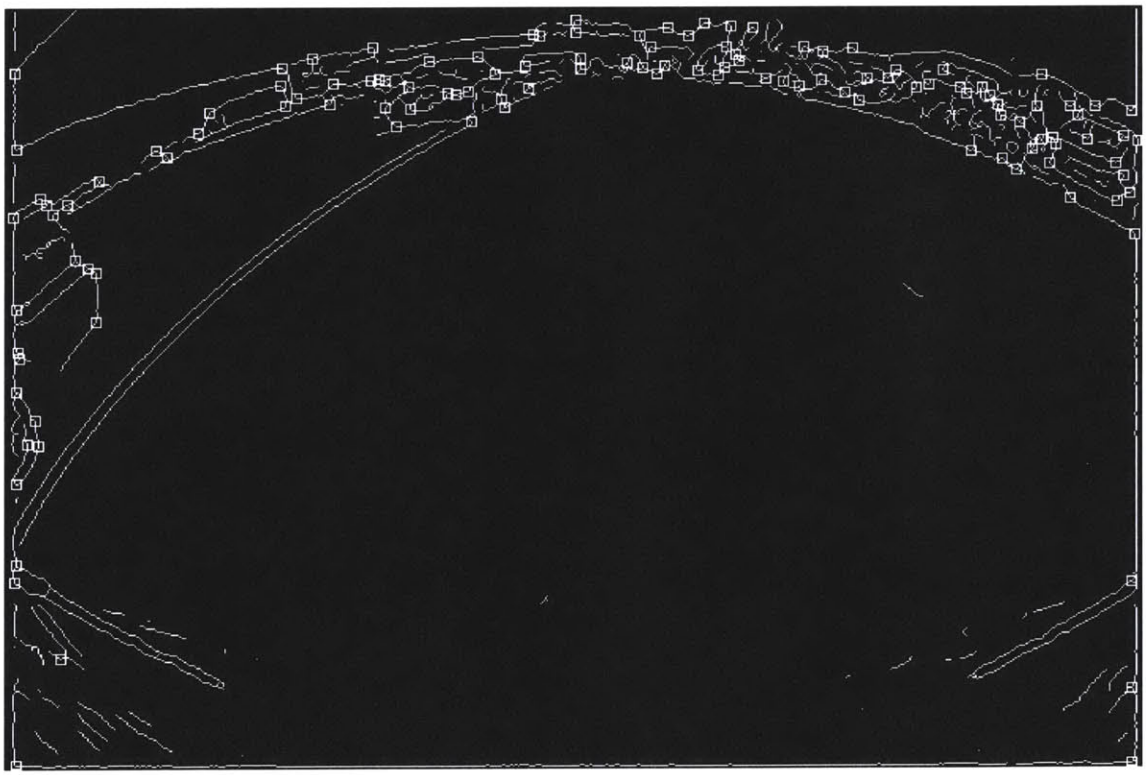
(a)



(b)

Figure 2-17: (a) Real image of several polyhedrons; (b) Detected corners

(a)



(b)

Figure 2-18: (a) A full image of a parking lot; (b) Detected corners

Chapter 3

Correspondence Matching and Depth Calculation

After feature points in input images have been extracted, the subsequent step concerns how to correctly establish correspondence among the feature points. Several algorithms have been proposed over the years in an attempt to solve this problem. This chapter first reviews several existing methods to reduce correspondence mismatching. Next, a convenient algorithm that performs well in coordination with our particular corner detector is proposed. Then, a depth estimation algorithm that makes use of the motion parameters from the inertial sensors and the distance information of the wheel encoders is presented in detail. Several issues regarding camera calibration and lens geometrical distortion correction from the use of wide-angle lens are also discussed.

3.1 Correspondence Matching

In many depth recovery systems, correspondence matching plays a central role in determining the accuracy and reliability of the estimated depth. Many binocular stereo as well as monocular stereo systems with known camera motions use epipolar constraints to restrict the search for corresponding points to be along the matching epipolar lines. Nevertheless, correspondence mismatching still frequently occurs.

The correspondence problem in binocular stereo and monocular stereo systems are slightly different, thus the two cases are presented separately in this chapter as different approaches are usually taken to address them. The next section presents a comparative overview of the correspondence problem in both binocular stereo and monocular stereo.

3.1.1 Correspondence Matching in Binocular Stereo

The key problem in stereo processing is how to accurately locate the corresponding points in a stereo pair of images. In general, the possibility of false matches increases with the search range for correspondence. Recall that in a binocular stereo system with parallel optical axes, the relationship between the difference in pixel locations of the two corresponding points, i.e., disparity d, and the distance between the two camera stations, i.e,, baseline B, is as follows:

$$d = \frac{B \cdot f}{Z}$$

From the above equation, d is directly proportional to B. With a wide baseline, the disparity between corresponding points increases, hence, the search range, becomes large. Therefore, binocular stereo systems with the two camera stations located further apart face the most severe problem in matching. Decreasing the baseline reduces the search range, i.e., disparity, and thus helps simplify the matching process. However, the small disparity, in turn, causes more uncertainty in estimating depth. This relationship can be proven as follows. Let Z be the actual distance of an object to be estimated. Let b and B be the smaller and larger baselines of the two binocular stereo systems. Let d and D be the disparities of the corresponding points in the smaller and larger baseline systems accordingly. We have the following relationships:

$$Z = \frac{b \cdot f}{d} \quad and \quad Z = \frac{B \cdot f}{D}$$

Given a fixed measurement error δd in measuring disparities d and D, we can express the depth estimation error δz and δZ in the two binocular stereo systems as in the following.

$$\delta z = \frac{-Z\delta d}{d + \delta d} \quad and \quad \delta Z = \frac{-Z\delta d}{D + \delta d}.$$

That is, a larger disparity D results in a smaller error in approximating depth. In general, there is a tradeoff between accuracy in matching and precision in estimating depth in binocular stereo processing.

One of the most common techniques in dealing with the correspondence matching problem is to use a coarse-to-fine algorithm to eliminate the possibility of false matches. Matching is first done at a low resolution where the initial results are used to limit and guide the search in a subsequent higher resolution matching. This coarse-to-fine approach has been proven to help reduce correspondence mismatches in many systems [9], [2]. In many cases, however, ambiguity in matching is not removed in a coarse resolution matching, especially when input images consist of scenes with repeated features, e.g., a scene of a picket fence or a scene of a building with similar windows [24]. Establishing stereo correspondence in this case becomes a very difficult and challenging task.

3.1.2 Correspondence Matching in Monocular Stereo

In monocular stereo systems, on the other hand, the correspondence matching problem is less severe. This difference is due in part to the arbitrary baseline length between the camera stations that the two images are taken. To reduce the possibility of false matching in monocular stereo, correspondence matching is done between consecutive image frames in a densely sampled image sequence, e.g., at a video rate of 30 frames/sec. In estimating depth, however, a longer baseline is preferred to attain a higher precision of depth calculation. This process of selecting a longer baseline corresponds to picking two frames that are far apart in the image sequence.

In our parking assistant system, the input to the depth recovery system is a video sequence of images taken from a video camera mounted on a car that is attempting

to park. Since the traveling speed of the car inside the parking lot can safely be assumed to be small, with the video capturing rate of 30 frames/sec, i.e., an approximate sampling period of 33.33 msec/frame, feature points between consecutive image frames can only drift a few pixels away from their original positions. Let I_1 and I_2 be two consecutive frames in the sequence. After all the feature points, i.e., corner points, in I_1 and I_2 are extracted, to find the corresponding points in I_2 for all the corner points detected in I_1, a window of size W in I_2 around each corner location is searched. If only one corner exists in the search window, that corner point is declared a match. However, when more than one corner points are found in the searching window, the characteristics of the corner points are used to aid in the matching process. Specifically, the best match for point p in I_1 is the corner point inside the WxW neighborhood of p in I_2 with an absolute value of curvature closest to that of point p. This algorithm allows for a very robust correspondence matching between 2 consecutive frames in an image sequence.

Correspondence Tracking in Monocular Stereo

After the matching is done between consecutive image frames in the sequence, corresponding points are then tracked from frame to frame through the entire sequence. Our implementation keeps an internal table to store and track the locations of the moving feature points. However, many problems arise in practice when feature points disappear in some part of the sequence due to the use of a non-ideal feature detection scheme. If correspondence matching is only performed between consecutive image frames, the feature points that are missing from one or two frames in the sequence will disappear from the table. Later on in the sequence when the feature points reappear, new entries which in fact correspond to the existing feature points will then be entered as new feature points in the table.

To make our feature point tracking algorithm more robust against the problem of missing feature points and false correspondence, correspondence matching is done at several stages. The following example is used to illustrate the algorithm. Let $I_1, I_2, I_3, I_4, I_5, I_6, ..., I_k, I_{k+1}, ...$ be the image frames in a video sequence. Correspon-

dence matching is done between I_1 and I_2, I_1 and I_3, I_1 and I_4, and I_1 and I_5, using the search windows of size W, $W+1$, $W+2$, and $W+3$ respectively. In the next stage, correspondence matching is performed between I_2 and I_3, I_2 and I_4, I_2 and I_5, I_2 and I_6 with the search windows of size W, $W+1$, $W+2$, and $W+3$. With a larger search window, there is a higher chance of correspondence mismatching. Therefore the matching results obtained in the later stage when the search window size is smaller tend to be more precise and are then used to update the table accordingly. If a feature point is missing in one or two frames in the sequence (suppose a feature point disappears in I_3), the feature point will not disappear from the table entirely due to the matching results between the prior frame (I_2) and the later frame (I_4). This correspondence tracking method has been proven to work robustly against missing feature points and correspondence mismatching.

The next section discusses how to determine depth after correct correspondence has been matched and tracked through the entire image sequence.

3.2 Depth Estimation

After corresponding points have been matched and tracked through the image sequence, depth can be calculated using the disparities between the corresponding points. However, before proceeding to compute depth, the positions of the corresponding points must be corrected to account for the distortion introduced by the use of non-ideal optical systems. In practice, no real optical systems function exactly like an ideal pin-hole camera. Nevertheless, in general, a simple pin-hole camera model serves as a good approximation for many systems that use telephoto lenses. For wide-angle lens systems, distortion becomes very noticeable. Particularly, in the case of a depth recovery system where the values of disparities measured are used to compute depth in the subsequent stage, a small error in the corresponding point locations can lead to a large-scale error in depth estimation. Therefore, before the disparity values can be used to calculate depth, distortion in the positions of the corresponding points must be rectified.

3.2.1 Lens Geometrical Distortion Correction

In general, real optical systems suffer from many types of geometrical distortions. In optical systems that have spherical surfaces, a distortion occurs in the radial direction. There are two types of radial distortion–pin-cushion and barrel distortion. A pin-cushion distortion describes a distortion that occurs with the use of a maximum telephoto lens. A point P in the scene is imaged farther away from the center of the image than its true position. As illustrated on the right diagram in Figure 3-1, Y, denoting the distorted projection of P is located farther away from the center of the image than Y_o, the predicted projection using the pin-hole camera model. For the case where maximum wide-angle lenses are used, a barrel distortion occurs where point P in the environment is imaged closer to the center of the image than its true location as predicted by a pin-hole camera model. The left diagram in Figure 3-1 illustrates a barrel distortion. The displacement from the predicted location, $|Y_o - Y|$, is found to increase with the distance from the center of the image, hence the name radial distortion.

Another type of distortion worth mentioning in this section is tangential distortion, which occurs in the direction perpendicular to the vector from the center of the image. In images obtained from many electro-optical systems, besides the larger radial distortions compared to the optical systems made of glass, some degrees of tangential distortions are also visible. Our parking assistant system uses a video capturing device with a wide-angle glass lens to cover a wide view of the parking lot environment. Thus, the input images to our system only suffer a barrel distortion. In this section, we only concern ourselves with the problem of compensating for such a radial distortion.

A radial distortion causes a pixel displacement that can be modeled using the following relationship:

$$\delta x = x(\kappa_1 r^2 + \kappa_2 r^4 + ...)$$

$$\delta y = y(\kappa_1 r^2 + \kappa_2 r^4 + ...)$$

where x and y are the distorted distance measured from the center of the image in

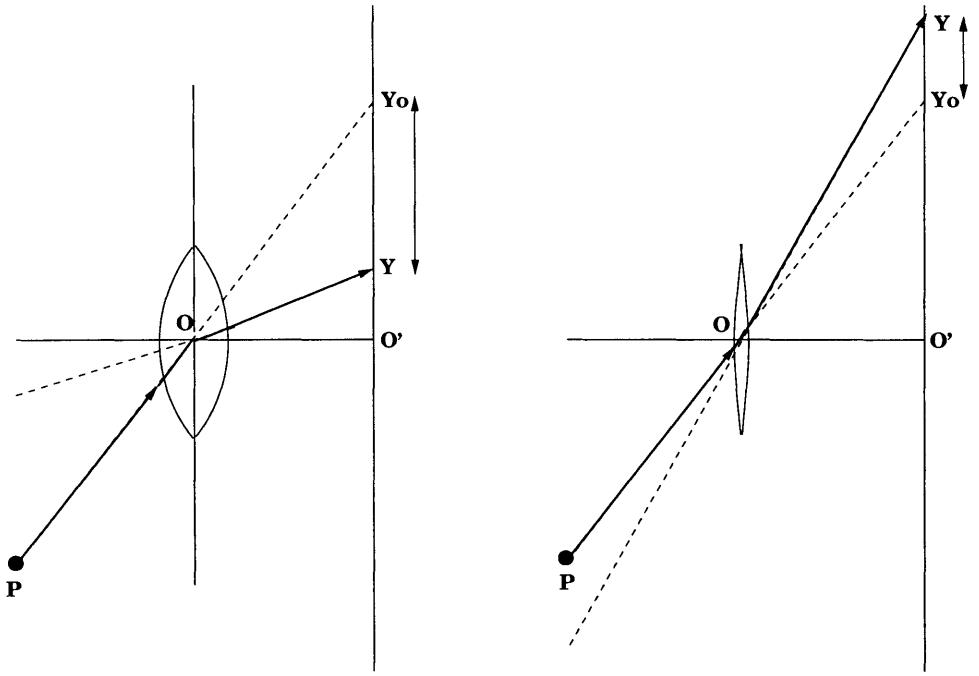


Figure 3-1: Two types of radial distortion. The diagram on the left displays a barrel distortion where the projection of point P in the scene is closer to the center of the image, O', than its actual position as predicted by the pin-hole camera model. On the right is a pin-cushion distortion where the projection of P is farther away from O' than the predicted position.

the x- and y-direction respectively; r is the distance measured from the center of the image and is calculated using $\sqrt{(x^2+y^2)}$. It is observed that only even powers of the distance r from the principle point appear in the equations. In general, the displacement, δx and δy, can be well-approximated using only the first two terms of the sum. That is,

$$\delta x \approx x(\kappa_1 r^2 + \kappa_2 r^4)$$

$$\delta y \approx y(\kappa_1 r^2 + \kappa_2 r^4).$$

The problem now becomes how to obtain the parameters κ_1 and κ_2 in the above equations. If an image contains objects with a known geometry, e.g., a straight line describing the contour of a building or a lane marker on the street, the process of self-calibration can be done. First, an equation of a predicted straight line, e.g., the line describing a lane marker, is obtained. Then, the values of δx and δy at several

locations are estimated from displacement of pixel locations from the predicted line. A least-square fitting technique is used to solve the above system of equations to obtain an approximation for κ_1 and κ_2. For the case when the geometry of objects in the scene is not known, a separate calibration must be done by imaging targets with known geometry, e.g., a chess-board patterned object. The same algorithm as the above is applied to attain the approximation of κ_1 and κ_2.

In general, this radial distortion compensation step can be applied to the input images prior to the step to detect feature points. However, the drawback to compensating for radial distortion before detecting feature points is that a point might now be imaged onto a space between pixel locations. Figure 3-2 illustrates this problem. Let P_1', P_2', P_3', and P_4' be the new calculated positions of P_1, P_2, P, and P_4 after barrel distortion compensation. Since the new positions fall in between pixel locations, the new grid structure of the new image is no longer rectangular. In order to stick with the original rectangular grid, pixel intensity values on the original grid must be recalculated. In Figure 3-2, the new intensity value at point P is obtained from interpolating its four neighboring values P_1', P_2', P_3', and P_4'. This interpolation process corresponds to low-pass filtering the image, which blurs the high frequency area, i.e., edges, of the image. The performance of an edge detection scheme in the subsequent feature extraction step therefore will be poor. Figure 3-3a and Figure 3-3b compare the quality of the output image after distortion correction to the original input image before compensating for barrel distortion.

3.2.2 Computing Depth

After the correct positions of the corresponding points have been determined, depth can be calculated using the disparity between the corresponding points. Similar to the case of binocular stereo, in monocular stereo settings larger disparities yield more accurate depth estimation. Therefore, after corresponding points are tracked through the entire image sequence, $I_0, I_1, I_2, ..., I_k, I_{k+1}, ...$, instead of using the disparity of corresponding points between consecutive frames, the disparity in the positions of the corresponding points in I_1 and I_k are used to compute depth. Through investigation,

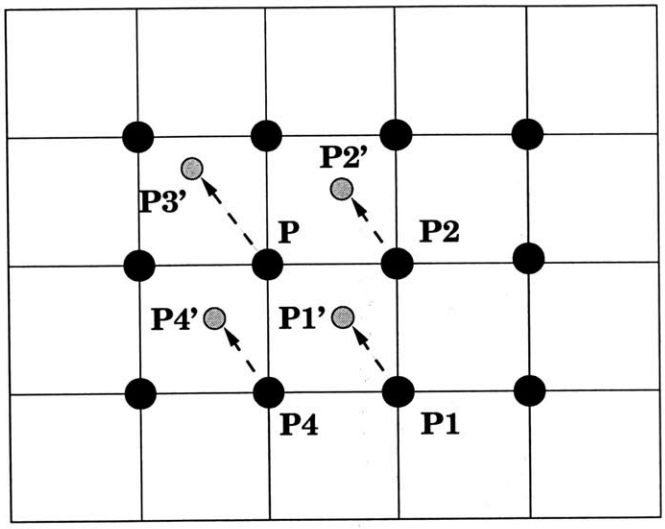


Figure 3-2: After compensating for radial distortion, the new locations P_1', P_2', P_3', P_4' lie in between the pixel locations. The new intensity value at point P is obtained from interpolating the intensity values from its neighbor–P_1', P_2', P_3' and P_4'

the k value of 20 is shown to yield good experimental results.

Let the terms *left* and *right* denote the earlier and later frames used in the matching process. A point $p = (X, Y, Z)$ in the environment is described in the *left* and *right* camera coordinates as $p_l = (x_l, y_l, z_l)$ and $p_r = (x_r, y_r, z_r)$ respectively. The transformation between the *left* and *right* camera coordinates can be decomposed into a rotation as described using a rotation matrix, $\mathbf{R}$, and a translation described using a translation vector, $\mathbf{t}$. That is,

$$p_r = \mathbf{R}p_l + \mathbf{t}.$$

In component form, this relationship can be expressed as:

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} + \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} \tag{3.1}$$

Let (x_l', y_l') and (x_r', y_r') be the two corresponding projections of point p on the *left*

and *right* image planes. Using the perspective projection equations,

$$\frac{x'_r}{f} = \frac{x_r}{z_r} \quad and \quad \frac{y'_r}{f} = \frac{y_r}{z_r}, \tag{3.2}$$

$$\frac{x'_l}{f} = \frac{x_l}{z_l} \quad and \quad \frac{y'_l}{f} = \frac{y_l}{z_l}, \tag{3.3}$$

we obtain the following relationship:

$$\left(r_{11}\frac{x'_l}{f} + r_{12}\frac{y'_l}{f} + r_{13}\right) z_l + X_0 = \frac{x'_r}{f} z_r, \tag{3.4}$$

$$\left(r_{21}\frac{x'_l}{f} + r_{22}\frac{y'_l}{f} + r_{23}\right) z_l + Y_0 = \frac{y'_r}{f} z_r, \tag{3.5}$$

$$\left(r_{31}\frac{x'_l}{f} + r_{32}\frac{y'_l}{f} + r_{33}\right) z_l + Z_0 = z_r. \tag{3.6}$$

The camera geometry f is assumed known from a camera calibration process. The motion parameters between two camera coordinates, $r_{11}, r_{12}, r_{13}, ..., r_{32}, r_{33}, X_0, Y_0, Z_0$ are obtained from inertial sensors and wheel encoders as will be described in the next chapter, and x'_l, y'_l, x'_r, y'_r are measured from correspondence matching. The only two unknowns in this system of equations are the distance z_l and z_r of point p from the camera at the two image acquisitions. Solving this system of equations yields the recovery of depths z_l and z_r.

Simplification of Rotation Matrix R and Translation Vector t

Fortunately, in most depth recovery systems for ITS applications, the rotation matrix $\mathbf{R}$ contains less than 9 degrees of freedom. Since the road surface can be assumed flat in most situations, the rotation of the vehicle, i.e., the rotation between the two camera positions, is restricted to be about the axis perpendicular to the road surface plane. By imposing the additional orthonormality constraint, $\mathbf{R}$ can be expressed in terms of one free variable–the rotation angle θ. According to the coordinate axes defined in Figure 3-4, the rotation between the left and right camera coordinates is restricted to be around the y-axis Y_l coming out of the road surface plane. Imposing the orthonormality constraint, a rotation around one axis can be described using

trigonometric functions, i.e., sin, and cos, of the angle of rotation θ. That is, a rotation around Y_l in the counter-clockwise direction is represented using a rotation matrix $\mathbf{R}$ as follows:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Under the assumption of a translation along a flat road surface, y_r in Eq.(3.1) must be the same as y_l. The second row of the rotation matrix $\mathbf{R}$, $[r_{21}, r_{22}, r_{23}]$, is simplified to [0, 1, 0] and the Y_0 component of the translation vector $\mathbf{t}$ must be zero. That is, the rotation matrix $\mathbf{R}$, the translation vector $\mathbf{t}$ and Eq.(3.1) can be further simplified to:

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} + \begin{pmatrix} X_0 \\ 0 \\ Z_0 \end{pmatrix} \tag{3.7}$$

Expressing x_r, y_r, z_r, x_l, y_l, and z_l in terms of x'_r, y'_r, x'_l, y'_l, and f using the perspective projection relationships, Eq.(3.4), Eq.(3.5), and Eq.(3.6) which are used to compute depth can then be simplified to:

$$\left(\cos\theta\frac{x'_l}{f} + \sin\theta\right) z_l + X_0 = \frac{x'_r}{f} z_r, \tag{3.8}$$

$$\frac{y'_l}{f} z_l = \frac{y'_r}{f} z_r, \tag{3.9}$$

$$\left(-\sin\theta\frac{x'_l}{f} + \cos\theta\right) z_l + Z_0 = z_r. \tag{3.10}$$

The only two unknowns in this system of equations are the depths z_l and z_r. Solving for two unknowns in three linear equations, we use the least square approximation method. In the next chapter, we present a method to obtain the motion parameters θ, X_0, and Z_0.

(a)

(b)

Figure 3-3: The new pixel locations after compensating for a radial distortion might fall in between pixel locations. Interpolating from the neighboring intensity values to obtain the new intensity values at the original pixel locations blurs the image especially along the edges. (a) Original image of a parking lot before compensating for barrel distortion; (b) Corrected image with some blurring effect along the edges

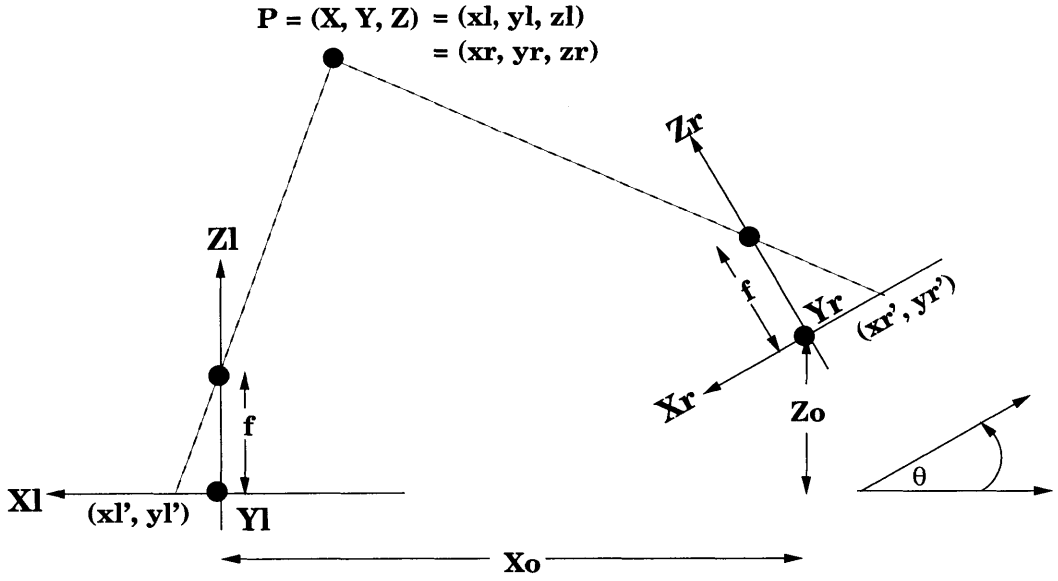


Figure 3-4: Assume a vehicle moving on a flat road surface, the translation vector $\mathbf{t}$ is simplified to $(X_0, 0, Z_0)^T$. The rotation between the two camera stations is restricted to be around the Y_l-axis. Imposing the orthonormality constraint, the rotation matrix $\mathbf{R}$ can be represented as a function of one variable–the rotation angle, θ.

Chapter 4

Motion Parameters and Depth Calculation revisited

In the previous chapter, we derive an algorithm to compute depth from two image frames in a sequence when the rotation between of the camera coordinates the two frames is restricted to be around one axis. We found that the rotation matrix $\mathbf{R}$ can be described as a function of one variable θ, and the translation vector $\mathbf{t}$ has only two nonzero components (X_0 and Z_0). In this chapter, we begin by describing the process of obtaining the transformation parameters θ, X_0, and Z_0. Next, a practical implementation issue concerning the camera orientation with respect to the road surface is discussed and the depth calculation is revisited.

4.1 Obtaining Parameters θ, X_0, and Z_0

This section presents a summary of how the parameters θ, X_0, and Z_0 that describe the rigid body motion between the *left* and *right* camera stations can be obtained. In our parking assistant system, a video camera with wide-angle lens is mounted to overlook the back view of the vehicle as shown in Figure 4-1. Since the camera is mounted to have a fixed orientation with respect to the body of the car, the motion of the camera between the time instances that the *left* and *right* images were acquired should be the same as the motion of the vehicle between the same time period. As

described in the previous chapter, under the assumption that a vehicle travels on a flat road surface, this transformation can be represented as a rotation by θ about an axis perpendicular to the road surface plane followed by a translation $[X_0, 0, Z_0]^T$ parallel to the road plane.

The rotation parameters are obtained using the inertial sensors to measure the angular moment of the steering wheel. We use this information to derive the steering wheel angle, i.e., the angle which the steering wheel has been rotated, which in turn is used to calculate the instantaneous tire inclination angle and the gyration curvature of the moving vehicle. Using a specific conversion table for a 2-wheel vehicle model, the information on tire inclination angle and gyration curvature is used to approximate the rotation angle θ that the body of car has been turned, and hence the rotation between the *left* and the *right* camera coordinates. The translation parameters X_0 and Z_0 are obtained by projecting onto the $x-$ and the $z-$axes the micro-movement distance measured from the wheel encoders attached to the rear wheels of the car.

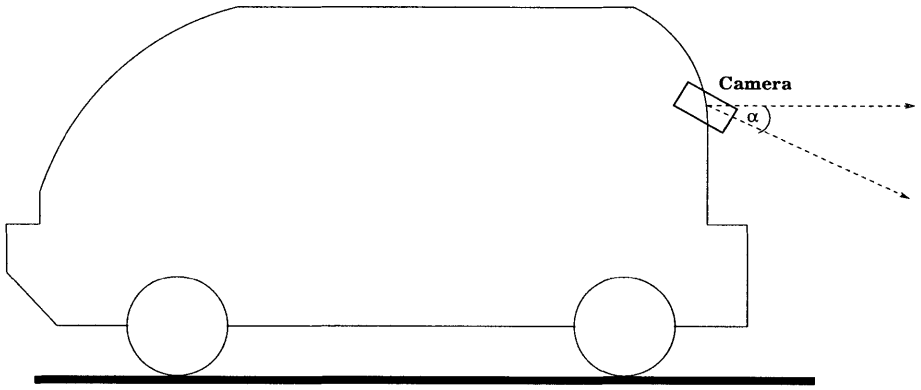


Figure 4-1: A video camera with a wide angle lens is mounted on the rear window to capture the rear view of a car. In practice, depending on the height of the vehicle used in the experiment, the optical axes of the camera will not be parallel to the road surface plane, but will be tilted as seen in this figure to be able to capture the obstacles that are closer to the vehicle.

In practice, however, the measured rotation angle θ that the body of the car has turned is generally not equal to the rotation angle of the camera between the two camera positions. This discrepancy is due in part to the geometry of the vehicle used in the experiment. If a tall vehicle, e.g., a crown van as shown in Figure 4-1, is used to

mount the camera, the optical axis of the camera needs to be tilted down to capture the obstacles located closer to the van, making an appreciable angle α with the road surface plane. In this experiment, the angles that different axes of the camera make with the axes defined by arrows pointing in the direction of the rear and the right of the vehicle are shown in Figure 4-2. In particular, the optical axis of the camera makes a significant angle (40.5^o) with the road surface plane. The assumption used in the previous chapter to derive Eq.(3.8), Eq.(3.9) and Eq.(3.10) no longer holds and the depth determination problem must be revisited.

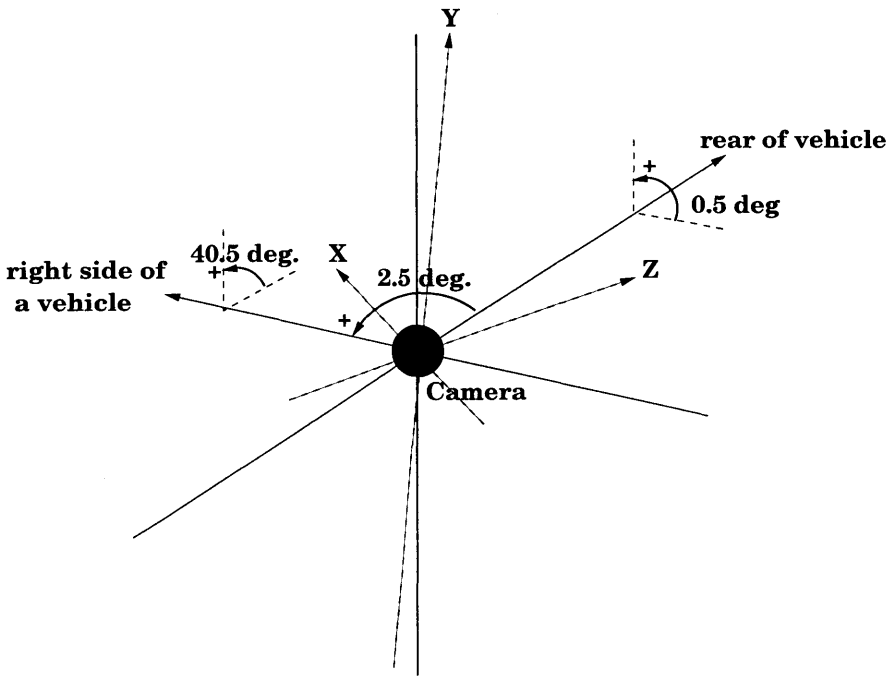


Figure 4-2: The orientation of the camera used to capture images in this experiment is shown above. The $Y-$axis makes a 40.5^o angle with the axis perpendicular to the road surface plane. The $Z-$ and $X-$ axes of the camera make 2.5^o and 0.5^o angles with the line in the direction of rear and the right of the vehicle respectively.

4.2 Depth Calculation Revisited

The coordinate axes are defined as in Figure 3-4. First, recall Eq.(3.7), which relates the corrected *left* and *right* camera coordinates.

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} + \begin{pmatrix} X_0 \\ 0 \\ Z_0 \end{pmatrix} \tag{4.1}$$

Given a fixed angle α that the optical axis of the camera makes with the road surface plane, we can describe the transformation between the tilted right camera coordinate $[x_{rt}, y_{rt}, z_{rt}]^T$ and the corrected right camera coordinate $[x_r, y_r, z_r]^T$ as a rotation by α in the clockwise direction around the $x-$axis.

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x_{rt} \\ y_{rt} \\ z_{rt} \end{pmatrix} \tag{4.2}$$

Similarly, rotating the tilted left camera coordinate $[x_{lt}, y_{lt}, z_{lt}]^T$ clockwise by α, we obtain the corrected left camera coordinate $[x_l, y_l, z_l]^T$.

$$\begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x_{lt} \\ y_{lt} \\ z_{lt} \end{pmatrix} \tag{4.3}$$

Plugging Eq.(4.2) and Eq.(4.3) into Eq.(4.1), we obtain

$$\begin{pmatrix} x_{rt} \\ y_{rt} \\ z_{rt} \end{pmatrix} = \mathbf{R} \begin{pmatrix} x_{lt} \\ y_{lt} \\ z_{lt} \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}^{-1} \begin{pmatrix} X_0 \\ 0 \\ Z_0 \end{pmatrix} \tag{4.4}$$

where

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}^{-1} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}.$$

That is, lumping the product of three rotations by α, θ and -α into one rotation matrix, we obtain a rotation $\mathbf{R}$ and a translation $\mathbf{t}$ between $[x_{rt}, y_{rt}, z_{rt}]^T$ to $[x_{lt}, y_{lt}, z_{lt}]^T$ as follows.

$$\begin{aligned} \mathbf{R} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}^{-1} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \\ &= \begin{pmatrix} \cos\theta & \sin\theta\sin\alpha & \sin\theta\cos\alpha \\ -\sin\theta\sin\alpha & \cos^2\alpha + \cos\theta\sin^2\alpha & \frac{\sin 2\alpha}{2}(\cos\theta - 1) \\ -\sin\theta\cos\alpha & \frac{\sin 2\alpha}{2}(\cos\theta - 1) & \sin^2\alpha + \cos\theta\cos^2\alpha \end{pmatrix} \end{aligned} \tag{4.5}$$

and

$$\begin{aligned} \mathbf{t} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}^{-1} \begin{pmatrix} X_0 \\ 0 \\ Z_0 \end{pmatrix} \qquad (4.6) \\ &= \begin{pmatrix} X_0 \\ Z_0\sin\alpha \\ Z_0\cos\alpha \end{pmatrix}. \qquad (4.7) \end{aligned}$$

When we apply the perspective projection equations to Eq.(4.4),

$$\frac{x'_{rt}}{f} = \frac{x_{rt}}{z_{rt}} \quad and \quad \frac{y'_{rt}}{f} = \frac{y_{rt}}{z_{rt}}, \tag{4.8}$$

and

$$\frac{x'_{lt}}{f} = \frac{x_{lt}}{z_{lt}} \quad and \quad \frac{y'_{lt}}{f} = \frac{y_{lt}}{z_{lt}}, \tag{4.9}$$

we obtain the following relationship:

$$\begin{pmatrix} \frac{x'_{rt}z_{rt}}{f} \\ \frac{y'_{rt}z_{rt}}{f} \\ z_{rt} \end{pmatrix} = \mathbf{R} \begin{pmatrix} \frac{x'_{lt}z_{lt}}{f} \\ \frac{y'_{lt}z_{lt}}{f} \\ z_{lt} \end{pmatrix} + \mathbf{t} \tag{4.10}$$

Again, we have a system of three linear equations with two unknowns. A least square estimation method can be used to solve for z_{rt} and z_{lt}.

4.3 Error Analysis in Depth Computation

How reliable is our estimated depth? Many depth recovery applications have different design specifications that impose constraints on the required depth accuracy. As seen from Eq.(4.10), the estimated depths z_{lt} and z_{rt} depend on several parameters which are obtained from different sources. x'_{lt}, y'_{lt}, x'_{rt}, and y'_{rt} are determined from matching/tracking corresponding points through the image sequence. The effective focal length, f, is estimated from a camera calibration stage. As discussed above, θ is obtained using steering sensors to measure the angular moment of the steering wheel, and the translation distance X_0 and Z_0 along the $x-$ and $z-$axes are determined by the wheel encoders attached to vehicle rear wheels. Starting from a given desired accuracy of depth, how accurate must each parameter that z_{lt} and z_{rt} depend on be? In this section, we attempt to derive the relationship between individual measurement errors of different parameters and the final errors in the depth estimation results. Specifically, we would like to see how errors in the measurements of X_0 and Z_0 from the wheel encoders and θ from the steering sensors contribute to the final depth calculation errors.

In the first case, assume no errors occur in the measurements of the rotation angle θ and the translation distance along the $z-$direction. Let the measured value of X_0 be X_{0m} which deviates from its ideal value by δX_0. And let the estimated depths z_{ltm} and z_{rtm} be the ideal depths z_{lt} and z_{rt} plus some errors δz_{lt} and δz_{rt} which are

caused solely by δX_0. That is,

$$\begin{aligned} X_{0m} &= X_0 + \delta X_0, \\ z_{ltm} &= z_{lt} + \delta z_{lt}, \\ z_{rtm} &= z_{rt} + \delta z_{rt}. \end{aligned}$$

We expand Eq.(4.10) in the previous section into three equations as follows:

$$\frac{x'_{rt}}{f} z_{rt} = (r_{11}\frac{x'_{lt}}{f} + r_{12}\frac{y'_{lt}}{f} + r_{13}) z_{lt} + X_0 \tag{4.11}$$

$$\frac{y'_{rt}}{f} z_{rt} = (r_{21}\frac{x'_{lt}}{f} + r_{22}\frac{y'_{lt}}{f} + r_{23}) z_{lt} + Z_0 \sin\alpha \tag{4.12}$$

$$z_{rt} = (r_{31}\frac{x'_{lt}}{f} + r_{32}\frac{y'_{lt}}{f} + r_{33}) z_{lt} + Z_0 \cos\alpha \tag{4.13}$$

where $r_{11}, r_{12}, r_{13}, ..., r_{32}, r_{33}$ denote the corresponding entries in the rotation matrix $\mathbf{R}$ in Eq.(4.5).

To find the relationship between δX_0 and δz_{rt} and δz_{lt}, we take the partial derivative of Eq.(4.11), Eq.(4.12) and Eq.(4.13) with respect to X_0. Rewriting the above equations in a matrix form, we obtain

$$\begin{pmatrix} \frac{x'_{rt}}{f} & -A \\ \frac{y'_{rt}}{f} & -B \\ 1 & -C \end{pmatrix} \begin{pmatrix} \frac{\partial z_{rt}}{\partial X_0} \\ \frac{\partial z_{lt}}{\partial X_0} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \tag{4.14}$$

where

$$\begin{aligned} A &= (r_{11}\frac{x'_{lt}}{f} + r_{12}\frac{y'_{lt}}{f} + r_{13}), \\ B &= (r_{21}\frac{x'_{lt}}{f} + r_{22}\frac{y'_{lt}}{f} + r_{23}), \end{aligned}$$

and

$$C = (r_{31}\frac{x'_{lt}}{f} + r_{32}\frac{y'_{lt}}{f} + r_{33}).$$

We have a system of three linear equations in two unknowns. The least square approximation method can be used to solve for $[\frac{\partial z_{rt}}{\partial X_0}, \frac{\partial z_{lt}}{\partial X_0}]^T$ as follows:

$$\begin{pmatrix} \frac{\partial z_{rt}}{\partial X_0} \\ \frac{\partial z_{lt}}{\partial X_0} \end{pmatrix} = (M^T M)^{-1} M^T \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \tag{4.15}$$

where

$$M = \begin{pmatrix} \frac{x'_{rt}}{f} & -A \\ \frac{y'_{rt}}{f} & -B \\ 1 & -C \end{pmatrix}.$$

Since the matrix M is of size 3x2, the product matrix $(M^T M)^{-1} M^T$ will be of size 2x3. We can denote $(M^T M)^{-1} M^T$ by the following matrix

$$\begin{pmatrix} m & n & o \\ p & q & r \end{pmatrix}_{\text{2X3}}$$

where

$$\begin{aligned}
m &= \frac{1}{\det}\left(\frac{x'_{rt}}{f}(B^2+C^2) - AB\frac{y'_{rt}}{f} - AC\right), \\
n &= \frac{1}{\det}\left(\frac{y'_{rt}}{f}(A^2+C^2) - AB\frac{x'_{rt}}{f} - BC\right), \\
o &= \frac{1}{\det}\left(A^2+B^2 - AC\frac{x'_{rt}}{f} - BC\frac{y'_{rt}}{f}\right), \\
p &= \frac{1}{\det}\left(B\frac{x'_{rt}y'_{rt}}{f^2} + C\frac{x'_{rt}}{f} - A(\frac{y'^2_{rt}}{f^2}+1)\right), \\
q &= \frac{1}{\det}\left(A\frac{x'_{rt}y'_{rt}}{f^2} + C\frac{y'_{rt}}{f} - B(\frac{x'^2_{rt}}{f^2}+1)\right), \\
r &= \frac{1}{\det}\left(A\frac{x'_{rt}}{f} + B\frac{y'_{rt}}{f} - C(\frac{x'^2_{rt}}{f^2}+\frac{y'^2_{rt}}{f^2})\right), \\
\det &= (A^2+B^2+C^2)(\frac{x'^2_{rt}}{f^2}+\frac{y'^2_{rt}}{f^2}+1) - (A\frac{x'_{rt}}{f}+B\frac{y'_{rt}}{f}+C)^2.
\end{aligned}$$

That is, Eq.(4.15) now becomes

$$\begin{pmatrix} \frac{\partial z_{rt}}{\partial X_0} \\ \frac{\partial z_{lt}}{\partial X_0} \end{pmatrix} = \begin{pmatrix} m & n & o \\ p & q & r \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{4.16}$$

Multiply out the right-hand side, we obtain the relationships:

$$\frac{\partial z_{rt}}{\partial X_0} = m,$$

$$\frac{\partial z_{lt}}{\partial X_0} = p.$$

Therefore, an error δX_0 in measuring the camera translation along the $x-$axis will result in errors δz_{rt} and δz_{lt} in the estimation of z_{rt} and z_{lt} as follows:

$$\delta z_{rt} = m\delta X_0, \tag{4.17}$$

$$\delta z_{lt} = p\delta X_0. \tag{4.18}$$

For particular values of Z_0, θ, (x'_{lt}, y'_{lt}) and (x'_{rt}, y'_{rt}), m and p are constant. The plot in Figure 4-3 shows a linear gain factor between the measurement errors in X_0 and the corresponding errors in depth estimation, δz_{lt}. In this plot, Z_0 stays fixed at 0.1830m and θ at 1.28^o. The corresponding gain factor of this plot is 24.8361. That is, an error of 1 meter in the measurement of X_0 will result in an error of 24.8361 meters in estimating z_{lt}.

Now, if instead we assume that the only erroneous measurement is the measured distance of the wheel encoders along the $z-$axis, using a similar technique as in the derivation above, we obtain the following relationships:

$$\begin{pmatrix} \frac{\partial z_{rt}}{\partial Z_0} \\ \frac{\partial z_{lt}}{\partial Z_0} \end{pmatrix} = \begin{pmatrix} m & n & o \\ p & q & r \end{pmatrix} \begin{pmatrix} 0 \\ \sin\alpha \\ \cos\alpha \end{pmatrix}. \tag{4.19}$$

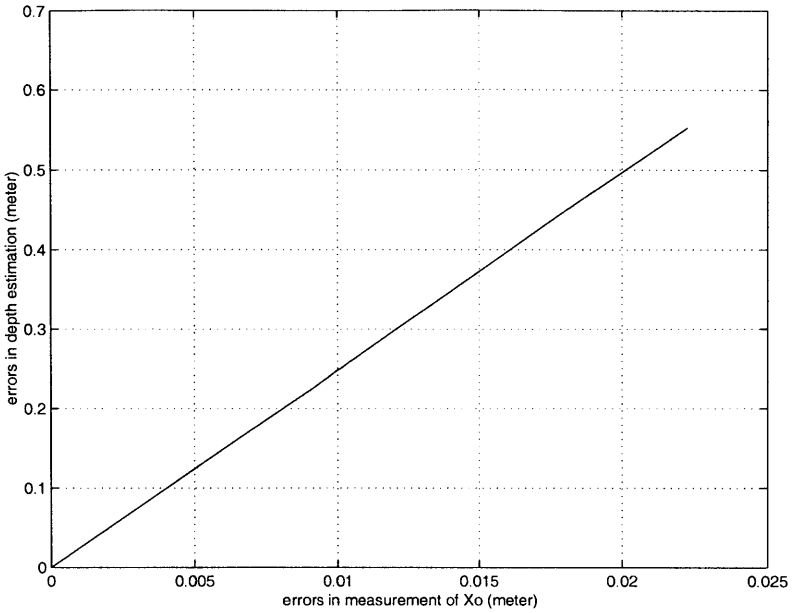


Figure 4-3: The relationship between the measurement errors in X_0 and the corresponding errors in depth estimation results, δz_{lt}, is linear. In this particular plot, Z_0 stays fixed at 0.1830m and θ at 1.28^o. The gain factor of this plot is 24.8361.

Multiply out the right-hand side, we obtain the equations that relate the final errors in depth estimation with the measurement error δZ_0.

$$\frac{\partial z_{rt}}{\partial Z_0} = (n \sin\alpha + o\cos\alpha)$$

$$\frac{\partial z_{lt}}{\partial Z_0} = (q \sin\alpha + r\cos\alpha)$$

That is,

$$\delta z_{rt} = (n \sin\alpha + o\cos\alpha)\delta Z_0, \tag{4.20}$$

$$\delta z_{lt} = (q \sin\alpha + r\cos\alpha)\delta Z_0. \tag{4.21}$$

For particular values of X_0, θ, (x'_{lt}, y'_{lt}) and (x'_{rt}, y'_{rt}), n, o, q and r are constant. Figure 4-4 shows a linear gain factor between the measurement errors in Z_0 and the corresponding errors in depth estimation, δz_{lt}. In this plot, X_0 is fixed at 0.089m and θ at 1.28^o. The corresponding gain factor of this plot is 5.6966. That is, an error of

1 meter in measuring Z_0 will result in an error of 5.6966 meter in estimating z_{lt}.

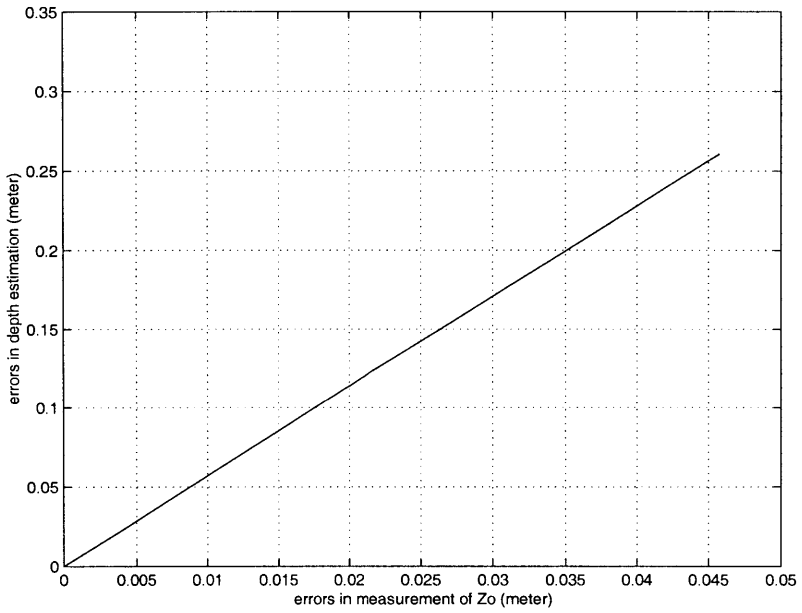


Figure 4-4: The relationship between the measurement errors in Z_0 and the corresponding errors in depth estimation results, δz_{lt}, is linear. In this particular plot, X_0 stays fixed at 0.089m and θ at 1.28^o. The corresponding gain factor is 5.6966.

In the last and final case of our error analysis, errors are allowed to occur only in the measurement of the rotation angle θ. An error in the measurement of the rotation angle causes the measured value of θ to deviate from its true value by $\delta\theta$ which in turn causes the results of depth calculation to be different from their ideal values by δz_{rt} and δz_{lt} respectively. Again, we apply the same technique as in the previous two cases to arrive at the following relationship:

$$\begin{pmatrix} \frac{\partial z_{rt}}{\partial \theta} \\ \frac{\partial z_{lt}}{\partial \theta} \end{pmatrix} = \begin{pmatrix} m & n & o \\ p & q & r \end{pmatrix} \begin{pmatrix} \frac{\partial r_{11}}{\partial \theta} & \frac{\partial r_{12}}{\partial \theta} & \frac{\partial r_{13}}{\partial \theta} \\ \frac{\partial r_{21}}{\partial \theta} & \frac{\partial r_{22}}{\partial \theta} & \frac{\partial r_{23}}{\partial \theta} \\ \frac{\partial r_{31}}{\partial \theta} & \frac{\partial r_{32}}{\partial \theta} & \frac{\partial r_{33}}{\partial \theta} \end{pmatrix} \begin{pmatrix} \frac{x'_{lt}}{f} \\ \frac{y'_{lt}}{f} \\ 1 \end{pmatrix} z_{lt}. \tag{4.22}$$

Taking the partial derivative of the rotation matrix $\mathbf{R}$ from Eq.(4.5), we obtain

$$\begin{aligned}\frac{\partial \mathbf{R}}{\partial \theta} &= \frac{\partial}{\partial \theta}\begin{pmatrix} \cos\theta & \sin\theta\sin\alpha & \sin\theta\cos\alpha \\ -\sin\theta\sin\alpha & \cos^2\alpha+\cos\theta\sin^2\alpha & \frac{\sin 2\alpha}{2}(\cos\theta-1) \\ -\sin\theta\cos\alpha & \frac{\sin 2\alpha}{2}(\cos\theta-1) & \sin^2\alpha+\cos\theta\cos^2\alpha \end{pmatrix}, \\ &= \begin{pmatrix} -\sin\theta & \sin\alpha\cos\theta & \cos\alpha\cos\theta \\ -\sin\alpha\cos\theta & -\sin^2\alpha\sin\theta & -\frac{\sin 2\alpha}{2}\sin\theta \\ -\cos\alpha\cos\theta & -\frac{\sin 2\alpha}{2}\sin\theta & -\cos^2\alpha\sin\theta \end{pmatrix}. \end{aligned} \tag{4.23}$$

Plugging Eq.(4.23) into the right-hand side of Eq.(4.22) and multiplying out, the following equations can be obtained:

$$\begin{aligned}\frac{\partial z_{rt}}{\partial \theta} &= \left[-m\frac{x'_{lt}}{f} - n\left(\sin^2\alpha\frac{y'_{lt}}{f}+\frac{\sin 2\alpha}{2}\right) - o\left(\frac{\sin 2\alpha}{2}\frac{y'_{lt}}{f}+\cos^2\alpha\right)\right] z_{lt}\sin\theta + \\ &\quad \left[m\sin\alpha\frac{y'_{lt}}{f}+m\cos\alpha-n\sin\alpha\frac{x'_{lt}}{f}-o\cos\alpha\frac{x'_{lt}}{f}\right] z_{lt}\cos\theta, \\ \frac{\partial z_{lt}}{\partial \theta} &= \left[-p\frac{x'_{lt}}{f} - q\left(\sin^2\alpha\frac{y'_{lt}}{f}+\frac{\sin 2\alpha}{2}\right) - r\left(\frac{\sin 2\alpha}{2}\frac{y'_{lt}}{f}+\cos^2\alpha\right)\right] z_{lt}\sin\theta + \\ &\quad \left[p\sin\alpha\frac{y'_{lt}}{f}+p\cos\alpha-q\sin\alpha\frac{x'_{lt}}{f}-r\cos\alpha\frac{x'_{lt}}{f}\right] z_{lt}\cos\theta. \end{aligned}$$

That is, the relationships between $\delta\theta$ and δz_{rt} and δz_{lt} can be described as follow:

$$\begin{aligned}\delta z_{rt} &= \left\{\left[-m\frac{x'_{lt}}{f} - n\left(\sin^2\alpha\frac{y'_{lt}}{f}+\frac{\sin 2\alpha}{2}\right) - o\left(\frac{\sin 2\alpha}{2}\frac{y'_{lt}}{f}+\cos^2\alpha\right)\right] z_{lt}\sin\theta + \right. \\ &\quad \left.\left[m\sin\alpha\frac{y'_{lt}}{f}+m\cos\alpha-n\sin\alpha\frac{x'_{lt}}{f}-o\cos\alpha\frac{x'_{lt}}{f}\right] z_{lt}\cos\theta\right\}\delta\theta, \qquad (4.24) \\ \delta z_{lt} &= \left\{\left[-p\frac{x'_{lt}}{f} - q\left(\sin^2\alpha\frac{y'_{lt}}{f}+\frac{\sin 2\alpha}{2}\right) - r\left(\frac{\sin 2\alpha}{2}\frac{y'_{lt}}{f}+\cos^2\alpha\right)\right] z_{lt}\sin\theta + \right. \\ &\quad \left.\left[p\sin\alpha\frac{y'_{lt}}{f}+p\cos\alpha-q\sin\alpha\frac{x'_{lt}}{f}-r\cos\alpha\frac{x'_{lt}}{f}\right] z_{lt}\cos\theta\right\}\delta\theta. \qquad (4.25) \end{aligned}$$

Figure 4-5 shows the linear gain factor between errors in the measurement of θ and the final errors in depth estimation. When X_0, Z_0 stay fixed, a small error of 1 radian in measuring θ will result in a corresponding error of 47.8634 meter in estimating z_{lt}.

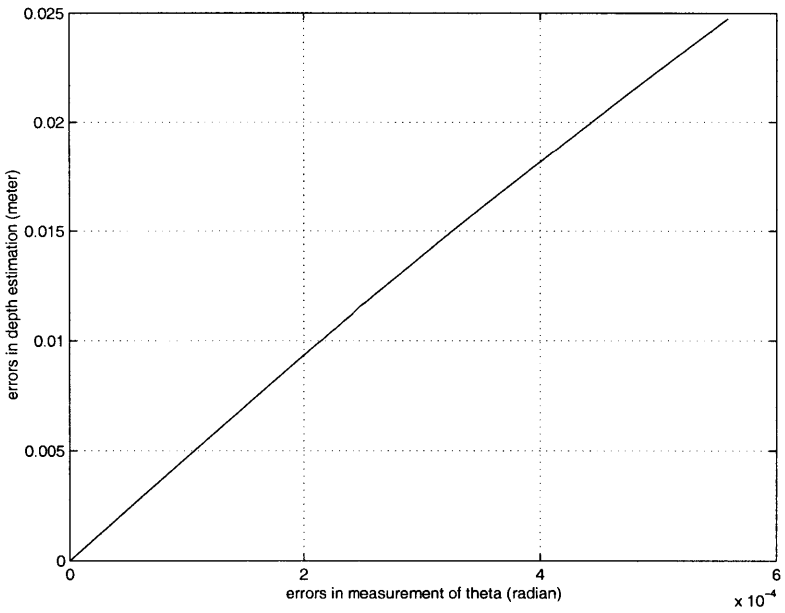


Figure 4-5: A plot shows the linear gain factor between the measurement errors in θ in radian and the corresponding errors in estimating depth, z_{lt}. The gain factor of this plot is 47.8634. That is, an error of 1 radian in measuring the rotation angle θ will result in an error of 47.8634 in estimating z_{lt}.

Chapter 5

Experimental Results

In this experiment, a video camera mounted on the rear window of a testing van supplies the source image sequences to our system. We tested our algorithm on several sets of video images taken in two parking scenarios–parallel and right-angle parking. This chapter presents the calculated distance in the form of sparse depth maps and analyzes errors in the results with the possible sources of their occurrences.

5.1 Experiments

We tested our algorithm on three sets of real video sequences taken from inside a parking lot. To avoid the problem of shadowing and excessive specular reflection, the shooting was done during an evening time. The first two sequences are obtained when the testing van attempts to park in parallel as shown in Figure 5-1. The initial distances X_{ori} and Z_{ori} between the testing van and the parked car in the front (the car labeled as **A** in Figure 5-1) are measured. At this starting point, the angle that the body of the car makes with the $z-$axis of the world coordinate is also measured to be θ_0. In the first test sequence, X_{ori} and Z_{ori} are measured to be 135 cm and 190 cm respectively. And the value of θ_0 is $+2.47^o$ in the counter-clockwise direction measured from the $z-$axis. As the testing van is being driven into the parking space between car **A** and car **B**, the video camera mounted on the rear window captures the different views of the parking lot. Simultaneously, as the images are captured,

the wheel encoders located at both the rear wheels of the van and the steering sensor record the distance that the van has translated and the change in rotation angle from its original value.

Figure 5-2 and Figure 5-3 show sample images from the first test sequence. Mainly, the trunk part of car **A** and the front part of car **B** are visible from the camera. Figure 5-2 corresponds to the first frame in the sequence and Figure 5-3 shows the 21^{st} image of the same sequence.

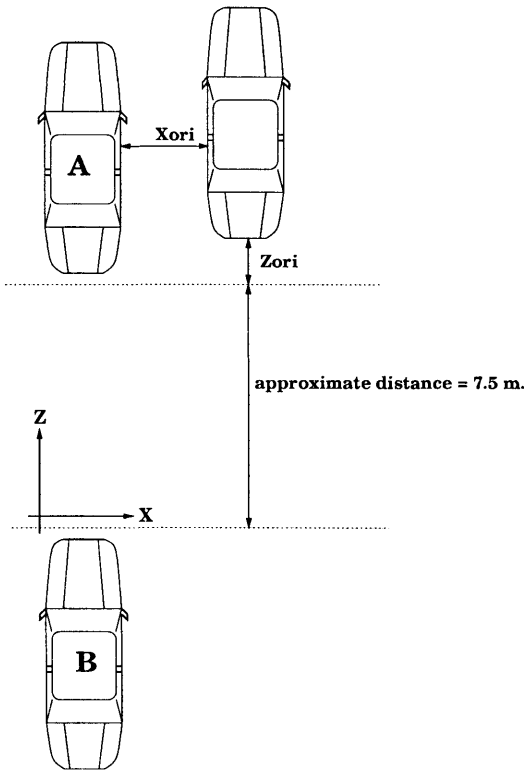


Figure 5-1: Parallel parking scenario

The second test sequence is obtained from another attempt of parallel parking. This time, the initial distance X_{ori} and Z_{ori} are measured to be 105 cm and 195 cm from car **A** respectively, and the initial angle of the car with respect to the $z-$axis is $+1.52^{o}$ measured in the counter-clockwise direction.

The last test sequence is obtained when the testing van attempts to park from the right angle position. Figure 5-4 illustrates this scenario. The initial distance X_{ori} and Z_{ori} from car **A** in Figure 5-4 are measured to be 162 and 355 cm and the initial

Figure 5-2: The first frame in the first test sequence

angle of the vehicle is -52.99^o measured in the counter-clockwise direction from the $z-$axis. The front part of car **A** and the side of car **B** are visible from the camera. We would like to estimate the distance of all objects in the scene from the vehicle position.

Feature points are first extracted from each image using the CSS based corner detector. Corresponding features are then matched and tracked through the sequence using the matching algorithm described in Chapter 3. Our system keeps track of the locations of feature points via the use of an internal table which is updated when the new frame is obtained. With the rotation angle obtained from inertial sensors and the distance information from the wheel encoders, depths of objects in the scene can be calculated from the difference in locations of the feature points in any two image frames in the sequence. In this research, we tried experimenting with the different lengths of baselines in computing distance. A plot between the calculated distance and the length of baseline is shown in Figure 5-5, which illustrates that the use of a longer baseline, i.e., picking the two image frames that are far apart in the sequence, allows for depth calculation to be more accurate and stable. Two approaches can be

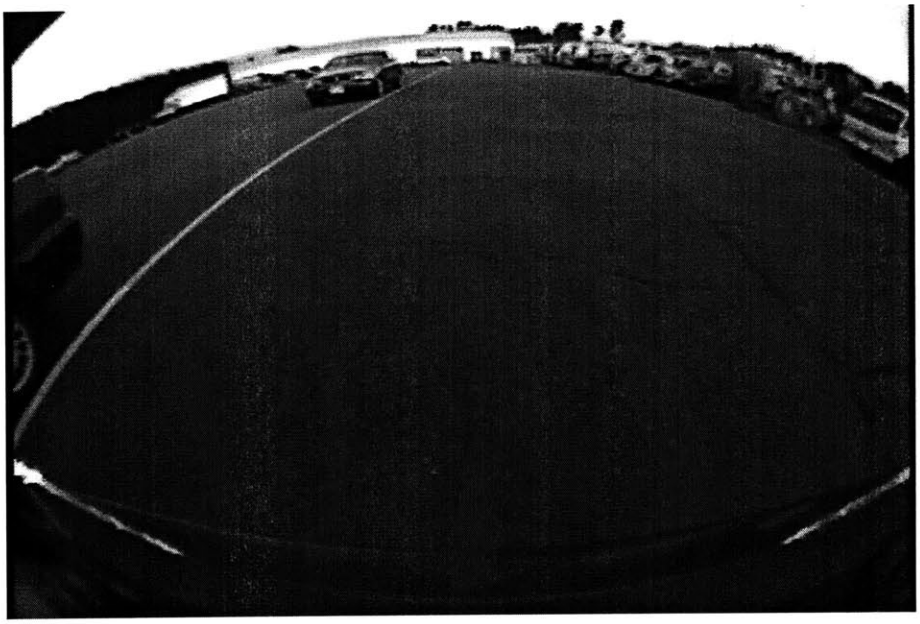


Figure 5-3: The 20^{th} frame in the first test sequence

taken in obtaining the final depth estimation. One approach is to average the results of different estimation from the use of different baselines by putting more weights on the results from longer baselines. Another approach is to fix the length of baseline or fix the number of frames that the two images used in depth calculation must be apart. The optimal value of this fixed length or fixed number of frames must be obtained via experimentation in order for the depth calculation results to stabilize. As seen in Figure 5-5, the value to use for a fixed baseline length can be set to 0.4m for the calculated depth results to stabilize. For convenient purposes, we, however, chose to fix the number of frames. Through experimentation the fixed number of frames that yields optimal depth calculation results is found to be in the range of 20-30.

5.2 Generating a Sparse Depth Map

By marking the feature points in the scene that are of different distance from the camera with different colors, a sparse depth map of the environment in the parking lot can be obtained. We use 20 different colors to denote 20 distance ranges. The color assigned to each feature point in a particular map will be based on the range

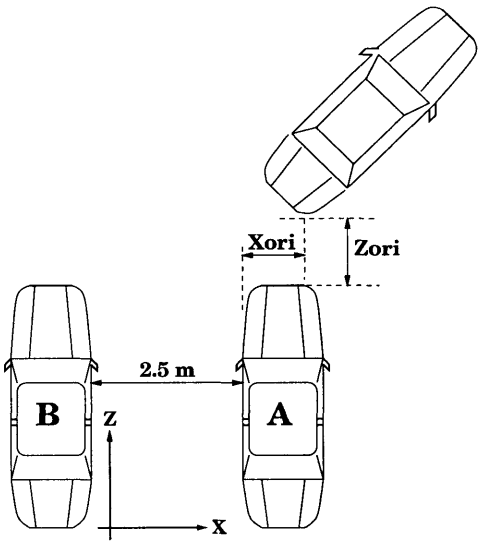


Figure 5-4: Right-angle parking scenario

distribution of all the feature points in the scene. Specifically, after the distance of each feature point from the camera is computed, the maximum and minimum depths are determined for the color assignment. Let Z_{max} denotes the maximum and Z_{min} the minimum distance of all feature points in a particular map, the distance resolution of this map is $d = \frac{Z_{max}-Z_{min}}{20}$. In other words, two subsequent colors will represent points that are d meters apart in depth. In addition, our convention is such that an object located closer to the camera will be assigned a darker color, and the further away the lighter the color represented.

The depth map from the first test sequence of parallel parking is shown in Figure 5-6. All objects in this map can be roughly grouped into three big ranges. The first group contains the feature points from the part of the scene that corresponds to the trunk of car **A** in the parallel parking scenario in Figure 5-1. The second group contains the feature points from the distant background of the parking lot and the front part of car **B**, which is located rather far away from the camera. The last group consists of feature points from the cars in the near background that are located closer to the camera (on the right side of the image in Figure 5-3). In the depth map of the first test sequence, the maximum calculated distance from the camera is 9.4305 m and the minimum 0.25 m. That is, the distance resolution for this particular depth

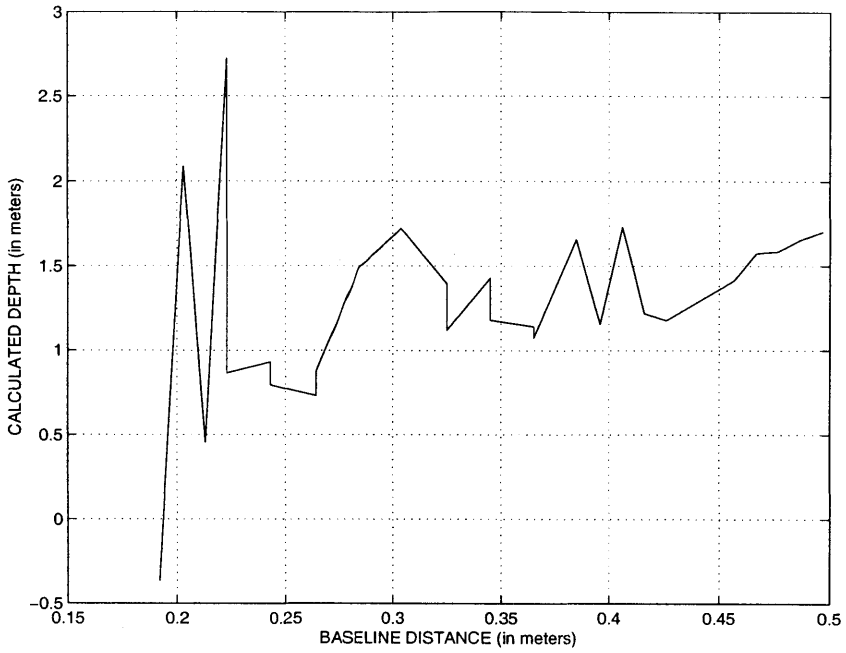


Figure 5-5: Plot of different baseline length and the estimated distance. With a smaller baseline, the depth estimation is not very reliable as can be seen from the graph above. Although a longer baseline is preferred in calculating depth, the tradeoff is that tracking the same feature point in a long sequence of images can become very difficult. In the worse scenario, objects might disappear from the scene and thus depth calculation using a longer baseline is not always feasible.

map is $\frac{9.4305-0.25}{20} = 0.4590m$.

The second test sequence is obtained from another shooting of a parallel parking experiment when the testing van has moved closer to the parked car **B**. As seen in the sparse depth map shown in Figure 5-7, the background environment and car **B** can be distinctly identified from the difference in their relative depths from the camera. The distance resolution of this map is $\frac{13.1694-0.4192}{20} = 0.6375m$.

In the third test sequence, the testing van attempts a right-angle parking. In the particular frame that we show in Figure 5-8, car **A** is already invisible in the map. The detection of the distance of feature points along the lane markers and the side of car **B** again shows a clear distinction in the distance from the feature points in the background. The distance resolution of this map is $\frac{14.3038-0.3688}{20} = 0.6967m$.

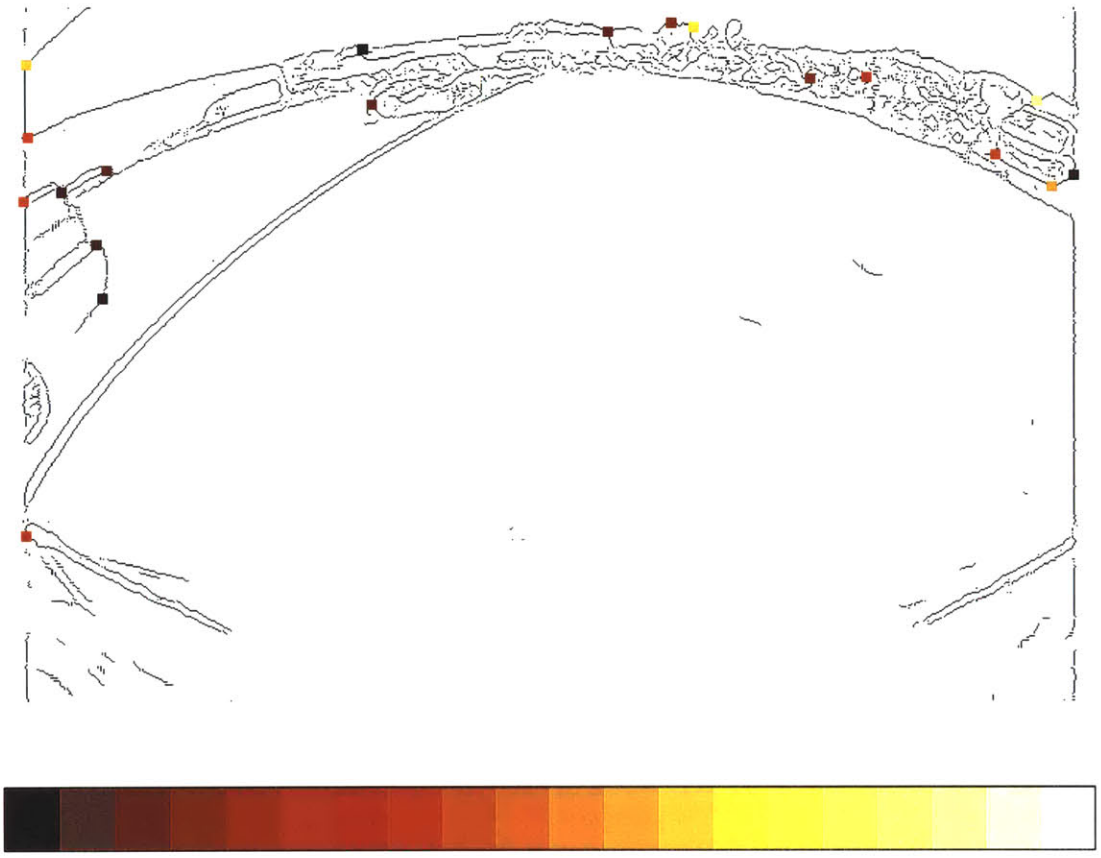


Figure 5-6: Sparse depth map of the parking lot obtained from the first test sequence. Feature points in the scene can be grouped into three big ranges. The first group on the left side of the map contains those feature points that belong to the trunk of car **A**. The second group is located at the top center of the map contains the feature points of the distant background and the feature points of the front part of car **B**. The third group on the right side of the map contains feature points of the cars in the background that are closer to the camera.

5.3 Accuracy of the Computed Depth

In the first test sequence of the parallel parking scenario, the distance between the camera location and a point at the rear end of car **A** in Figure 5-1 is known by subtracting the distance the testing van has moved from the original Z_{ori} distance. We, therefore, have obtained a close approximation of the actual distance of an object in the scene, i.e., car **A**. The calculated distances when the testing van is located at different distances away from car **A** are obtained and the results are presented in Table 5.1. As apparent in the positive signs in the errors, our calculation, most of

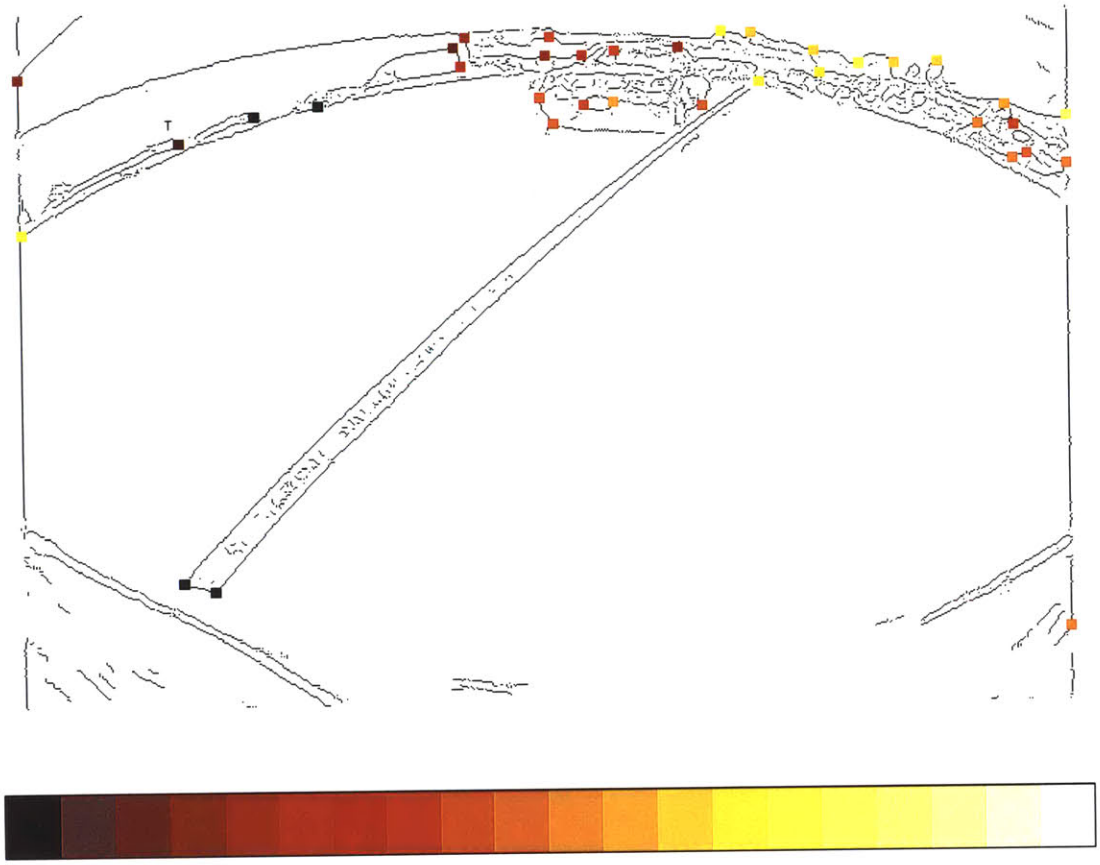


Figure 5-7: Sparse depth map of the parking lot obtained from the second test sequence. Car **A** is already invisible from the camera and the testing van has moved closer to car **B**. There are distinct differences in colors. The feature points that belong to the front of car **B** are in orange while most of background have distance in the yellow range.

the time, gives a lower estimate of the actual depth. The peak magnitude of errors in the sample results of this test sequence is 0.5846m.

Similarly in the second test sequence, the initial distance between the camera position and the front part of car **B** is known *apriori* to be $Z_{ori} + 7.50m$. By subtracting the distance the van has moved along the $z-$axis from the original separation, we can determine the actual distance between the camera and the front part of car **B** as listed in the Measured Distance column in Table 5.2. In this second test sequence, the peak magnitude of error is found to be 0.7654m.

Table 5.1: A comparison between the actual measured distance and the computed distance between the camera location and car **A** in parallel parking scenario in Figure 5-1

Measured Distance(m)	Computed Distance(m)	Error(m)
1.5934	1.2070	+0.3864
1.5651	1.2802	+0.2849
1.5382	1.1870	+0.3512
1.5199	1.2804	+0.2395
1.4689	1.0347	+0.4342
1.4525	1.0223	+0.4302
1.4367	0.8521	+0.5846
1.4209	1.0924	+0.3285
1.4046	1.1058	+0.2988
1.3898	0.8206	+0.5692
1.3671	0.9242	+0.4429
1.3602	0.9704	+0.3898
1.3528	1.0450	+0.3078
1.3454	1.0879	+0.2575
1.3386	1.0680	+0.2706

Table 5.2: A comparison between the actual distance and the computed distance between camera location and car **B** in the setting in Figure 5-1

Measured Distance(m)	Computed Distance(m)	Error(m)
3.5287	3.4268	+0.1019
3.4919	2.7265	+0.7654
3.4688	2.9333	+0.5355
3.4201	3.4066	+0.0135
3.3600	2.6903	+0.6697
3.3354	3.7195	-0.3841
3.3126	2.8212	+0.4914
3.3126	2.8262	+0.4864
3.2890	2.7993	+0.4897
3.2890	3.1628	+0.1262
3.2771	3.7073	-0.4302
3.2655	3.5001	-0.2346
3.2421	3.1337	+0.1084
3.2191	3.3212	-0.1021
3.1957	3.5830	-0.3873
3.1715	3.2626	-0.0911

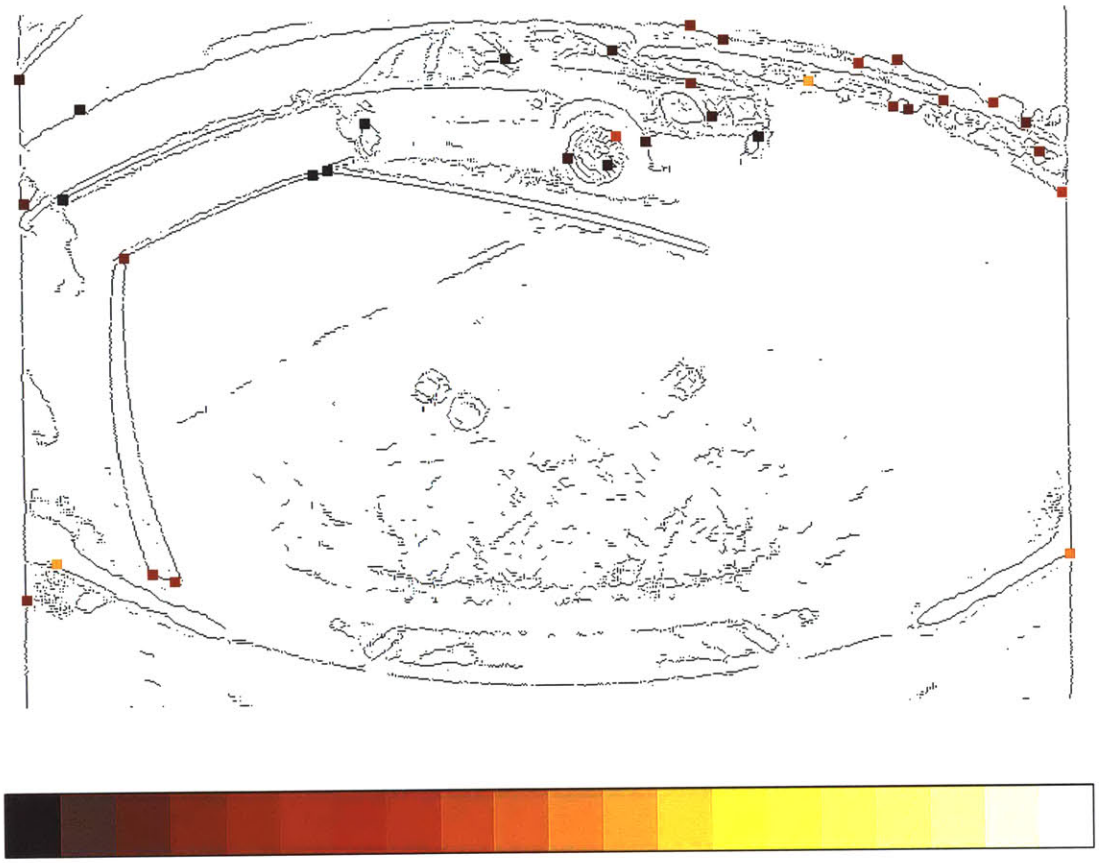


Figure 5-8: Sparse depth map of the parking lot of the third sequence–right angle parking. Again, in this scenario, car **A** is already invisible from the camera. Although some feature points (especially on the road) that should be detected closer to the camera appear to be farther way, in all, there is a clear distinction between the feature points belonging to car **B** and those of the distant background.

5.4 Sources of Error

Due to the imperfection of the physical implementation of the system, errors are inevitable in our estimation of distance. In general, we can broadly identify two different sources of errors. The first are errors from the imperfect measurements of parameters obtained from other sensors, namely, errors introduced in the measurement of θ, X_0, Z_0 and α. As described in the previous chapter, θ is measured using the steering sensors attached to the steering wheel of the vehicle, and X_0 and Z_0 are obtained from the wheel encoders attached to the two rear wheels of the vehicle. In general, errors in distance measurement from the use of wheel encoders and inertial

sensors are cumulative in nature. Thus, the depth estimation results will tend to be less reliable and the sensors will need to be reset at times. Other sources of errors in this class are small measurement errors in different relevant parameters such as the fixed tilting angle α that the optical axis of the camera makes with the road surface plane. As the vehicle moves, small perturbations on the road will constantly change the camera fixed orientation, thus causes some errors in the depth calculation results. Errors are also caused by the imperfection of the optical system itself. Even though we have attempted to compensate for some radial distortion for the use of wide-angle lens in our calculation, the results are still not truly ideal.

Another source of errors comes from the imperfect nature of our calculation. In the feature extraction stage, mis-localization of feature points can occur that will distort the final results of depth calculation. In particular, with our corner detection scheme based entirely on the reliability of the edge contour detection results, slight changes in the lighting condition will result in dissimilarity between edge contours detected between subsequent frames in the sequence, hence raising an issue of consistency and stability of the corner points detected.

Similarly, in correspondence mismatching, despite the fact that matching is done between consecutive frames in densely sampled video sequences, missed correspondence and false correspondence still frequently occur and the errors will directly propagate to become errors in depth estimation. In fact, we found that the majority of the spurious depth estimation results in our depth map are attributed to correspondence mismatching.

Chapter 6

Conclusions

6.1 Conclusions

In this thesis, we explore the problem of 3-D reconstruction of the environment in a parking lot from a sequence of video images taken from a moving camera. We propose an algorithm to recover distance of objects in the scene provided that the motion and position of the camera are known. First, our algorithm extracts feature points from each image in the sequence using the CSS based corner detection scheme. Feature points are then matched and tracked through the sequence, and depths can be calculated to generate a sparse depth map of the environment. The sample results from testing our algorithm on real video image sequences have proven the reliable performance of our algorithm in reconstructing a relative depth map of the environment. However, wrong distance estimation can be spuriously spotted in the depth map results. The main cause of these errors is attributed to false correspondences which occur in the matching and tracking stage.

6.2 Future work

We propose the following extensions to this research:

1. Improve reliability of the depth estimation by exploring different schemes in matching and tracking corresponding corner points in the image sequence. A

method that makes use of a combination of different cues in matching correspondence should be explored. For example, the matching results from our CSS based method can be used as the initial solutions in matching. The precise locations of the matching points can then be refined using the normalized correlation coefficient as the new matching criterion.

2. Make the algorithm more robust under the existence of shadows on the road. This case is an extension to the current research to cover the case where parking is attempted in an outdoor parking lot during the day.

Appendix A

Edge Linking

This section describes in details an edge linking algorithm used in the thesis. Since the final binary edge detection output is obtained from OR'ing the binary edge maps of the three color components–R, G, B. Due to the slightly different response from each channel, many gaps have automatically been filled and edges are automatically linked via the OR'ing process. Therefore, the linking algorithm implemented in this thesis can be simplified. The algorithm inspects a 3x3 local neighborhood window and compares the intensity pattern with the different patterns for vertical and horizontal edge linking to determine if a gap between edge pixels should be filled. For the case of a vertical edge linking, 10 patterns as shown in Figure A-1 are compared. The 3x3 local window is then compared with the 90^o-rotated, 180^o-rotated, 270^o-rotated, transposed versions of each of these 10 masks. If a match occurs, the middle pixel is filled.

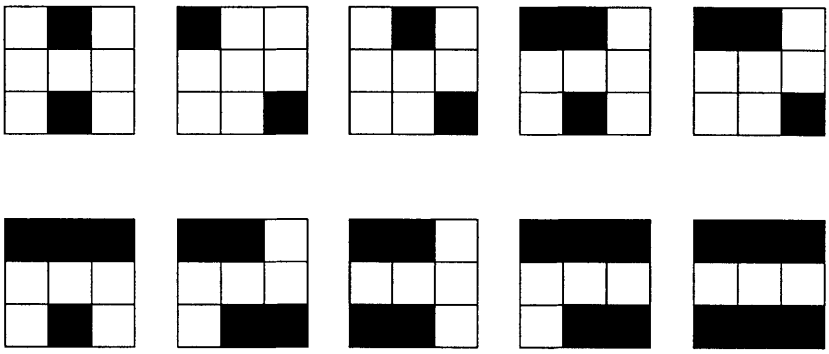


Figure A-1: A simplified edge linking algorithm used in this thesis inspects a 3x3 local neighborhood window to find the linking patterns. For vertical edge linking, 10 masks as shown above are used. The local 3x3 intensity pattern window is compared with the original, 90^o-rotated, 180^o-rotated, 270^o-rotated, and the transposed versions of these masks. If a match occurs, the middle pixel is filled.

Appendix B

Edge Thinning I

In this section, we describe in details an edge thinning algorithm based on the algorithm proposed by Heipke et al. in [11]. The goal is to remove any extraneous edge pixels that are not parts of the edge contour skeletons. Heipke et al. considered an edge pixel a part of a skeleton if there exists only two other edge pixels in its 3x3 neighborhood. To properly thin edges so that only the skeleton remains, Heipke et al. proposed three thinning conditions under which a thinning algorithm must satisfy. These three conditions are:

1. Thick edges are not split into several thin edges but are thinned to one edge
2. No holes can be created in edges
3. Edges are not shortened but must remain of the same length.

Using the above three thinning rules, Heipke reduced from all the possible patterns in a 3x3 window to only 17 combinations of thick edge masks. Figure B-1 listed all the 17 thick edge masks. Heipke's algorithm then examines a 3x3 neighborhood of an edge pixel and compares the edge pattern with the original, transpose, 90^o-rotated, 180^o-rotated, 270^o-rotated, 90^o-rotate of the transpose, 180^o-rotate of the transpose, and 270^o-rotate of the transpose versions of each of the 17 thick edge patterns. If a match occurs, the middle edge pixel will be removed. If edge contours are not too thick, only one pass of the algorithm is sufficient to thin edges to one-pixel wide. Otherwise, multiple passes of the algorithm are required.

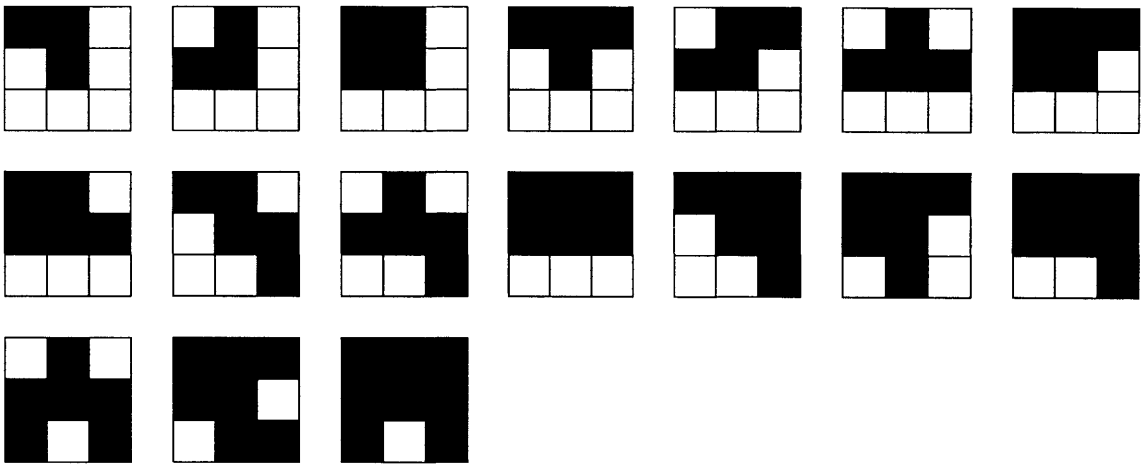


Figure B-1: Under Heipke's proper thinning rules, the 17 possible 3x3 thick edge masks, their transposes and their rotated versions are used to compare with a 3x3 neighborhood of an edge pixel. If a match occurs, the pixel in the middle is considered extraneous and thus will be removed.

Appendix C

Edge Thinning II

This section discusses a fast parallel thinning algorithm proposed by Chen and Hsu in [5]. This algorithm is a modified version of the original Zhang-Suen method [32] which is based on a 4-neighbor connectivity. The original Zhang-Suen algorithm is well-known for its thinning merit against contour noise immunity by employing a 2-subiteration scheme in thinning. However, because the algorithm was designed based on 4-neighbor connectivity, the concept of "one-pixel wide" edge contours cannot be achieved entirely using the original Zhang-Suen algorithm. In 1988, Chen and Hsu proposed their algorithm based on 8-neighbor connectivity and still preserved the original merit of the Zhang-Suen algorithm. In Chen and Hsu algorithm, a 3x3 window around an edge pixel P_1 is examined as seen in Figure C-1. Let $A(P_1)$ be the number of 0-1 patterns in the ordered set $P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9$, and P_2. $B(P_1)$ denotes the number of nonzero neighbors of P_1. One pass of the algorithm is divided into two subiterations. In the first subiteration, the edge pixel P_1 is removed if it satisfies all the following conditions:

1. $2 \leq B(P1) \leq 7$;
2. $A(P1) = 1$;
   (a) $P_2 \cdot P_4 \cdot P_6 = 0$ and
   (b) $P_4 \cdot P_6 \cdot P_8 = 0$
3. $A(P1) = 2$;

(a) $(P_2 \cdot P_4 = 1$ and $P_6 + P_7 + P_8 = 0)$ or

(b) $(P_4 \cdot P_6 = 1$ and $P_2 + P_8 + P_9 = 0)$

The criterion in (2) is considered a criterion of 4-neighbor connectivity. Chen and Hsu incorported additional conditions in (3) to thin edge lines to one-pixel wide from the standpoint of 8-connectedness. In the second subiteration, the isotropic property proposed in the original Zhang-Suen algorithm that gives this algorithm an immunity against contour noise is employed. The edge pixels P_1 will be removed if all the following conditions are met.

1. $2 \leq B(P1) \leq 7$;
2. $A(P1) = 1$;
   (a) $P_2 \cdot P_4 \cdot P_8 = 0$ and
   (b) $P_2 \cdot P_6 \cdot P_8 = 0$
3. $A(P1) = 2$;
   (a) $(P_2 \cdot P_8 = 1$ and $P_4 + P_5 + P_6 = 0)$ or
   (b) $(P_6 \cdot P_8 = 1$ and $P_2 + P_3 + P_4 = 0)$

P9	**P2**	**P3**
P8	**P1**	**P4**
P7	**P6**	**P5**

Figure C-1: Chen and Hsu algorithm examines a 3x3 window around an edge pixel P_1. The middle pixel, P_1, is deleted when the criteria for 4-neighbor connectivity, i.e., logical operations on the subsets of (P_2, P_4, P_6, P_8), and 8-neighbor connectivity, i.e., logical operations on the subsets of (P_3, P_5, P_7, P_9) are met.

Bibliography

[1] H. H. Baker. Depth from edge and intensity based stereo. Technical report, Stanford University, Department of Computer Science, Stanford, CA, September 1982.

[2] S. T. Barnard. Stochastic stereo matching over scale. *International Journal of Computer Vision*, pages 17–32, 1989.

[3] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communication*, 31:532–549, April 1983.

[4] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[5] Y. S. Chen and W. H. Hsu. A modified fast parallel algorithm for thinning digital patterns. *Pattern Recognition Letters*, 7:99–106, February 1988.

[6] A. K. Dalmia and M. M. Trivedi. Depth extraction using lateral or axial camera motion: an integration of depth from motion and stereo. In *Proc. of International Conference on Image Processing*, volume 1, pages 414–417, 1995.

[7] E. D. Dickmanns and V. Graefe. Dynamic monocular machine vision. *Machine Vision and Applications*, 1:223–240, 1988.

[8] Y. Fujii. *Robust Monocular Depth Perception.* PhD dissertation, University of Michigan, Ann Arbor, Department of Nuclear Engineering, 1992.

[9] W. E. L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 7(1):17–34, January 1985.

[10] C. G. Harris. Determination of ego-motion from matched points. In *Proc. Alvey Vision Conf.*, Cambridge, UK, 1987.

[11] C. Heipke, A. Englisch, T. Speer, and R. Kutka. Semi-automatic extraction of roads from aerial images. In *Proc. SPIE ISPRS Commission III Symposium*, volume 2357, pages 353–360, 1994.

[12] B. K. P. Horn. *Robot Vision.* MIT Press, Cambridge, MA, 1986.

[13] B. K. P. Horn. Relative orientation revisited. *Journal of the Optical Society of America*, 8:1630–1638, October 1991.

[14] L. Kitchen and A. Rosenfeld. Gray level corner detection. *Pattern Recognition Letters*, pages 95–102, 1982.

[15] S. Mallat. Wavelets for a vision. *Proceedings of the IEEE*, 84(4):604–614, April 1996.

[16] D. Marr and T. Poggio. A theory of human stereo vision. In *Proceedings of Royal Society (London)*, pages 301–328, 1979.

[17] I. Masaki. Machine-vision systems for intelligent transportation systems. *IEEE Computer Society on Intelligent Systems*, 13(6):24–30, December 1998.

[18] R. Mehrotra, S. Nichani, and N. Ranganathan. Corner detection. *Pattern Recognition*, 23(11):1223–1233, 1994.

[19] F. Mokhtarian. Multi-scale description of space curves and three-dimensional objects. In *Computer Society Conf. on Computer Vision and Pattern Recognition*, pages 298–303, 1988.

[20] F. Mokhtarian and R. Suomela. Robust image corner detection through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1376–1381, December 1998.

[21] H. P. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, Jet Propulsion Laboratory, Computer Science Department, Stanford, CA, September 1980.

[22] Y. L. Murphy, J. Chen, J. Crossman, J. Zhang, P. Richardson, and L. Sieh. Depth finder–a real-time depth detection system for aided driving. In *Proceedings of the IEEE Intelligent Vehicle Symposium*, pages 122–127, October 2000.

[23] C. L. Novak and S. A. Shafer. Color edge detection. In *DARPA Image Understanding Workshop*, volume 1, pages 35–37, February 1987.

[24] M. Okutomi and T. Kanade. A multiple-baseline stereo. In *Conf. on Computer Vision and Pattern Recognition*, pages 63–68, June 1991.

[25] A. Rosenfeld and E. Johnston. Angle detection on digital curves. *IEEE Transactions of Computers*, 22:875–878, 1973.

[26] I. K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1), January 1987.

[27] C. Shekhar and R. Chellappa. Visual motion analysis for autonomous navigation. Technical report, University of Maryland, College Park, MD, 1992.

[28] S. M. Smith and J. M. Brady. Susan–a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[29] Katherine Treash. Automatic road detection in grayscale aerial images, 1998.

[30] A. Witkin. Scale space filtering. In *Proceedings of International Joint Conference in Artificial Intelligence*, 1983.

[31] X. Xie, R. Sudhakar, and H. Zhuang. Corner detection by a cost minimization approach. *Pattern Recognition*, 26(8):1235–1243, 1993.

[32] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *ACM*, 27:236–239, 1984.

[33] X. Zhang. A moving corner detector for dynamic vehicular images. In *IEE Colloquium on Image Processing for Transport Applications*, pages 5/1–5/4, 1993.

[34] Q. Zheng and R. Chellappa. Automatic feature point extraction and tracking in image sequences for arbitrary camera motion. Technical report, University of Maryland, Center for Automation Research, Computer Vision Laboratory, College Park, MD, June 1992.