University of Alberta

Panoramic Computer Vision

by

Mark Fiala

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2002

Your file Votre référence
ISBN: 0-612-82193-5
Our file Notre référence
ISBN: 0-612-82193-5

University of Alberta

Library Release Form

Name of Author: Mark Fiala

Title of Thesis: Panoramic Computer Vision

Degree: Doctor of Philosophy

Year this Degree Granted: 2002

M Fiala

Mark Fiala
360 Weber Way
Edmonton, AB
Canada, T6M-2H2

Date: MAY 17/2007

To percieve one must recreate

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Panoramic Computer Vision** submitted by Mark Fiala in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dr. Anup Basu

Dr. Max Meng

Dr. Terry Caelli

Dr. Nelson Durdle

Dr. Petr Musilek

Dr. Gerhard Roth

Dr. Hong Zhang

Date: May 10, 2002.

To my mother and father
for their moral and financial support
over the many, many
years it took for this degree

Abstract

Computer vision is the study of image capture and image understanding. A computer vision system can enable a robot to build a model of its surroundings and navigate by sight. The use of a *panoramic* camera to enable omni-directional (360°) computer vision is explored. Such a view can be obtained with just one image sensor if a curved mirror is introduced into the optical path of a conventional digital or video camera. Using a curved mirror introduces complications when attempting to extract the projections of straight line features from an image. The current literature with panoramic imagery has introduced a method for detecting these important features. However, this limits the freedom of what mirror shapes can be used. This thesis provides a new approach to recognizing straight lines in panoramic images from omnidirectional cameras free of these restrictions, and is motivated by the goal of creating a *panoramic computer vision* for robotic systems. The theory for line detection is applied to three computer vision tasks: stereo model making, mobile robot localization and shape-through-motion. Three-dimensional model-making is achieved with a novel mirror design which provides two viewpoints thus enabling *stereo vision.* A mobile robot positioning system is designed using a non-stereo panoramic mirror in which the reflection of environment landmarks are recognized and tracked. Prototype vision systems for modeling and localization were built with theoretical and practical considerations which are described in this thesis.

Acknowledgements

The author wishes to acknowledge the support of Dr. Anup Basu, of the University of Alberta. Dr. Basu formed a collaboration with three industry partners to study panoramic vision in 1993. PVSI (Panoramic Viewing Systems Inc.), DRES (Defense Research Establishment Suffield) and NUS (Northern Underwater Systems) provided funding support to develop viewing systems using panoramic image sensors. These companies were interested in imaging systems for human viewers, not automatic computer vision systems, but their work enabled the novel panoramic sensors to be available for this research. The support of Dr. Max Meng in research supervision, Jevan Grylls in editing, and all of the committee members is appreciated. The assistance provided by my mother must be acknowledged, whose extensive English background and many long hours of patient proof-reading helped bring this thesis to completion.

Contents

Chapter 1

Vision

Vision is the act of creating a useful abstraction of the world through light emitted and reflected by objects in it. Computer vision can be defined as the processing of image data with the resultant observations and abstractions being used by some automatic system, such as a robot.

Classical computer vision research has typically used narrow field of view image sensors, such as digital or video cameras, mostly because of their availability and not necessarily as a choice based on optimal design. Many robot systems that could use computer vision would benefit from a wider or panoramic field of view.

1.0.1 Motivation and Paradigm for this Research

The work in this thesis is motivated by a desire to advance the technology of mobile robotics, particularly that in man-made environments such as factories, warehouses and offices. Parts of a vision system were designed in this thesis for this purpose. The language of a polyhedral world was used, and vision systems were designed to model and navigate within a world assumed to consist of these primitives. The intermediate levels of abstraction are intensity edges within an image, and the projections of straight edges of polygons in an image.

The work of this thesis seeks to create a vision system that produces scene models of polygons from image pixels, and to allow a mobile robot to track its position given that it is navigating in an environment described by an existing model in an abstraction language of polyhedrons. The paradigm of polyhedral vision is not new, but what is new is the application of new image sensors, new "eye" designs to computer vision and the successful adaptation of the this paradigm to a new class of image sensors.

1.1 "Eye Design": Image Sensors

Computer vision systems have typically used conventional video cameras or digital cameras as the light capturing device and sought to program computers to understand the scene projected onto a planar surface through a virtual pinhole. The pinhole model is the basis of the perspective projection that we as humans expect to see in an image (although the image sensors in our eyes are not planar). Light rays within a viewing range that pass through a single point are projected onto a planar light sensitive imaging surface. Lenses are an invention to capture more light but closely approximate the equivalent view through a very small aperture. However, the range of view that a conventional video, film or digital camera can see is quite small, one only needs to watch a home movie to realize how constricting a view these devices provide.

Recent work has seen the use of non-perspective image projections for use in capturing images with a wider field of view. Omni-directional sensors that can capture a 360° field of view can be built using a mixture of mirrors and lenses in the optical path. These so called *catadioptric* cameras can simultaneously capture light with a wider field-of-view than a conventional *dioptric* (containing only lenses in the optical path) camera.

Omni-directional viewing would require many standard narrow field of view cameras, or one panoramic camera. A single camera system has the advantages of processing only one continuous image, without the cost of muliple cameras and discontinuities at the boundaries of view that multiple cameras would introduce.

1.1.1 Panoramic Catadioptric Imaging Systems

Imaging systems, either biological or man made, use the basic components of pinholes, lenses and mirrors to form images on a photosensitive two-dimensional surface. An imaging system that uses only lenses to capture light is a *dioptric* system. One that uses mirror elements only, such as some telescopes, is a *catoptric* system. One that utilizes both lenses and mirrors in the image forming process is a *catadioptric* imaging system. An imaging system that utilizes a mirror can be combined with the conventional dioptric paradigm to produce a catadioptric system capable of capturing a (360°) field of view with only one image sensor (Figure 1.1).

Basu [14], Yagi [116] and others [18, 17] have demonstrated such systems. Many researchers have participated in this research in recent years as evidenced by the number of researchers at *Omnivis 2000, 2001*, a special forum on omnidirectional viewing held at the Computer Vision and Pattern Recognition (CVPR) conferences [70, 78, 21, 26, 62].

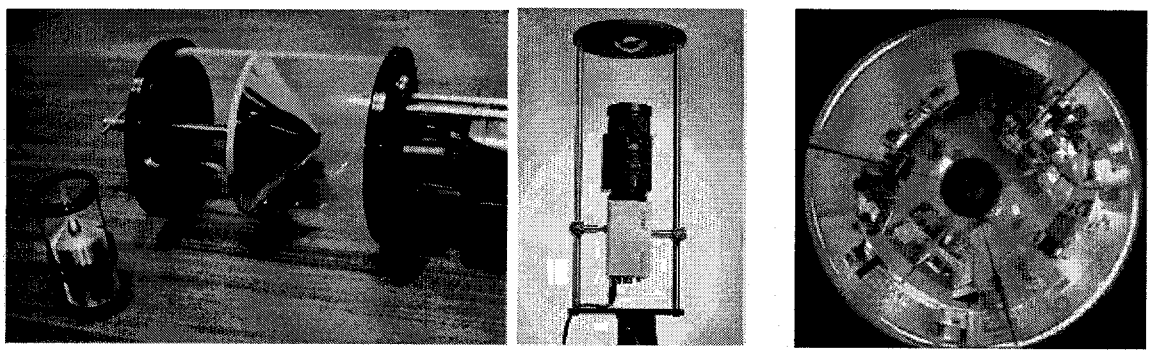


Figure 1.1: *A catadioptric imaging system consists of a lens, mirror and image plane sensor. A conventional video or digital camera with carefully chosen lens and a mirror can constitute a panoramic computer vision sensor.*

1.1.2 The View With Catadioptric Sensors and the *Panoramic Hough Transform*

However, the view from a catadioptric camera/mirror system curves straight line edge in the environment as in Fig. 1.2. The shape of the projection of straight lines, the edges of the desired polyhedral objects, are dependent on the curvature shape of the mirror and in some cases allow for easy extraction and in other cases not, according to whether the radial profile of the mirror fits certain criteria. This so-called *Single Viewpoint* (SVP) criteria greatly restricts the sensor design, and so a processing technique called the *Panoramic Hough Transform* was developed in the research for this thesis to allow extraction of lines with the general class of a panoramic catadioptric sensor free from the SVP restriction.

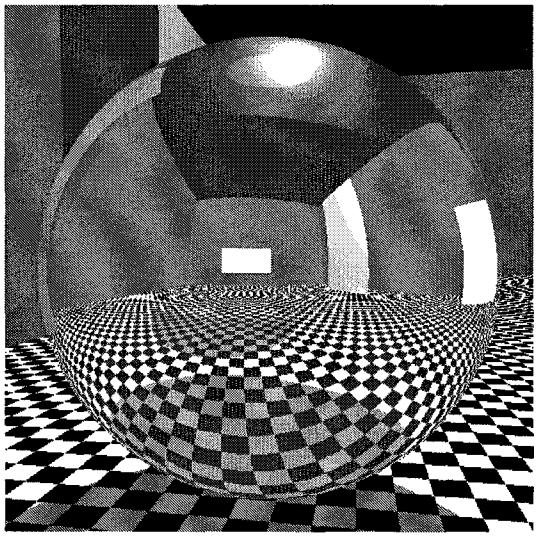


Figure 1.2: *Synthetic image demonstrating curved reflection from a convex mirror. Only straight lines parallel to the camera-mirror axis are projected as straight on the image.*

1.1.3 Finding Features in non-SVP Panoramic Images

To find line features in panoramic images, recent work in the field has shown a predictable projection can occur with SVP mirrors of parabolic and hyperbolic profiles with correct lenses and geometry. This has led to a belief that panoramic catadioptric systems should be built with these two profiles only, because of this feature detection capability.

The *Panoramic Hough Transform* developed in this thesis enlarges the useful family of catadioptric panoramic sensors by providing feature extraction methods to process imagery from sensors that do not meet the *Single Viewpoint* criteria. This transform expands the set of usable family of catadioptric panoramic sensors from those that are only of a parabolic or

hyperbolic radial profile to all catadioptric systems that have a convex radially symmetric profile . This theory enables panoramic vision systems to be simpler and less costly, and permits the use of mirrors with a custom resolution distribution.

The Panoramic Hough Transform is applied in this thesis to three applications using non-SVP mirrors. First is a mobile robot localization system that recognizes and tracks landmarks in the imagery from a single-lobed panoramic sensor. A second mobile robot application is a *shape-through-motion* system whereby the relative trajectories of image points are recognized as horizontal lines and a 3D model built. Thirdly, the transform is applied also to a bi-lobed system, where the image contains reflections from two non-SVP mirror surfaces and permits three-dimensional reconstruction in one image.

1.2 Panoramic Stereo

This catadioptric single-camera panoramic imaging concept has been extended to two lobes to provide a stereo panoramic view [31, 11] as shown in Fig. 1.3. The centers of the mirror lobes are collinear with the dioptric camera main axis, and the mirror lobes have a profile radially symmetric around this axis. Basu, this author and others originally developed such a system and shown real-time applications in processing the imagery from such a sensor [13]. Ollis [71] recently investigated various configurations for achieving such a view.

A sketch of the geometry of a bi-lobed mirror and two example systems built in the course of this thesis research is shown below in Fig. 1.3.

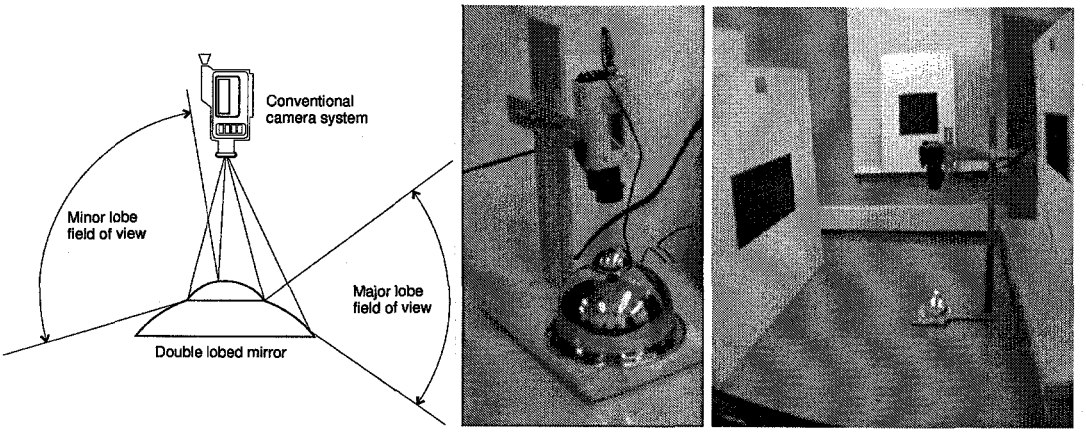


Figure 1.3: *Panoramic Stereo Image Sensor using concentric double convex lobed mirror.*

One of the two main thrusts of this thesis research is 3D model generation with this sensor. An example image captured by the high resolution sensor in Fig. 1.3 (right image), and the reconstructed model is shown in Fig. 1.4.

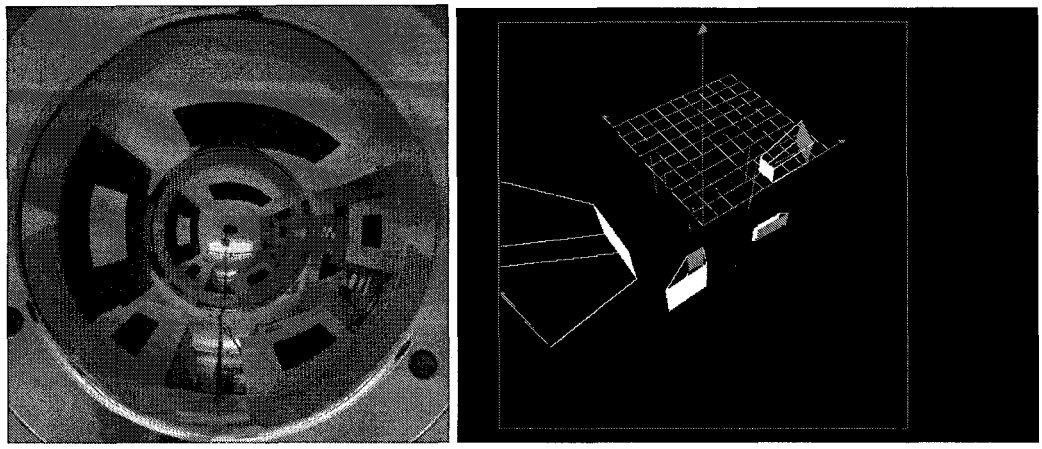


Figure 1.4: *Example of stereo reconstruction using the bi-lobed panoramic image sensor. (Left) Captured image, (Right) Reconstructed model compared to correct model. Correct polygons are shown in white, automatically reconstructed ones in grey.*

1.2.1 Survey of Panoramic Stereo

Several methods exist using multiple panoramic cameras, such as Sogo's and Ishiguro's [104] work where single lobed catadioptric panoramic cameras are positioned throughout a room, intending to track people walking in between. The sensors are arranged with parallel but not collinear central axis, for tracking moving objects as opposed to reconstructing scenes. Images are subtracted from the background, and the centers of the differences seen from the multiple viewpoints are triangulated. More than two cameras are used to eliminate false matching when more than two targets are in the environment.

Ishiguro et al. create a similar system to the above using imagery from rotating cameras [51, 52], but use only two cameras and attempt to reconstruct a depth model through correlation rather than tracking objects.

Peleg et al. [36] create panoramas from an off-center rotating camera, and reorder the image columns to form a family of panoramic images taken from effectively different viewpoints. They choose different virtual panorama with varying baselines from this set to get different disparities, since smaller baselines yield less disparity and easier matching whereas a longer baseline yields better reconstruction accuracy. Shum and Szeliski [101] demonstrate a similar system with reconstruction performed from various viewpoints.

Benosman and Devars [15, 96] rotate two linear image sensor elements around an axis that contains the focal points to create two cylindrical projection images. The intensity and edge images are correlated to produce a 3D range map to assumed vertical faces.

Shimamura developed an example of a multi-camera stereo catadioptric system with two vertically arranged assemblies of 6-sided pyramidal mirrors [62].

Gluckman, Nayar and Thoresz demonstrate panoramic stereo reconstruction with two

panoramic cameras arranged along a vertical, collinear axis [61, 83]. This is similar to the stereo work done in this thesis, except that two separate cameras are used. This of course yields better resolution and a greater baseline, however it uses two sensors. Gluckman demonstrates a near real-time system whereby a composite image combining two cylindrical projections, one from each camera, is obtained through warping the camera image so that the images have the same azimuth angle as the horizontal coordinate, and are aligned so that the azimuth angles line up. Correlation matching is then done to produce a depth map [61].

Correlation matching for stereo depth with catadioptric panoramic images at the same collinear central axis is also investigated by Ollis, Herman and Singh [71]. This work analyzes various configurations for two viewpoint stereo with separate sensors and a bi-lobed single sensor, using the non-SVP equi-angular profiles of Chahl and Srinivasan [24, 25, 26]. Ollis et al. perform synthetic image tests with correlation matching to determine error estimates for the different configurations. As one would expect, having two separate panoramic sensors (along a collinear axis), separated by as much height as possible, creates the best stereo reconstruction accuracy.

All these panoramic stereo vision systems have different applications and benefits. The application motivating the work of this thesis is the development of a system that is simple and compact, capturing the scene in one image rather than waiting for a revolving mirror, utilizing only one dioptric element and permiting three-dimensional reconstruction.

1.3 Mobile Robot Localization

The feature extraction method devised for the stereo reconstruction contains a verification stage, whereby image features were *tracked* in the image to compare hypothetical image features against the actual acquired image.

The tracking procedure was extended to mobile robot localization with a single-lobed (non-stereo) panoramic sensor, whereby a mobile robot could have a non-SVP sensor mounted on top that would allow it find its position by tracking landmark features. The estimated position was used to hypothesize the location of image features, which were then tracked to update the robot's position. A prototype system was built, the robot and an image of the recovered position is shown below in Fig. 1.5.

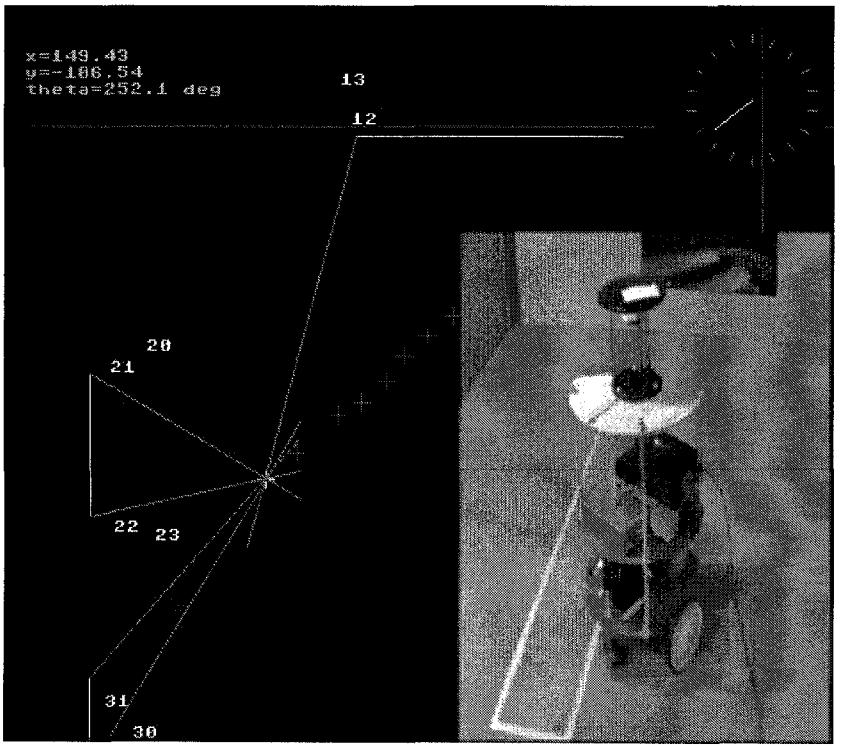


Figure 1.5: *Screen shot of prototype system showing triangulation of junctions in Fig. 9.17. The robot's trajectory is drawn by crosses indicating the position at previous frames. In this case the robot is translating towards the lower left.*

1.4 Thesis Objectives and Direction

The motivation for this thesis was to create a vision system that can be useful to automatic systems operating in a polyhedral world. The focus changed slightly to create this system using the newer family of panoramic catadioptric image sensors in place of the conventional narrow field of view perspective cameras. This further developed into panoramic catadioptric image sensors that had a general radial profile since many sensors fell into a category that the current literature did not address, the category being *non-SVP*, or *non-central* catadioptric optical sensors. The research direction was initially intended for stereo reconstruction only but the vision processing techniques were found to apply to mobile robot localization and to shape-through-motion. Furthermore, the mobile robot localization system is claimed to be the most advanced passive panoramic feature-based navigation system in the literature at the time of this writing.

It is believed by the author that both an analysis of panoramic image sensors and the development of a panoramic vision system was achieved in this thesis.

Chapter 2

Panoramic Hough Transform

The basic tool used in this work is the *Panoramic Hough Transform* (PHT) which is a process for finding the curved projections of horizontal lines in panoramic images. Unlike other approaches for computer vision using panoramic optics, this frees the design of the mirror component from the restrictions of having a parabolic or hyperbolic radial profile. This opens the field for using all convex, radially symmetric mirror profiles in panoramic computer vision applications.

The process is a Hough transform based approach, by which a parameter space image is formed. Cluster peaks in this image indicate the presence of horizontal line projections in the original image.

The PHT in itself is useful only for the detection of infinitely long lines, but when combined with the *Identify and Remove* algorithm of Chapter 3 provides a useful image processing system for automatically detecting the presence of finite edge segments. Together with the vertical line detection also presented, the most salient features in a man made environment can be found with a mirror of any convex radial profile.

2.1 Dioptric, Catoptric and Catadioptric Imaging Systems

Imaging systems, either biological or man made, use the basic components of pinholes, lenses and mirrors to form images on a photosensitive two-dimensional surface. An imaging system that uses only lenses to capture light is a *dioptric* system. Ones that use mirror elements only, such as some telescopes, are *catoptric* systems. One that utilizes both lenses and mirrors in the image forming process is a *catadioptric* imaging system.

Panoramic image capture devices are those that can capture light from a 360° field of view, and have advantages for computer vision systems. Computer vision applications of panoramic imaging include navigation, object tracking and world model creation. Indeed much of the general concepts in computer vision research, including structure-from-motion and stereo reconstruction, can be extended to these sensors with useful applications.

Capturing a panoramic (360°) field of view with dioptric optical systems such as standard narrow field of view (less than 90° field of view) cameras requires many such cameras. An imaging system that utilizes a mirror can be combined with the conventional dioptric paradigm to produce a catadioptric system capable of capturing a (360°) field of view with only one image sensor.

Catadioptric imaging systems, those employing mirrors as well as lenses in the optical path, allow for the capture of a panoramic image on one or a group of planar image sensors. If only one image sensor is desired, a catadioptric arrangement with a curved mirror is the

most common way used in the literature to capture a panoramic view. Basu [14], Yagi [116] and others [18, 17] have demonstrated such systems. Many researchers have participated in this research in recent years as evidenced by the number of researchers at *Omnivis 2000, 2001*, special forums on omnidirectional viewing held at the Computer Vision and Pattern Recognition (CVPR) conferences [70, 78, 21, 26, 62].

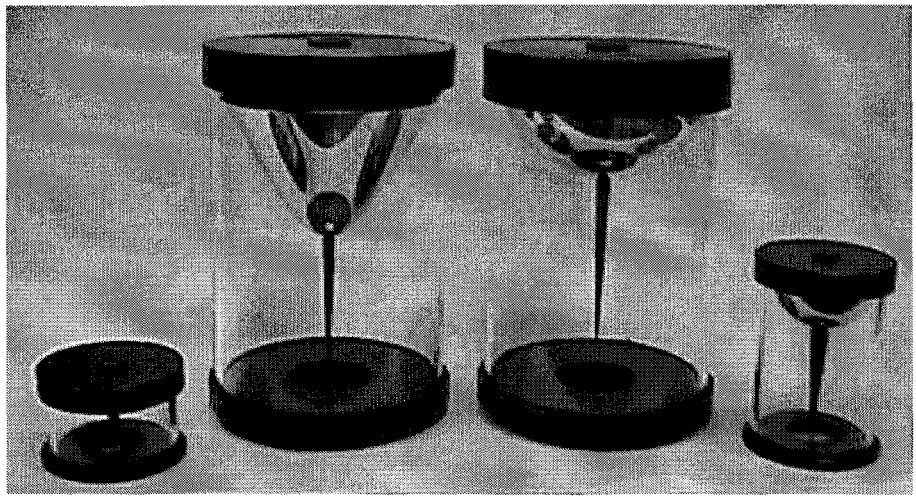


Figure 2.1: *Examples of mirrors to attach to regular cameras to create catadioptric panoramic sensors.*

This thesis considers only catadioptric mirrors with a *radially symmetric* mirror aligned with the central axis of the dioptric camera component. The dioptric camera main axis is the axis perpendicular to the image plane that passes through the focal point. The mirror in all the cases considered can be represented by revolving a two-dimensional profile around this axis. For example a spherical mirror has a circular profile.

2.2 The Single Viewpoint (SVP) Criteria

In the classic perspective pinhole model of standard narrow field-of-view optics, the nature of the projection allows a straight line three dimensional feature to appear as a straight line or as a point in the captured image. This perspective projection model requires that all light rays pass through a real or virtual convergence point, the so called *Single Viewpoint* (SVP). This is most commonly referred to as the *focal point* in dioptric systems, with the distance from this point to the imaging plane known as the *focal length.*

Straight lines and edges in the environment are a principle feature for vision systems to detect and one major advantage of a pinhole camera model is the maintenance of collinearity of edge points on the image plane. Without satisfaction of the SVP criteria the benefits of

a pinhole camera model, such as detecting straight lines, cannot be employed. A challenge posed by non-SVP sensors is that straight lines are more difficult to detect. Since straight line edges are salient features in man made environments where many robots will function, this shortcoming needs to be addressed to make such non-SVP systems useful in practice.

2.2.1 SVP Panoramic Imagery With Planar Mirrors

A SVP panoramic camera was designed and built by the author for an industrial partner of the University of Alberta which achieved a hemispherical field of view with planar mirrors. Fig. 2.2 demonstrates the geometry, that of an inverted pyramid of four planar mirrors. Five CCD image sensors are integrated to form an image covering the hemisphere. A central CCD captures a view upwards directly, and the other four capture reflections from the outside of the pyramidal mirror. In this way the system is a mixed dioptric/catadioptric camera, the inner camera captures light directly with a lens only and the other four view light reflected from the mirrored outside of the pyramid.

The proportions are chosen such that all five CCD's capture light from the same effective focal point, a true SVP system. The distance from each of the surrounding CCD/lens assemblies to the mirror is equal to the distance from the mirror surface to the focal point of the central CCD/lens dioptric camera. The author also designed and built processing electronics to combine five digital video streams from the CCD's into one panoramic video representation.

The system offered the benefit of an SVP system in that virtual views could be generated that were proper perspective projections. Also the view was more than panoramic, the "sky" was visible in the image too, which is not true for most panoramic systems, including the ones used in this thesis. The principle disadvantage was that image stitching had to occur, and there were inescapable intensity differences between the component views. Even if the camera views were perfectly matched, the different CCD's would have slightly different responses, especially in colour hue. Also each CCD had its own automatic gain control such that the intensity value of an image pixel of a 3D point depended on the total light impinging on that CCD, this produced the phenomenon of seeing one half of an object which crossed the stitching boundary go dark as a bright light source fell on one of the CCD's.

Similar systems were developed by Nayar around the same time (1996), and later by Shinamura et al. in 2000. Shinaruma's system [62] used two SVP assemblies of pyramidal mirrors, each with 6 sides, to create a stereo vision system for reconstructing textured meshes obtained by correlation matching.

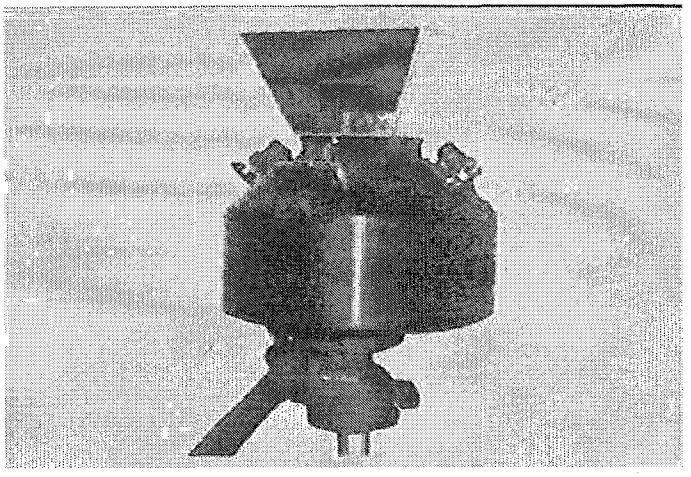


Figure 2.2: *A catadioptric imaging system consists of four planar mirrors, designed and built by the author for an immersive panoramic viewing system. Five CCD image sensors are arranged so their effective focal points are coincident.*

2.2.2 Panoramic Imagery With and Without the Single Viewpoint Criteria

Panoramic viewing systems can be constructed with a single rounded mirror and a single image sensor, indeed can even be stereo as developed in this thesis. The choice of mirror profiles can permit SVP geometry as detailed below, but it is possible to depart from the SVP criteria all together.

Baker and Nayar [10, 9] demonstrate that an effective single viewpoint (SVP criteria) with panoramic view catadioptric systems with a single curved mirror can only be achieved with mirrors of parabolic or hyperbolic profile. The literature also refers to SVP catadioptric sensors as *Central Catadioptric Sensors* [48].

Parabolic mirrors are the most commonly used profiles that fit the SVP requirement. However the dioptric component should not be any dioptric camera, but an *orthographic* camera. An orthographic camera captures parallel rays, analogous to a perspective camera focused at infinity. This can be a camera with a large zoom positioned at a large distance from the mirror, or one with additional lenses to approach the qualities of an orthographic camera. Columbia University and Nayar own a patent [82] on this SVP system. Catadioptric systems based on the parabolic mirror operate on the principle that all light rays headed toward the parabola focus are redirected parallel, and hence captured by the orthographic camera. The geometry for this reflective property is the same as that of a concave parabolic mirror, used extensively in flashlights, satellite dishes where the task is to reflect light from a focal point to exit parallel, or reflect incoming parallel rays onto a focal point. The difference is that the mirror is convex, and rays heading towards a focal point are directed parallel

to the parabola main axis. In a similar fashion, mirrors with a hyperbolic profile have one focus of the hyperbola inside the mirror, light headed towards this effective focal point is redirected to focus on the other hyperbola focus, where the dioptric camera focal point is positioned.

A private company is marketing a panoramic SVP system, this "Omnicam" is shown in Fig. 2.3 (left). Note the large size of the system compared to the mirror, also the bulky telecentric lens mounted to allow capture of parallel rays, to approximate an orthographic projection. Other implementations with a parabolic mirror and telecentric lens are shown in DARPA Nayar and Boult's progress report [83].

An SVP image sensor using a hyperbolic mirror has been developed at the *Vista* lab at CMU, shown in Fig. 2.3 (right). One application at CMU was with robot localization [111]. Another mobile robot position finding approach using hyperbolic systems with an image-based matching approach was shown by Krose [8].

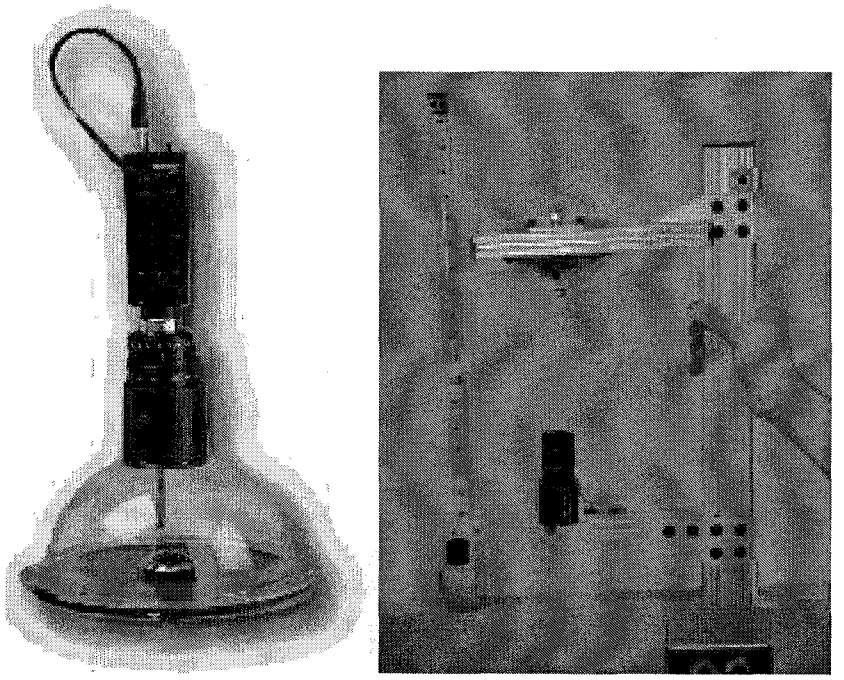


Figure 2.3: *The* Omnicam *(left) Commercial version of an SVP catadioptric system using a parabolic camera. (see http://comet.ctr.columbia.edu/ laitee/NewsLab/Omnicam). Hyperbolic SVP implementation (right) from the* Vista *project at CMU.*

The profiles designed by Hicks [55] are non-SVP, but do maintain the straightness for lines on a plane perpendicular to the central axis. The lines at elevations well below the horizon are reasonably maintained such that they can be detected by conventional Hough transform line finding techniques. This was used by a Portuguese team in the *Robocup* contest to localize the robots by looking down at the lines on the field [6, 73]. Hicks designed profiles for both telecentric (for capturing near-orthographic views, i.e. parallel light rays)

and standard short focal length cameras. These mirror profile are unfortunately not useful for the general case of finding polyhedral objects around an image sensor, for they do not really capture a panoramic view.

Mirrors of parabolic and hyperbolic profile can produce images that can be warped into equivalent pinhole perspective projections since the light rays incident on the camera image plane have a virtual single viewpoint. Thus the image captured can be converted into pseudo-perspective projections corresponding to views seen by equivalent regular pinhole cameras. This cannot be said for other profiles. Each point in the image taken from a spherical or other non single-virtual-perspective (SVP) point corresponds to a virtual ray emanating from a different point.

Many profiles however do not have this SVP property but are desirable for other reasons. Several profiles other than parabolic or hyperbolic have practical advantages that cause them to be considered such as better utilization of the imaging surface and more compact and practical physical implementation. Non-SVP profiles include spherical mirrors, which are the most amenable mirror profile for manufacturing, the equal solid angle resolution mirror profiles proposed by Conroy and Moore [30], the linear mapping profiles of Chahl and Srinivasan [25, 24] and results of Derrien's iso-angular profile design [32].

The theory developed in this chapter is intended for single lobed panoramic sensors that are not necessarily of a parabolic or hyperbolic profile, or panoramic stereo sensors of such lobe profiles. The PHT allows straight line detection in these non-SVP scenarios where such features do not project to a set of collinear points in the image. The theory and transform presented is valid for all non-SVP radially symmetric mirror lobe profiles, but the specific formulae presented and experiments performed in this thesis focus on lobes of a spherical curvature.

2.2.3 Recovery of Straight Lines in Single Viewpoint (Central) Panoramic Catadioptric Image Sensors

Baker and Nayar [10] impose a constraint on mirror design of maintaining a single virtual convergence point for rays impinging on the mirror. The presence of a single viewpoint, was shown by them to exist only in assemblies of planar mirrors or rounded mirrors of parabolic or hyperbolic radial profile.

Using an arrangement of planar mirrors to achieve a SVP will require multiple cameras. Using a single camera to capture a panoramic scene with a SVP requires the other option of a curved mirror of parabolic or hyperbolic radial profile. With an SVP system, all rays incident on the imaging plane correspond to rays that intersect at this SVP. Thus even though the image has curved projection of straight-lined features, it is possible to extract

virtual viewpoints that a classic pinhole camera would see if its focal point was coincident with this SVP.

Thus to automatically find straight line features, one can warp the captured image to provide several equivalent perspective views in different directions and use the theory and algorithms designed to find straight lines in standard narrow field-of-view computer vision such as the classic Hough transform for straight lines.

An alternative to warping the image is to use geometric properties to find straight line projections directly in the source image. The catadioptric projective property described by Daniilidis [48, 49] causes straight lines to project to circular arcs if the mirror profile is parabolic. Indeed calibration techniques have been developed using this principle, Geyer and Daniilidis [46, 48, 47] demonstrate this to calibrate a camera of parabolic profile.

A similar method for detecting straight lines with SVP catadioptric stereo systems is [65] where a hyperbolic mirror profile is used, and the geometry causes lines to be projected as circular arcs, where circle finding routines can be used.

2.2.4 Recovery of Straight Lines in non-SVP Panoramic Systems

With the exception of Hicks' [55] profiles, the detection of straight lines is problematic in non-SVP imaging systems. The profiles of Hicks are non-SVP and can maintain line straightness, but do not provide a panoramic view.

For the general case of a truly panoramic catadioptric image sensor with a mirror of radial symmetry, the only straight lines in the scene that project onto straight lines in the image are those parallel to the main axis. The main axis includes the axis about which the mirror is symmetric and the dioptric camera's focal point. If this axis is vertical, then only vertical lines will be captured as straight lines in the image.

For a vision system operating in a man made environment, recognizing both vertical and horizontal lines is important, since they constitute the majority of features within a building. A horizontal straight line edge projects to a complex curve in the image, depending on the radial profile. If the mirror profile is parabolic, and the dioptric camera stereographic, this curve will be circular (the SVP case). For the general case this projection is curved, but not circular.

The procedure developed limits the search for non-vertical lines to just horizontal ones. The problem of recognizing and modeling horizontal line edges in panoramic imagery can be reduced to identifying projections of planes onto each mirror lobe image. If the mirrors are of a non-SVP profile, such as a sphere, extracting virtual viewpoint images in which to search for straight lines is not possible. This chapter proposes a method to handle these more complex cases.

2.3 Panoramic Hough Transform for Finding Lines in non-SVP (non-central) Panoramic Images

A ramification with building a panoramic catadioptric image sensor with a mirror which is neither planar, parabolic nor hyperbolic is that the image captured on the image plane is a complex curve not definable with standard geometric primitives such as circles, ellipses, etc.

Much computer vision research is performed with non-central (non-SVP) panoramic systems. Conical mirrors have been used by several researchers[14], spherical mirror implementations abound (Basu, Baldwin and this author [31, 11, 13]), Derrien and Konolige [32]. Mirrors of other profiles have been designed to control angular resolution at different elevations, such as the work by Chahl and Srinivasan [25, 26]. Chahl's profile provides a linear relationship between an image point's radius and the 3D scene point's elevation angle. Conroy and Moore [30] have designed profiles which equalize the solid angle (in steradians) that a pixels subtends). However all of the above fall into the category of non-central, non-SVP catadioptric systems. They all have mirrors with radial symmetry about the main camera-mirror axis, but none have an effective focal point.

A new method and theory is proposed to locate horizontal line features with non-SVP catadioptric sensors. By mapping edge pixels to a new two-dimensional parameter space for each mirror lobe, the existence and location of horizontal lines can be found. The motivation for recognizing horizontal and vertical line segments is for its use in three-dimensional model creation or single view landmark tracking in man-made environments where the majority of line edge features are either horizontal or vertical. For example, mobile robots using vision to navigate within a building could use this panoramic stereo sensor and detect features using the Panoramic Hough transform.

The well known Hough transform is a method to identify lines or curves by creating a separate parameter space, and finds groupings in this parameter space to define lines and curves. Each point in the original image is projected onto a set of points corresponding to possible lines or curves in the parameter space. The results are accumulated over all the original points and resulting peaks in the parameter space are identified. The classic straight line finding Hough transform creates a parameter space for the two dimensions which a line can have, for example m, b for the line equation $y = m \cdot x + b$. Many applications of this can be found such as Venkateswar [112], and extensions to other geometric entities as circles [95].

A new Hough transform parameter space is proposed for detecting horizontal lines in images formed from vertically posed non-SVP catadioptric panoramic image sensors. Each point in this new parameter space corresponds to a plane in space. The family of planes

containing horizontal lines which project to a curve of zero width on the image plane is a two-dimensional subset of the three degrees of freedom an unconstrained plane has in three-dimensional space. Hence horizontal lines are viewed by a panoramic catadioptric sensor as a family of curved lines with two two degrees of freedom. Thus this *Panoramic Hough Transform* space is two-dimensional. This parameter space is referred below as the *Panoramic Hough* space.

2.3.1 Finding Projections of Horizontal Lines in Panoramic Imagery

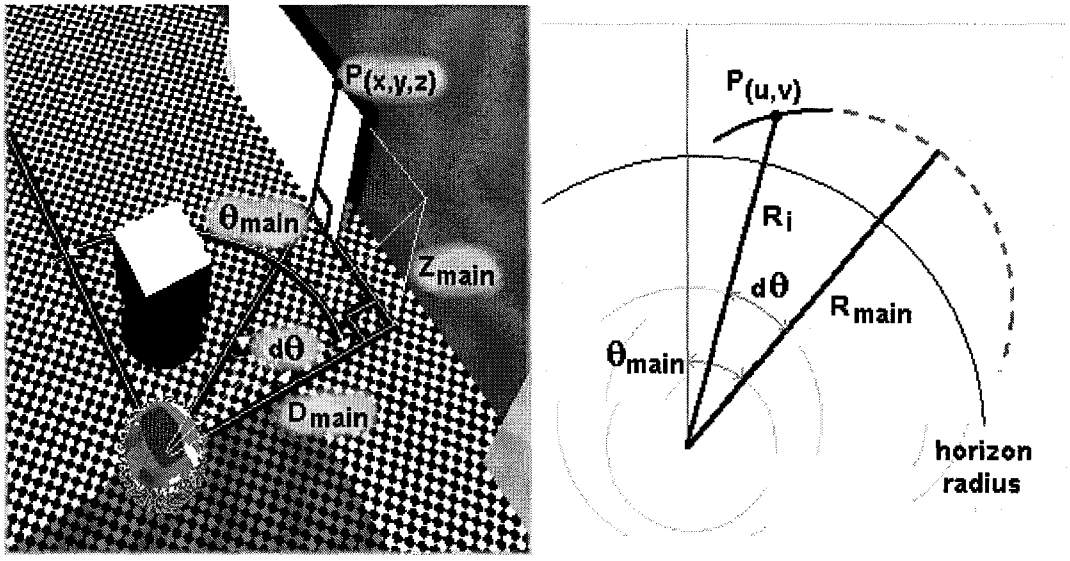


Figure 2.4: *Basics of Panoramic Hough Transform. (Left): Scene point $P(x, y, z)$ can be represented by angle $d\theta$ along a horizontal line defined by θ_{main}, D_{main} and Z_{main}. (Right): The projection on to the image plane can be represented by $(R_i, \theta_{main} + d\theta)$.*

The Hough transform is extended to locate the projection of straight horizontal lines in the imagery provided by SVP and non-SVP catadioptric panoramic image sensors. The geometry is shown in Fig. 2.4.

The relative position of a scene point $P(x, y, z)$ to the camera can be expressed as lying along a horizontal line whose closest approach to the camera axis occurs at a direction of θ_{main}. This line is defined by the direction θ_{main}, the distance D_{main} and height Z_{main}. $P(x, y, z)$ can be defined by a 4^{th} parameter $d\theta$ relative to θ_{main}. This is shown in Fig. 2.4 (left).

The point $P(x, y, z)$ belongs to a horizontal line which needs three parameters, θ_{main}, the distance D_{main} and height Z_{main}. This line projects on to a curved line on the image plane defined by θ_{main} and R_{main}. Due to the loss of depth information, a single parameter

R_{main} is a function of D_{main} and height Z_{main}. This function depends on the mirror profile, the focal length and the distance from the lens center to the mirror. R_{main} and θ_{main} are enough to define the projection of the horizontal line containing $P(x, y, z)$. Clearly the 3D line itself cannot be defined from this alone, only a plane containing the line can be defined. Depth information, if required, can be obtained by using two viewpoints given by the two mirror lobes.

A three dimensional horizontal line can be represented uniquely by θ_{main}, the distance D_{main} and height Z_{main}. A three dimensional point can be defined by four parameters, θ_{main}, the distance D_{main}, height Z_{main} and $d\theta$, however this is not unique. If there is no knowledge of which line a point is considered to belong to, then there is a one dimensional degree of freedom in θ_{main} and $d\theta$. In the same way there is a one dimensional degree of freedom between the projection curves this 3D point corresponds to on the image plane. The task of finding lines from a set of points can be expressed as finding the set of all lines each point can belong to, then finding the line common to all these sets. This is the essence of Hough transform based approaches.

Fig. 2.4(right) shows how a scene point $P(x, y, z)$ will appear in the image plane as $P(u, v)$. θ_{main} and $d\theta$ are preserved in the projection.

R_{main} and θ_{main} are the X and Y axis of the Panoramic Hough (PH) Transform parameter space. A single source image point maps to a loci of transform space points corresponding to the family of horizontal line projections which it may belong to. Conversely, one point in the parameter space corresponds to a curved family of points in the image space. This is similar to the point-line duality in the classic Hough transform, shown graphically in 2.7.

Eqns. 2.1 and 10.1 express the forward and reverse Panoramic Hough Transform respectively.

$$R_{main} = PH(R_i, d\theta) \tag{2.1}$$

$$\theta_{main} = \theta_i + d\theta$$

$$R_i = PH^{-1}(R_{main}, d\theta) \tag{2.2}$$

If the PH-Transform is being used for feature extraction within an image [40], $(R_{main}, \theta_{main})$ is calculated and plotted in the transform space for the range $(-\pi/2 < d\theta < \pi/2)$. If θ_{main} is expected to lie within a bounded range $\theta_{min} - \theta_{max}$, $(R_{main}, \theta_{main})$ is only calculated and plotted for points where $\theta_{min} < \theta_i + d\theta < \theta_{max}$. This can be conceptually described as finding all the possible projection curves an image point can belong to.

A set of selected image plane points are determined (shown as crosses in Fig. 2.5 (left), and for each source point, the curved loci of points $(R_{main}, \theta_{main})$ is plotted in the parameter space. The magnitude of value added to these mapped points can be unity or proportional

to the edge strength. After aggregating the mapping of all source points (Fig. 2.5 (left)), the transform space image is searched for peaks. If a peak is detected (Fig. 2.5 right), then the set of input points (R_i, θ_i) fits the projection of a horizontal line defined by $(R_{main}, \theta_{main})$.

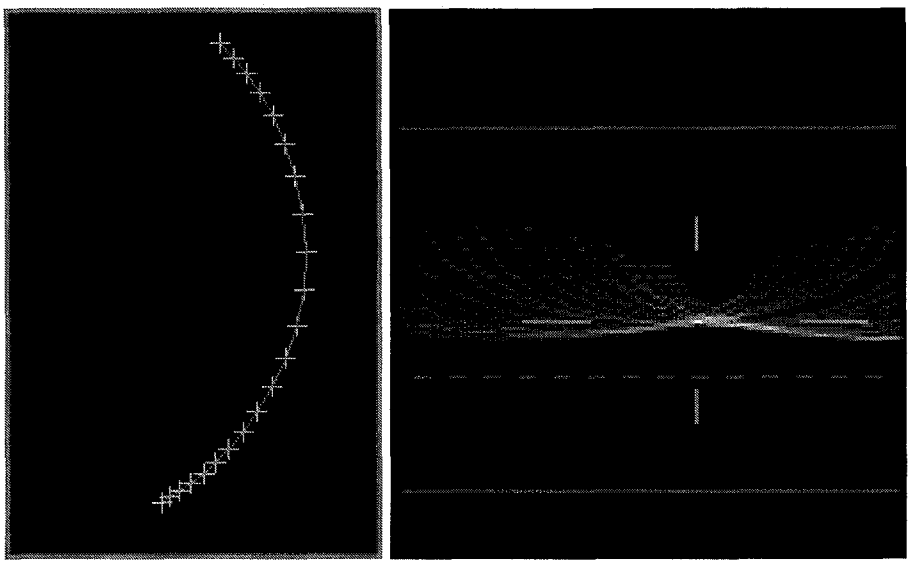


Figure 2.5: *(Left). A set of points in the source image. Each point maps to a set of points in the PH hough image (right). If a cluster is found in the PH image (shown with cross-hairs), then the source image points can lie upon a horizontal line defined by* R_{main} *and* θ_{main}.

2.3.2 Panoramic Hough Transform: Further Detail

A plane in three dimensional space relative to a given vertical axis can be described by a special line lying on this plane, referred to herein as the *fall line.* The fall line is the direction of greatest slope and is perpendicular to horizontal lines lying in this plane. The plane can be determined from a fall line by noting it contains both the fall line, and the cross product of the fall line with the axis. Three parameters uniquely define the fall line, the height Z_{fall} of the intersection with the vertical axis, and the azimuth θ and elevation ϕ_{fall} angles. Since only a two dimensional set of the three dimensional space of planes can project onto a curve of zero width on the image plane, this can be reduced to the two angle parameters. Two parameters θ_{fall}, ϕ_{fall} are sufficient to describe this line and thus the corresponding plane (the fall line is the elevation angle of the incoming ray in Figure 2.4).

Further insights can be gained by considering the line perpendicular to the 3D horizontal line, exists in the described plane, and passes through the main (vertical) axis. This *fall line* is that line which passes through the intersection point of the given axis and the plane, and forms the minimum angle relative to the axis. In this discussion, this axis is the vertical axis of the panoramic stereo image sensor, and the fall line is the line of steepest descent

(analogous to the *fall line* in surveying corresponding to the direction of steepest descent of a slope).

A plane described contains horizontal lines perpendicular to the fall line at different distances from the axis. The cross product of the axis direction and the fall line vector is the horizontal line of this set at zero distance. Being horizontal, its elevation is zero. All lines in the plane have a direction vector that can be defined as a linear combination of the fall line vector V_{fall} and this zero distance horizontal line L_{zero}. Lines of interest $L_{intersect}$ are those that intersect the axis, their azimuth is $\phi_{fall} + d\theta$. The coefficients of this linear combination can be the sine and cosine of $d\theta$. In this way the elevation $\phi_{intersect}$ of these lines of interest can determined as in Eqn. 2.3 as a function of $d\theta$ and ϕ_{fall}. In Eqn. 2.3, the main axis defines the Z axis, and the azimuth direction of the fall line defines the X axis for convenience.

$$V_{fall} = \begin{bmatrix} cos(\phi_{fall}) \\ 0 \\ sin(\phi_{fall}) \end{bmatrix} \tag{2.3}$$

$$L_{zero}directionvector = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$L_{intersect} = cos(d\theta) \cdot V_{fall} + sin(d\theta) \cdot L_{zero}$$

$$tan(\phi_{intersect}) = \frac{sin(d\theta)}{\sqrt{(cos(\phi_{fall}cos(d\theta)^2 + sin^2(d\theta)}}$$

One important outcome is that at $d\theta = \pm 90°$, $\phi_{intersect} = 0°$, regardless of ϕ_{fall}. This defines two vanishing points for the projection of all horizontal lines of a given θ_{main}. Fig. 2.6 demonstrates the projections of planes of varying ϕ_{fall} with a constant θ_{main} and these two vanishing points can be seen on the *horizon line.*

The two-dimensional slice of the critical geometry shown in Fig. 2.4 is a vertical plane containing both the central axis and this fall line.

The azimuth angle θ_{fall} and elevation angle ϕ_{fall} of this fall line are sufficient to describe the plane of points in three-dimensional space which project onto a fixed (curved line) loci of points on the image plane. For convenience, the ϕ_{fall} angle of the plane can be represented in the more convenient measure as that of the radius corresponding to the point which the fall line projects to on the image plane.

The application of the above derivation to identifying horizontal lines is that the projection of this line will lie along a unique curved line of pixels. If the presence of a sufficient number of points along this locus is found, then the existence of a horizontal line edge along a plane can be assumed.

A few mappings between the image and Panoramic Hough space give some insights, a circular line in the image, centered around the mirror center, corresponds to a conical loci

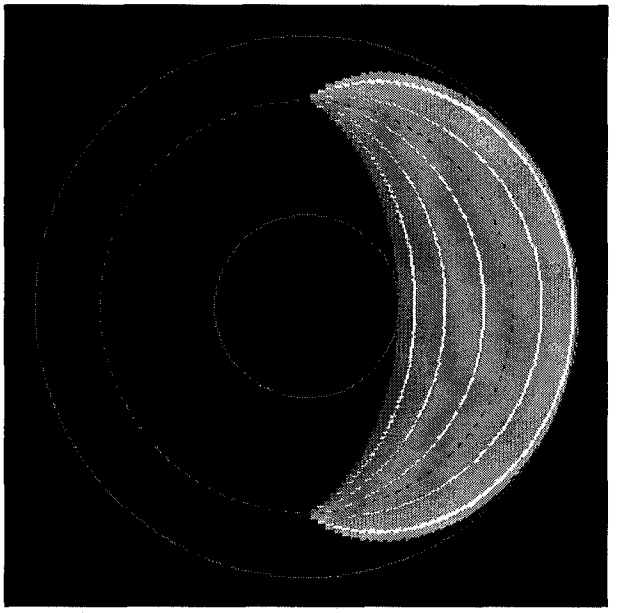


Figure 2.6: *Area of mirror image covered by lookup table for a spherical mirror with some example loci highlighted corresponding to horizontal edges of different heights.*

of points in 3D space. As this circle is increased in radius, the conical loci gets progressively more steep. At one circular line, this cone is actually flat and thus corresponds to the horizon. This horizon line is shown as a dotted circle in the image plots, and as a horizontal dotted line in the Panoramic Hough space plots. Also of interest is the minimum and maximum useful radii for each lobe which are plotted as solid circular bounding lines in the image plots, and solid horizontal lines in the Panoramic Hough space plots.

Figure 2.7 below demonstrates the point-line duality of the Panoramic Hough Transform of a spherical mirror.

The equations governing the projection of horizontal line onto the image plane is as follows: Since the *fall line* intersects the camera's main axis, this line and axis form a plane P_{main}. Since the mirror is radially symmetric, a point in space $P_w(X, Y, Z)$, where it reflects off the mirror $M_w(X, Y, Z)$, and the projection (u, v) onto the image plane all lie on on plane P_{main}. Hence the geometry can be reduced to a two dimensional case for analysis as is shown in Figure 2.8.

Plane P_{main} is redrawn for the current analysis in Figure 2.8. The main camera axis, assumed to be vertical, defines the $Y-axis$, and the $X-axis$ is the line perpendicular to the camera axis lying on P_{main} that passes through a convenient point in describing the mirror profile (the center for the spherical mirror case). World point $P_w(X, Y, Z)$ is represented by p with coordinates (p_x, p_y), and the point of reflection on the mirror is m with coordinates (m_x, m_y). The camera is located at $y = f_y$ above the mirror. The mirror profile for the general case is represented by a function involving x and y, Eqn. 2.4. Examples of several

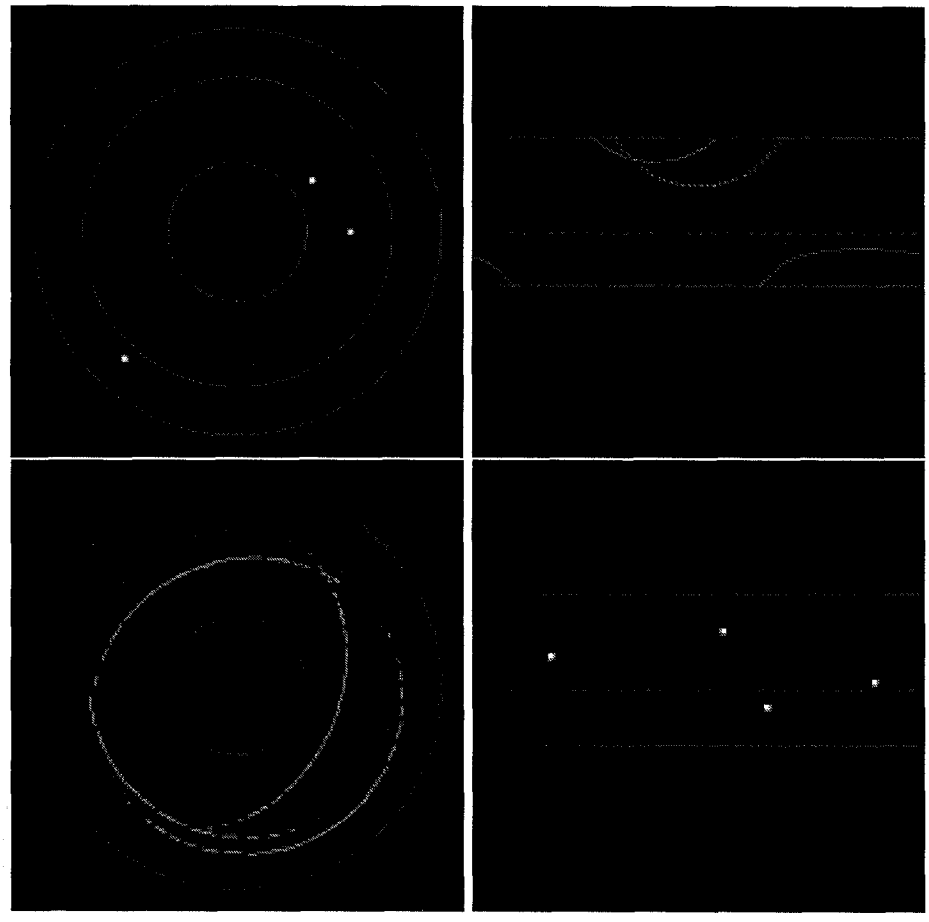


Figure 2.7: *Demonstration of point/line duality of the Panoramic Hough Transform. Mapping points from image space (top left image) to hough space (top right image) - a point in image space becomes a curved line in the Panoramic Hough space. Likewise mapping points backwards from hough space (bottom right image), a point in hough space corresponds to a curved line in image space (bottom left image). The horizontal axis in the two hough space images (top right and bottom right) is θ and the vertical axis is radius r, increasing top-to-bottom. The three circles in the image space images (top and bottom left) represent the minimum useful radius, the horizon radius, and the maximum usable radius (listed smaller to larger circles). These radii are drawn on the hough space images as three (horizontal) lines, the dotted line is the horizon radius.*

profile equations are given.

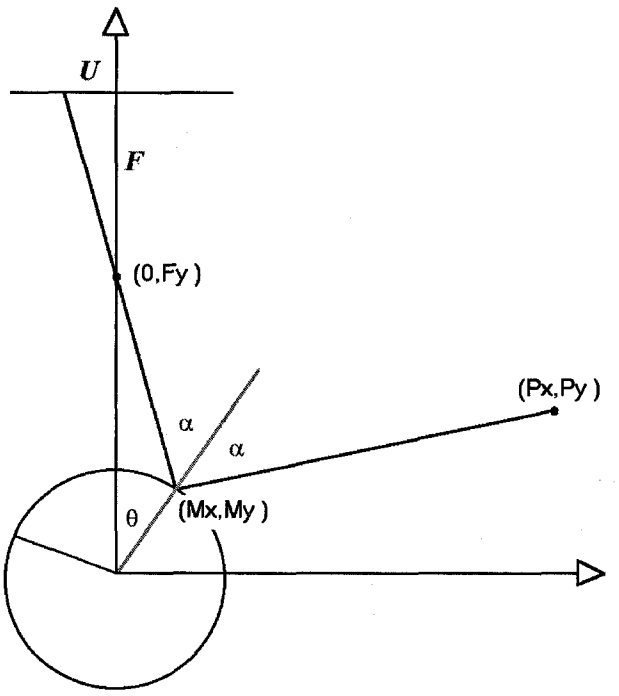


Figure 2.8: *Theory of Panoramic Hough Transform demonstrated for one mirror lobe. Modeling the projection of a point onto the image plane after reflection off of a spherical mirror of radius R.*

$$
\begin{aligned}
y &= F(x) && (2.4)\\
x^2 + y^2 &= r^2 \quad && CircularProfile\\
y &= a \cdot x^2 + b \quad && ParabolicProfile\\
a \cdot x^2 - b \cdot y^2 &= r^2 \quad && HyperbolicProfile
\end{aligned}
$$

Our derivation concentrates on spherical mirrors, whose normal vector at any point it given by Eqn. 2.5.

$$\vec{N} = \frac{1}{r}\begin{bmatrix} m_x \\ m_y \end{bmatrix} \tag{2.5}$$

The perfect reflection condition of the angle of incidence being equal to the angle of reflection (Eqn. 2.6).

$$\alpha_i = \alpha_r \tag{2.6}$$

These angles are calculated by the dot product between the unit direction vector from the point being imaged (p_x, p_y) (Eqn. 2.7) to the mirror point (m_x, m_y) and the unit direction vector from the focal point $(0, f_y)$ to this same mirror point (Eqn. 2.8). The corresponding point on the image plane is given by the simple perspective equation (Eqn. 2.13). Eqns. 2.4, 2.5, 2.6 and 2.7 through 2.13 are sufficient to constrain the geometry and provide one or two solutions (depending on the mirror profile) mapping a point in space to a point on the image plane.

$$\vec{V_i} = \frac{1}{((p_x - m_x)^2 + (p_y - m_y)^2} \begin{bmatrix} p_x - m_x \\ p_y - m_y \end{bmatrix} \tag{2.7}$$

$$\vec{V_r} = \frac{1}{(m_x^2 + (f_y - m_y)^2} \begin{bmatrix} -m_x \\ f_y - m_y \end{bmatrix} \tag{2.8}$$

$$MirrorNormalVector\vec{N} = \frac{1}{r} \begin{bmatrix} m_x \\ m_y \end{bmatrix} \tag{2.9}$$

$$cos(\alpha_i) = \vec{V_i} \cdot \vec{N} \tag{2.10}$$

$$cos(\alpha_r) = \vec{V_r} \cdot \vec{N} \tag{2.11}$$

$$\alpha_i = \alpha_r \tag{2.12}$$

$$\frac{u}{f_y} = \frac{m_x}{m_y - f_y} \tag{2.13}$$

The equation of the radial profile affects this mapping, a closed form for the projection of a horizontal line after reflecting off a spherical mirror as a function of a parametric line scalar could not be found. Numerical methods were required to find this mapping.

For each horizontal line whose projection onto the image plane was recognized to belong to a plane represented by fall line (θ, R_{fall}), the corresponding start point and direction vector to describe this fall line in 3D world coordinates can be found. Fig. 2.8 and Eqns. (2.7 - 2.13) describe the geometry as reduced to that on a 2-D plane, which contains the camera's main axis (typically vertical) and the closest point of the horizontal line to this main axis. I.e. Fig. 2.8 shows a cross-section of the 3-D geometry. This cross-section is at angle θ and the 3-D coordinates (with the Z-axis parallel to the camera's main axis) can be calculated simply as follows. The start point $S_{world}(X, Y, Z)$ and the direction vector $DIR_{world}(X, Y, Z)$ are given in Eqn. (2.14).

$$S_{world}(X, Y, Z) = \begin{bmatrix} m_x \cdot cos(\theta) \\ m_x \cdot sin(\theta) \\ m_y \end{bmatrix} \tag{2.14}$$

$$DIR_{world}(X, Y, Z) = \begin{bmatrix} (p_x - m_x) \cdot cos(\theta) \\ (p_x - m_x) \cdot sin(\theta) \\ (p_y - m_y)) \end{bmatrix}$$

Using this information from two viewpoints, such as reflected in the two lobes in a bi-lobed Panoramic Stereo Imaging System, the three- dimensional equation for this horizontal

line feature can be found. For convenience, the start point is moved back along DIR to the vertical axis so it can be represented by just a Z height value. To find horizontal line segments and not just line definitions, these end points can also be determined with these equations.

For practical application, this mapping would most likely be calculated beforehand and stored in a look-up table. For a spherical camera/mirror arrangement with one lobe, an example lookup table of ratio $D/R = 4.5$ is presented in Figure 2.6 (D=distance between camera focal point and mirror center, R=radius of spherical mirror). Each entry in the lookup table, in its minimal configuration, is the fall line radius R_{fall} indexed by the radius of a point and its angle from this fall line.

Each edge point in the captured image can lie along a curve of points in the Panoramic Hough space according to its θ azimuth angle from the fall line. Thus the simplest, but not most efficient, approach is to fill in all points in the Hough space for all angles $\pm 90°$, and recognize peaks in the Hough image as likely candidates for a corresponding θ, R_{fall} plane. While reasonable results can be achieved using only the edge magnitude and projecting all image edge points to this hough space, the noise rejection and processing time can be substantially improved by taking the image edge angle into account as shown in Chapter 4.

When a peak is detected in this Panoramic Hough space, the start point, and azimuth and elevation angles of the fall line is known. Since the fall line intersects the vertical camera's main axis, only the height is needed to define the start point. This height is given by the camera/mirror geometry and mirror profile. For parabolic and hyperbolic mirror profiles this point is constant, the SVP, but for spherical and other non-SVP profiles this value will vary.

The novel component of this work is the recognition of the projection of horizontal lines, but it is well complemented by recognizing vertical lines as well. The standard Hough transform for detecting vertical straight lines is not necessary since the projections will be of a known orientation. They project as straight radial lines, and many techniques can be used to find them. In Chapter 4 a method is shown whereby edge pixels whose direction are parallel to the radial direction are re-mapped to a quasi-cylindrical image, a rectangular image with axis representing radius and azimuth angle. This warped image is searched for connected components along columns warped from a single azimuth angle.

2.4 Experiments

Experiments with synthetic and real images were carried out, with the procedure for each mirror lobe being:

1-Generate a lookup table specific to the camera geometry. Specifically, the parameters are; D=Distance from Camera focal point to mirror center, R=Mirror Radius, F=Camera focal length (measured in pixels for convenience), coordinates U_{center}, V_{center} and R_{min}, R_{max} of the reflected image area were used to create a look-up table.

2-For each point in the camera image convert to polar coordinates relative to U_{center}, V_{center} for pixels in the range $R_{min} to R_{max}$.

3-Add the edge magnitude of each point in the image to all possible mapped points in the Hough image.

4-Locate peaks in this Panoramic Hough image and declare the existence of a plane corresponding to R_{fall} along which a horizontal line is suspected of lying.

Figs. 2.9, 2.10 and 2.11 demonstrate the technique described above to locate horizontal edges in synthetic images (generated with the POVRAY ray tracing software).

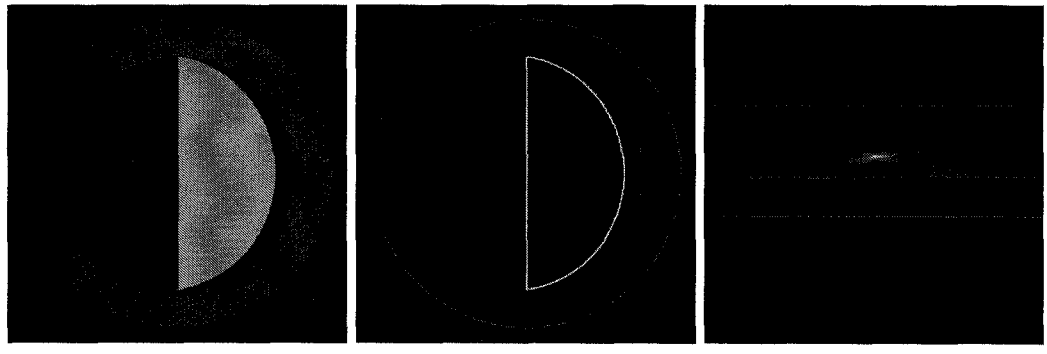


Figure 2.9: *A simulated image of a horizontal infinite edge, its edge magnitude image (using the Sobel edge mask pair) and the projection of the edge points to the hough space.*

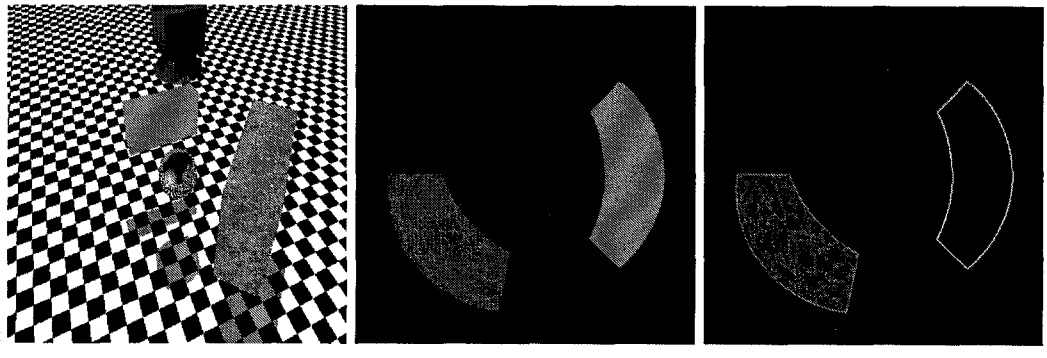


Figure 2.10: *A side view of a synthetic scene, the image seen by the spherical mirror view and the edge magnitude image.*

The detection of peaks in the Panoramic Hough parameter space was not done optimally in these experiments, yet very good results were obtained with the quite distinct peaks. In these images the response is shown as an intensity normalized to the maximum response.

Experiments were performed with real images (Figs. 2.12, 2.13 and 2.14, and results

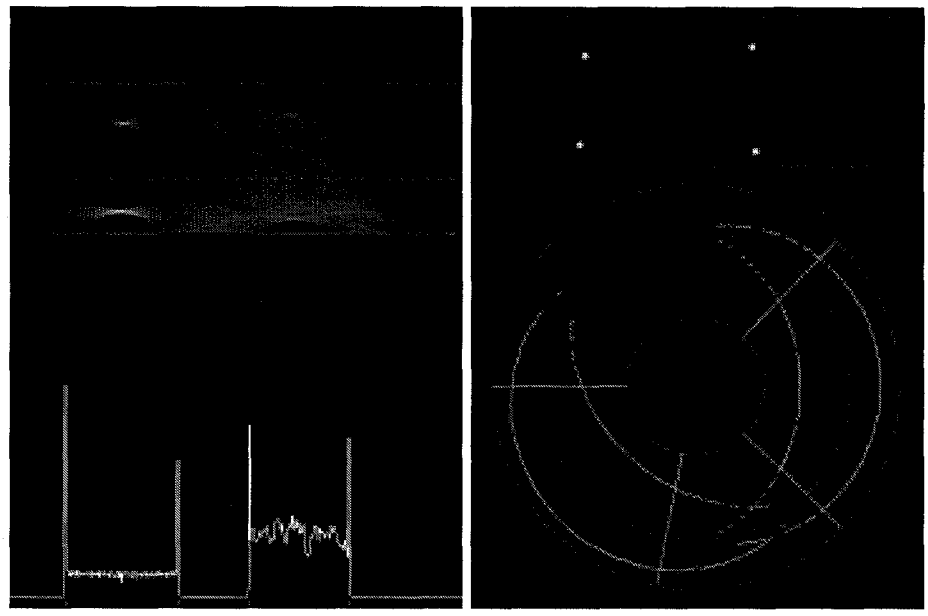


Figure 2.11: *Panoramic Hough projection of the image in Figure 2.10. Top Left: Panoramic Hough Transform Space. Top Right: Automatically detected cluster centers in this transform space. Botton Left: A histogram of edge pixels by angle allows detection of radial lines, which are projections of lines parallel to the main camera axis (assumed vertical). Bottom Right: Re-projection of corresponding points back to image plane loci for horizontal lines found from Panoramic Hough space, and vertical lines from peaks in the histogram.*

similar to the synthetic were obtained. The performance was based on qualitative analysis of horizontal lines detected.

2.5 Improvements to Basic Panoramic Hough Transform

Further extensions to the Panoramic Hough Transform are shown in Figure 2.14. Improvements of the technique used in the above experiments could entail finding connected groups of edge pixels and testing these groups separately to isolate horizontal lines of two types as shown. In the above section, the entire image was put into the Panoramic Hough space. No attempt was made to test independently connected edge groups, or to remove points from recognized peaks in the Hough image. Populating a subset of the points in Panoramic Hough space that correspond to a given radius by utilizing the edge angle will result in reduced computation and reduced noise added to other regions. The next step in the application of this technique would be to examine connectivity and range in azimuth angle of points identified as horizontal lines to provide line segment information and supplement the line equation with end points. Initial results of this are shown in Figure 2.14.

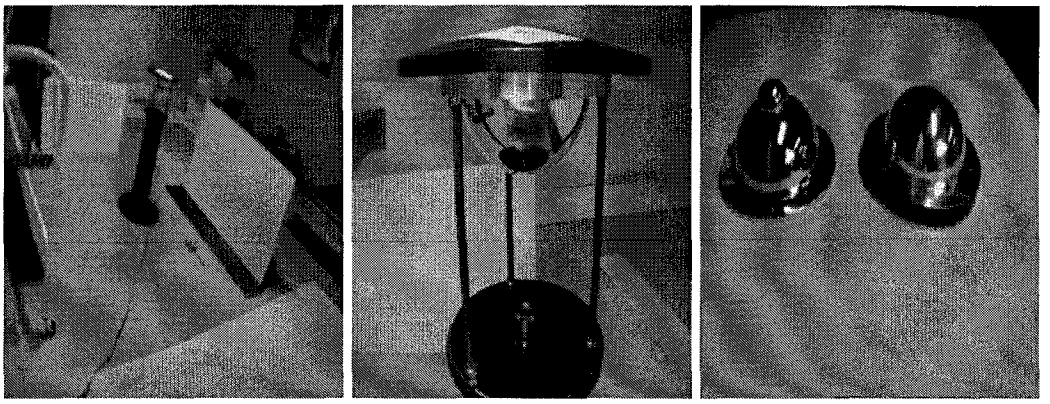


Figure 2.12: *Experimental apparatus for single and double lobed (stereo) experiments. A scene of horizontal and vertical rectified objects are imaged with both a single lobed mirror sensor. Future applications could use the Panoramic Hough Transform of a double-lobed mirror to achieve panoramic stereo reconstruction of horizontal line segments.*

2.6 Application to Panoramic Stereo Reconstruction

Traditional work on stereo computer vision systems used multiple narrow field of view cameras facing in similar directions [33, 66, 85]. Capturing a panoramic (360°) field of view would require a ring of narrow field of view cameras or motion of a single camera accumulating a panorama over several image frames. Likewise, capturing a panoramic stereo view would require many stereo camera pairs or a moving pair. Such systems are more expensive, complex and slow compared to a single-camera panoramic stereo system [14, 36, 51].

Stereo panoramic imaging can be accomplished with an extension of the catadioptric design with two concentric lobes at different distances from the camera [31]. Basu, Fiala and others have developed such a system and shown real-time applications [13, 11]. Such a design utilizes only one image sensor and a specially shaped, double-lobed mirror to achieve two panoramic views from different effective viewpoints achieving panoramic stereo imaging. (Figure 1.1).

The mirror in Figure 1.1 comprises two bi-convex lobes, a minor lobe embedded in a major lobe. The field of view of each is restricted by the geometry, but covers a large portion of a sphere. At most elevations a point in the environment is reflected in both lobes and is thus represented twice on the imaging plane of the camera. Since the object has been effectively imaged from two different positions in space, the essence of binocular imagery is present, and depth can be recovered. Ollis [71] investigated 3D reconstruction error of panoramic stereo systems of this type with synthetic image experiments.

If the stereo image sensor is placed with its main axis vertical, then straight vertical lines will project to straight radial lines, with their detection a solved problem. However straight

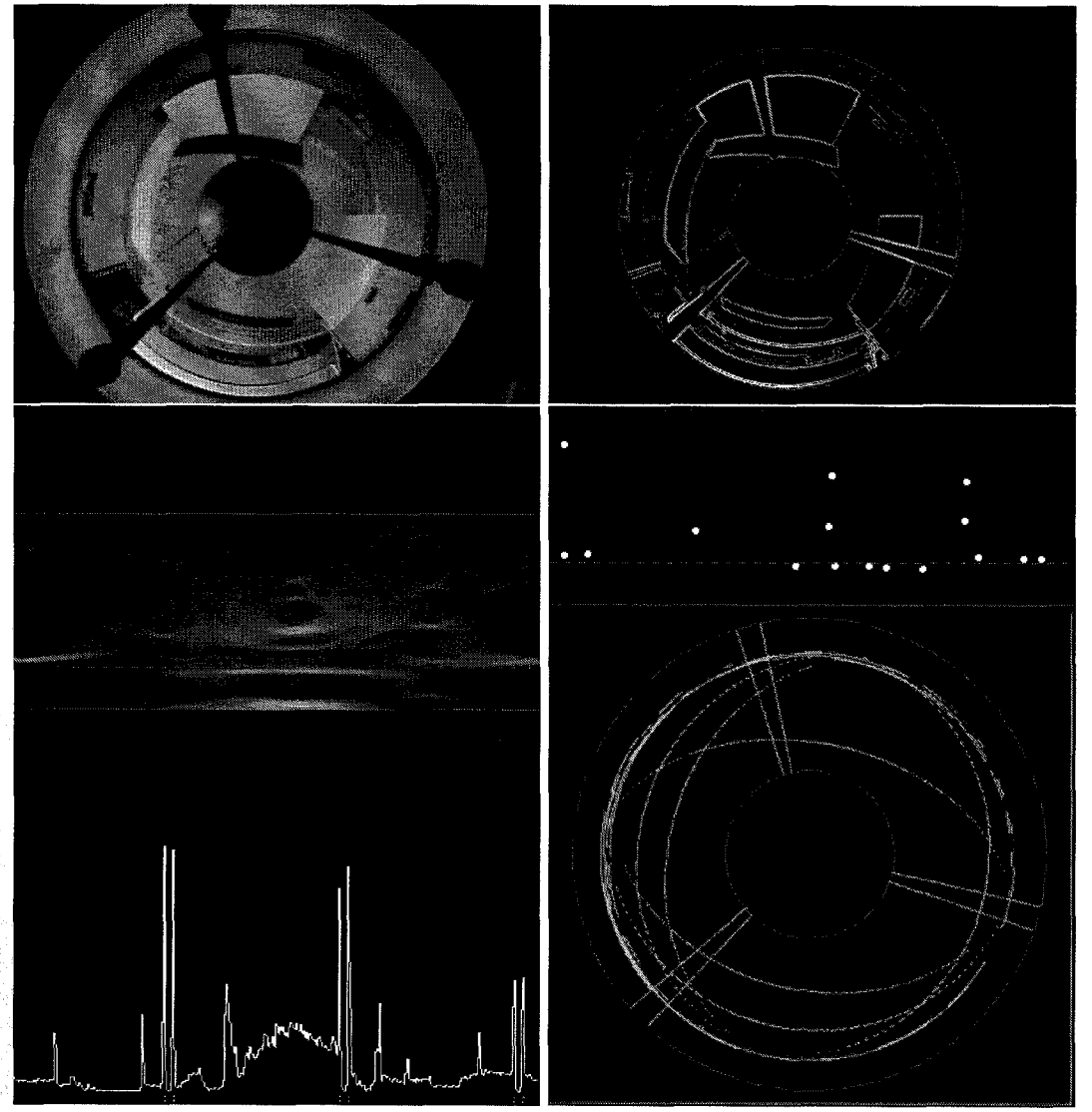


Figure 2.13: *Single lobe experiment (left to right, top to bottom) Original camera image, edge magnitude image, Panoramic horizontal and vertical Hough transform, automatic cluster detection of horizontal lines, re-projection of lines back onto image space for automatically detected lines.*

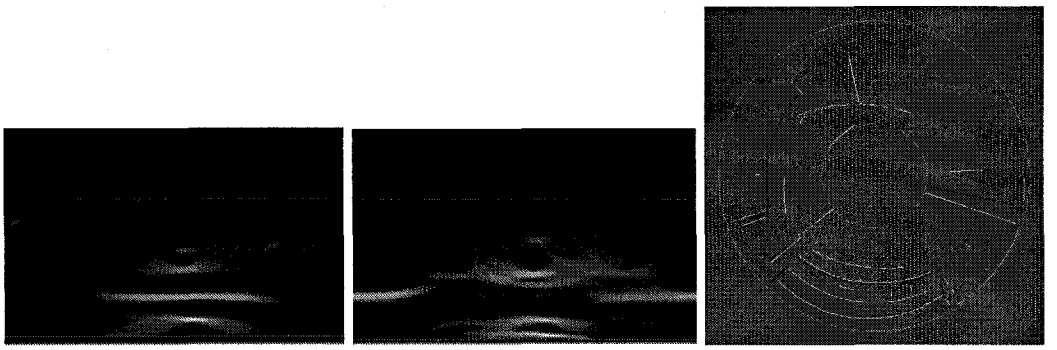


Figure 2.14: *Extensions to Original Panoramic Hough Transform: Classifying edge pixels into categories based on edge angle and line segment detection. Contrary to previous images in this chapter which put all edge pixels into one Panoramic Hough Transform space, superior results can be achieved by classifying edge pixels according to their edge angle relative to the radial line and projecting only candidate points to two separate Panoramic Hough Transform parameter spaces. This allows for improved horizontal line detection due to a less congested parameter space. Left: Panoramic Hough Transform of only those edge pixels close to orthogonal to the radial line with the intensity increasing with increasing radius. Middle: Panoramic Hough Transform of edge pixels with a decreasing intensity with increasing radius. Right: Results of finding segments within the clusters located in the Panoramic Hough Transform.*

non-vertical lines will project to some curved set of points with a complex description and are less trivial to recognize. The Panoramic Hough Transform could be applied to two or more separate non-SVP mirror lobes to find features that can then be matched to perform three-dimensional reconstruction. A mapping of a straight lined feature onto the image plane of the camera can only describe a plane along which this line must lie, it takes the matching of two different viewpoints to find the horizontal line itself.

The double-lobed catadioptric optical arrangement presented for the stereo work of this thesis allows two viewpoints to be captured by one image capture device (camera). In this case the process is simply applied twice to the same image. Matching of features is performed by the equivalent of the epi-polar constraint in traditional stereo. Observing that the projection of a 3D point lies along the same radial line in both annular regions of the two concentric mirror reflections seen in a panoramic image.

2.7 Discussion

The Panoramic Hough transform was shown in the synthetic and laboratory experiments as having good potential for detecting the projection of horizontal lines in non-SVP panoramic images, with robust performance achieved with only estimated calibration of the image sensor. Further robustness is evidenced by the good grouping of Panoramic Hough Transform

peaks with no preprocessing such as noise reducing smoothing. The parameter groupings in the parameter space appear to be distinct enough to allow acceptable horizontal line detection without advanced peak detection techniques or clustering algorithms.

The vertical line detection described above was simply a histogram of edge points by azimuth angle, a more robust method is introduced in chap 4 where a quasi-cylindrical space is created and searched for the presence of vertical line projections. This involves searching in an image that is a warp of all edge points that fit an orientation criteria.

2.8 Conclusions

The new *Panoramic Hough Transform* has been shown to offer a way to identify and localize commonly found horizontal lines to supplement the trivial detection of vertical lines in panoramic imagery. The transform is designed for catadioptric panoramic sensors with non-SVP mirror profiles such as spherical or conical mirrors.

The PHT is applied in three different ways in this thesis. First and foremost it is used for *feature detection*, for finding horizontal and vertical line segment projections without a priori knowledge from an image. Second it is applied to tracking, the presence of a predicted horizontal line projection can be verified and fine-tuned by using only a few sample points. Tracking is used in both the feature extraction for 3D panoramic stereo vision and in the tracking of known landmarks in mobile robotics. Thirdly the PHT is used in panoramic *shape-through-motion* world mapping whereby image features are tracked through an image sequence and this trajectory plotted in a PHT parameter space to characterize its trajectory relative to the moving camera.

Chapter 3

Identify and Remove Hough Transform Technique

The *Identify and Remove* technique was developed to help alleviate the difficulty in finding cluster peaks in the Panoramic Hough Transform (PHT) developed in Chapter 2. The Identify and Remove algorithm is a modification of the basic Hough technique, expanding the treatment of the transform data so that there is enhanced information linking the source and transform space images.

Since using the Identify and Remove technique in conjunction with the PHT, which is itself an extension of the basic Hough/radon transform, introduces two separate ideas it was decided to present them separately. This chapter explains the Identify and Remove technique with traditional perspective view images more familiar to those already aware of Hough transform methods.

The original Hough transform is a method for locating geometric primitives such as lines or circles in an image. Each source image point, usually an edge pixel, is mapped to a loci of points in a parameter space that correspond to the set of lines or shapes to which the original source point could belong. This is done for all candidate points, and probable line/shape definitions are located by isolating peaks in this parameter space. In even mildly cluttered scenes it can be difficult to automatically find these peaks. The most prominent peaks may be trivially found, but finding many of the lesser peaks can pose problems given the dynamic range of the different peaks. The *Identify and Remove* method is introduced where the problem is reduced to finding a single peak in several transform images rather than finding many peaks in one transform image. Only one peak has to be found for the each image, which is simply the transform point with the maximum response. The procedure starts with the most prominent peak, removes it and identifies the source pixels that contributed to it. The transform image is modified to what it would be with these source pixels removed from the original image. The *Identify and Remove* algorithm allows this new transform image to be found without having to recalculate the whole Hough transform. This cycle is performed successively, revealing smaller peaks when larger ones are removed. Experimental results are shown.

3.1 Hough Transform

The stage of image processing after edge detection is typically grouping edge pixels into the higher level abstraction features of line and line segment features [3, 98, 69].

The Hough transform is a common tool in image processing, its classic application being locating straight lines [57]. In a two-dimensional image, the location of a point alone is insufficient to identify a line, many points are available from an image but it is initially not known which points belong to one of several unknown lines. The classic Hough transform

method finds straight lines by recognizing peaks in a parameter space, each point of which represents a possible line in the image. Since a straight line in a two-dimensional image has two degrees of freedom, the set of all possible lines can be represented in a two-dimensional parameter space. Examples of line detection methods using the Hough transform can be found in [112, 3, 86, 43, 119, 72].

The method has been extended to other primitives [35] such as circles [95] and ellipses. The dimensions of the parameter space is then higher, equal to the degrees of freedom in specifying the primitive.

In Hough transforms of lines, circles, ellipses, etc, one maps a set of source image points to a set of points in a parameter space. The next task, for which the *Identify and Remove* method was created, is to extract peaks in this parameter space. It can be difficult to identify smaller peaks in the neighborhood of larger peaks. The most simple solution is to threshold this parameter space, and if the number of primitives in the original set of points is small, this can be sufficient. In the case of a large point set, such as all edge pixels in a noisy image, finding the true peaks automatically can be problematic.

3.2 Hough Transform of Straight Lines

Since a line primitive has two degrees of freedom, it is desirable to define a transformation that has is two-dimensional, has no discontinuities and is efficient. One approach is Ballard's *Foot-of-Normal* approach [12], similar to the *Radon Transform* [42]. In this case, the line a point (P_x, P_y) may lie upon is represented by two quantities, the angle of the normal to this line to a certain fixed point (C_x, C_y), and the perpendicular distance from the line to (C_x, C_y). For a line with along a direction θ, the perpendicular distance D to (C_x, C_y) can be found by finding the dot-product of (P_x, P_y) with the unit vector $(cos(\theta), sin(\theta))$ as in (Eqn. 3.1).

$$D = (P_x - C_x) \cdot cos(\theta) + (P_y - C_y) \cdot sin(\theta) \tag{3.1}$$

For a given single point (P_x, P_y), there is a set of many points in a parameter space (p, θ) that Eqn. 3.1 satisfies.

Fig. 3.1 below shows a simple image, the result of edge detection with the *Sobel* 3x3 masks, and the Hough transform of the edge image. (C_x, C_y) was chosen to be outside of the image to not introduce a singularity in the input range.

The accuracy of the located lines depends on the resolution of the transform space image.

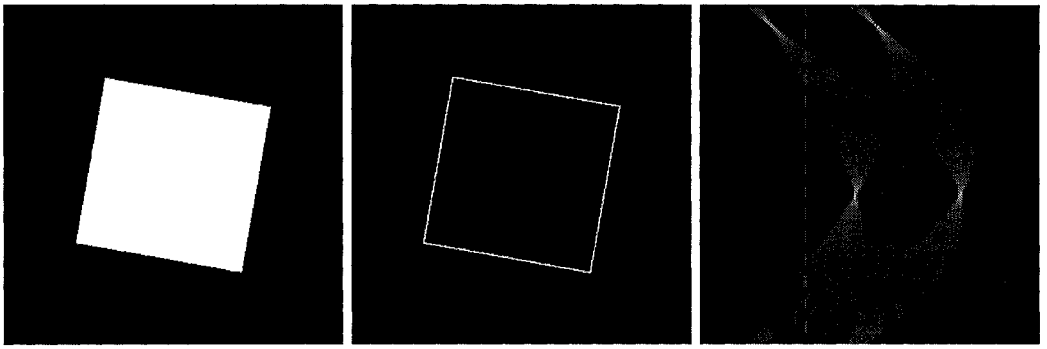


Figure 3.1: *Basic Hough Transform for Straight Lines. The horizontal axis of the parameter space is D from Eqn. 3.1 and the vertical axis is θ.*

3.3 Identify and Remove Cluster Location Algorithm

Finding peaks in parameter space images can be difficult if many of the desired primitive (for example lines) exist in the source image. One method to mitigate this effect is to find connected groups of pixels and project only the points in a connected group to a given transform space. Then only one or a few peaks will appear, depending on the complexity of the connected pixel groups.

Connectivity searches can be computationally expensive and potentially erroneous, and cannot always be done. In cases when connectivity searches are undesirable or cannot suitably break apart an image, the *Identify and Remove* algorithm can be employed. This method was originally developed for panoramic imagery with a specific application [40], but is extended to the general Hough transform case. This involves creating special data structures when performing the initial transform so that once a peak is identified the projections of all source image points that lead to that peak can be removed and thus reveal other peaks. This was done by iteratively selecting the cluster of maximal response and then identifying the source image pixels and removing their projections from this transform image. This procedure is done successively until the maximum peak in the transform image falls below a given threshold.

The Hough transform of the edge image shown in Fig. 3.1 (middle) is shown for the first three iterations of the *Identify and Remove* procedure in Fig. 3.2. Note that the intensities in the transform images are scaled to the value of the maximum peak and so lesser clusters get brighter as more dominant clusters are removed.

The data structure that allows the identification of source image pixels also allows the determination of start and end points of line segments that lie along that line. This is a second improvement over the basic Hough transform which just identifies the infinite line definition.

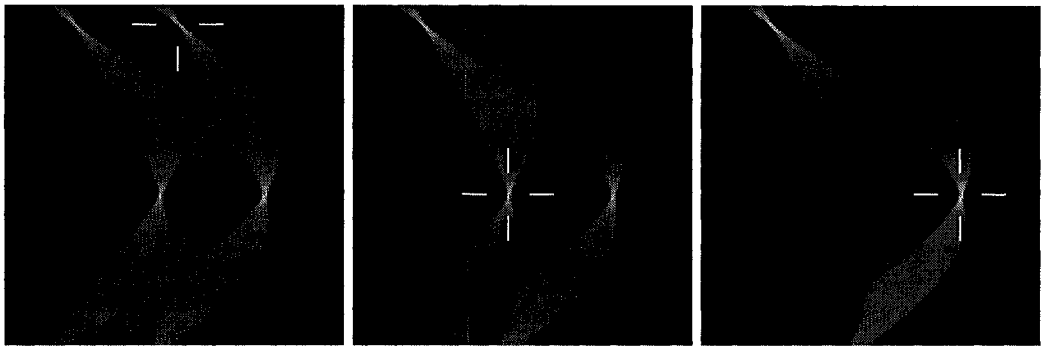


Figure 3.2: *Identify and Remove cluster detection algorithm. 3 iterations (left-to-right) on the image in 3.1. The peak around the maximum value in the PH transform image is identified and the contributing pixels are identified by linked list E (see Fig. 3.3). The effect of these pixels is removed from the Hough image for the next iteration.*

The data structures used when creating the transform is shown in Figure 3.3. One linear list, one two-dimensional list and two types of linked lists provide a circular structure for associating points in both directions between the source image and the transform space. A linear list (list B) of source image points contain the intensity and a pointer to a linked list (list C) of all the coordinates in Hough space that this pixel projects to. The transform image has an associated two-dimensional array (D, one entry for each transform image pixel), each entry of which points to a second linked list (list E) of pixel numbers (index for first linear list, list B) that projected to this point.

Hence these data structures are created along with the transform image and are used to locate all the source image pixels that correspond to a located cluster peak. The cluster is chosen not just as the maximum value point in the transform image but as a region of transform pixels around this point since degrading effects of noise and quantization will cause the pixels from a single horizontal edge to not project perfectly to one point in PHT space. The statistics of width and height of this cluster allow the determination of confidence in the existence and location of this edge.

The clusters detected using the *Identify and Remove* stage are written out to a database with each feature entry containing the line description θ_{main}, R_{main}, and the start,end points $(\theta_{begin}, \theta_{end})$ detected of segments along this line. The confidence and matching aid statistics of cluster spread (width of bounding box in PH space), number of pixels contributing to this edge and the average edge pixel strength are also provided in this output feature list.

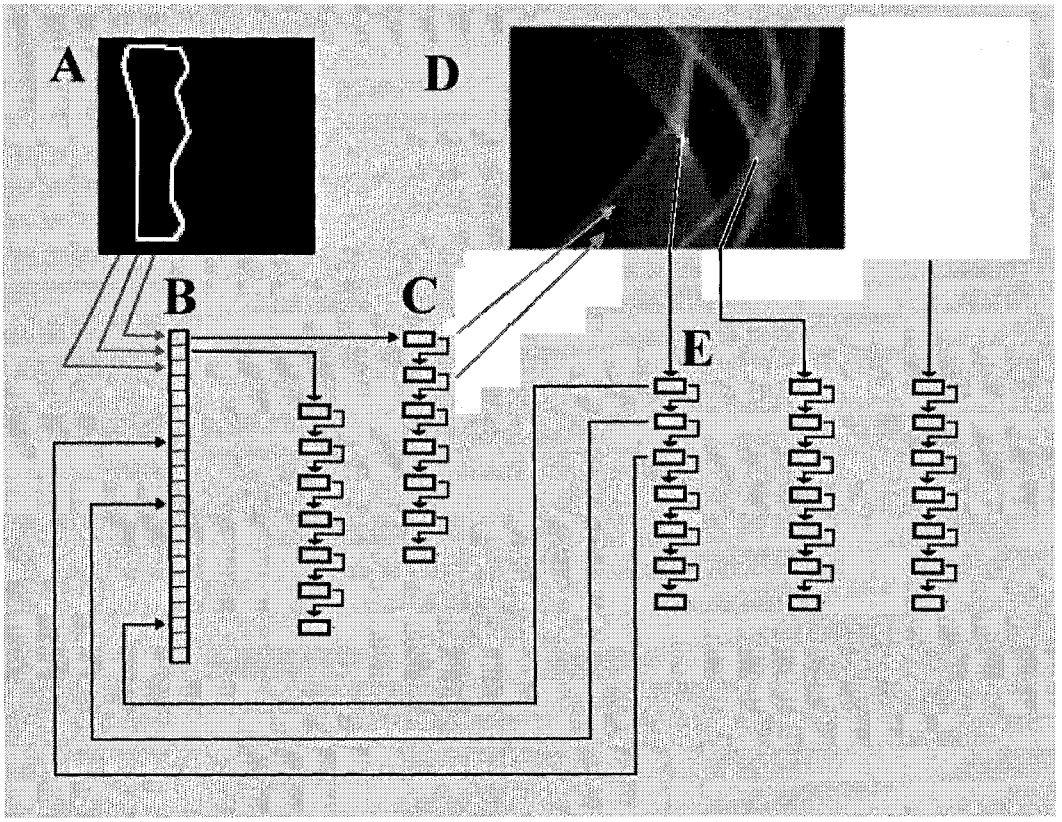


Figure 3.3: *Horizontal cluster detection algorithm (in either the north or south facing space). The purpose of this data structure is to detect clusters in the horizontal parameter image D. Edge points are selected from image A create an entry in linear array B. B is indexed by an arbitrary pixel number p_i and each entry contains the edge magnitude and pointer to the beginning of the linked list C that contains all the u, v locations in the parameter image D that the edge point maps to. The edge value is added to the parameter image D at each of the u, v locations. Each pixel in D has a pointer to the beginning of another linked list E that contains the pixel numbers p^i for all source image pixels that project onto this image pixel, i.e. each link in E points to a pixel in linear array B. This allows all the source pixels to be identified that project onto a given point in the image. A cluster is identified and all pixels responsible for that peak are removed in order to find the next cluster. Thus the clusters are identified and removed in sequence.*

3.4 Experiments

The method is demonstrated on a real image shown in Fig. 3.4, the image is noisy and has only 5 bits of resolution. The image is edge detected using the euclidean magnitude $\sqrt{G_x^2 + G_y^2}$ of the convolution with the sobel horizontal and vertical 3 x 3 masks. This edge image is thresholded at half the greyscale range (Fig. 3.5A) and result processed to the Identify and Remove algorithm. (Fig. 3.5B) shows the extracted lines.

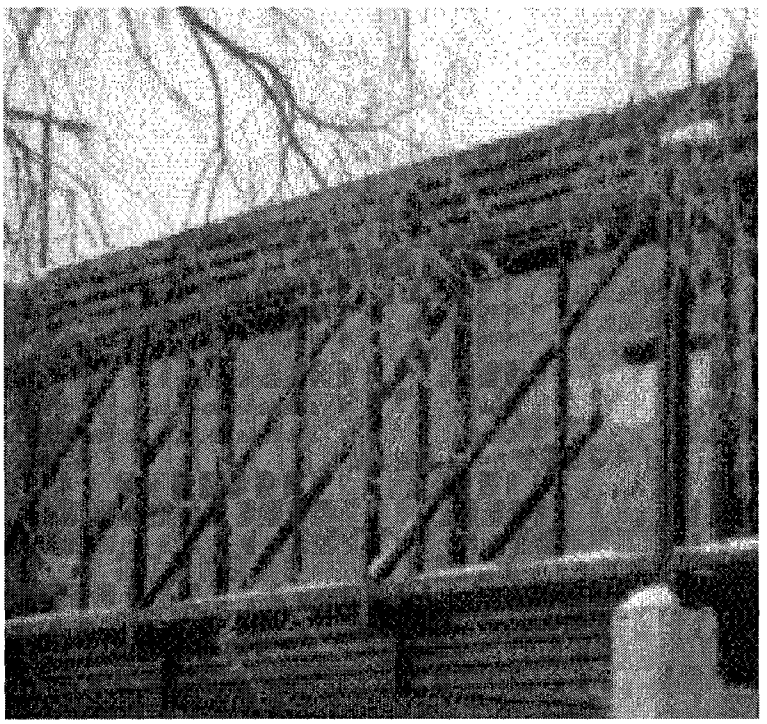


Figure 3.4: *Original walkway image.*

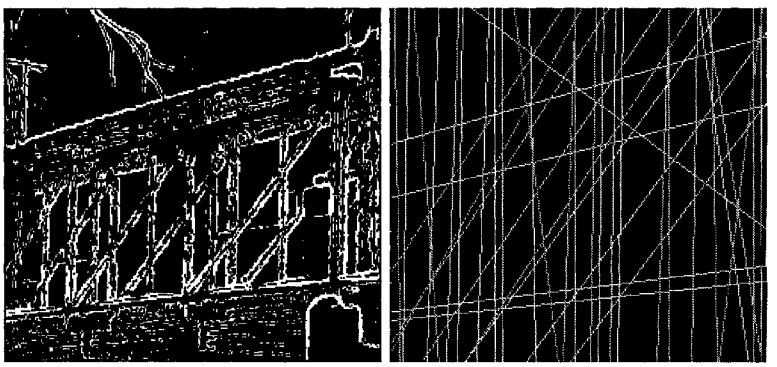


Figure 3.5: *A (left): Edge detected image of Fig. 3.4 using the Sobel edge masks. B(right) automatically detected lines using the* Identify and Remove *method.*

The original (before any peak removal) Hough transform image, and the resultant auto-

matically detected peak locations are shown in Fig. 3.7. Eight stages of the Hough transform image as peaks are identified and removed are shown in Fig. 3.4.

The experiment shown uses a simple peak grouping procedure, that of choosing all pixels within a square fixed range (± 2 pixels) of the maximal transform image response. Using a fixed range can produce the artifacts of a non-distinct line showing up more than once if the range was too small, or of one cluster incorrectly claiming pixels from another if the range was too large. A practical system would likely use a more sophisticated method for determining the extent of the peak's spread. For example, chosing a rectangular or elliptical window with a major and minor axis to define a peak could alleviate problems associated with having to define a fixed range.

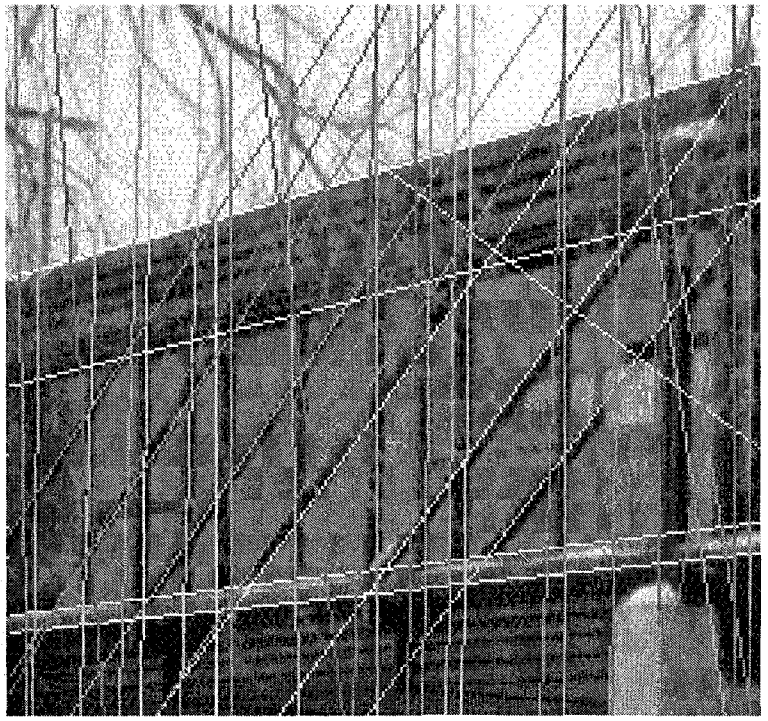


Figure 3.6: *Extracted line superimposed on original walkway image.*

3.5 Conclusions

The *Identify and Remove* extension of the Hough transform was introduced. It is a way of arranging data structures to allow the identification of all source points that lead to a given point in the parameter space, location of line segment(s) that these source points may belong to, and removal of their affect on the transform image without the necessity of recalculating the transform. In this way peaks can be trivially identified by finding the maximal response

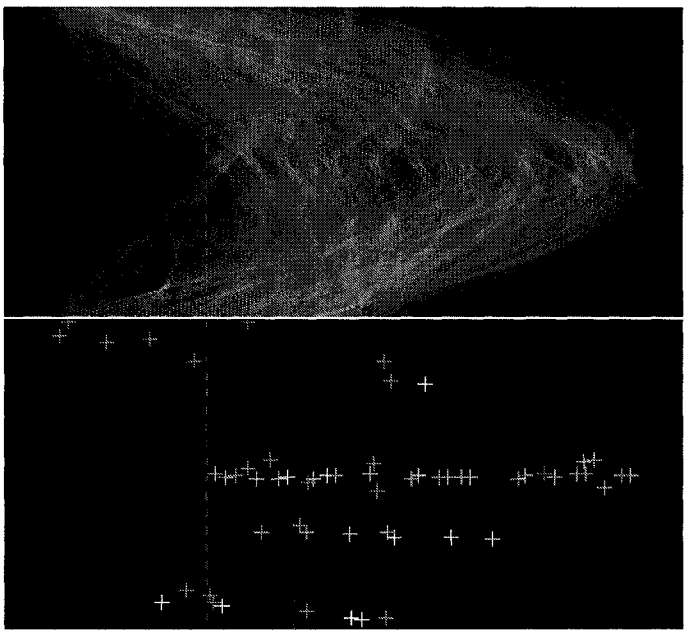


Figure 3.7: *(Above) Original Hough transform of walkway image. (Below) Automatically detected peaks using the Identify and Remove method.*

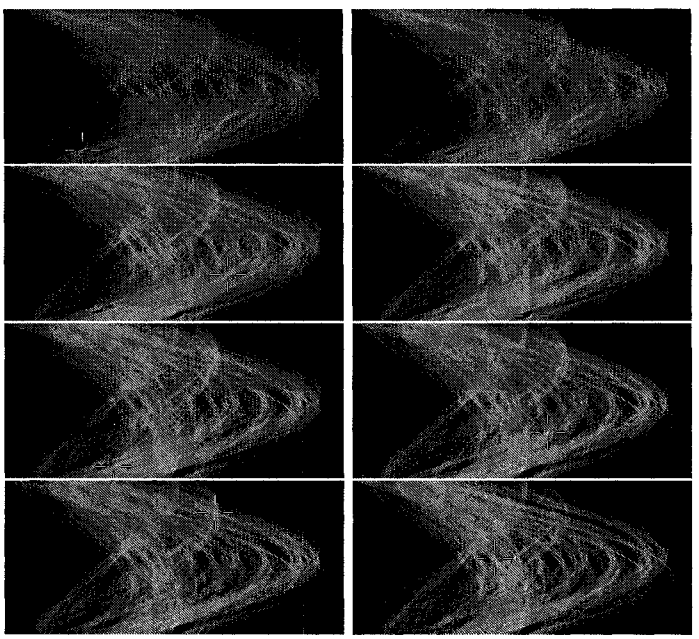


Figure 3.8: *First 8 successive stages of the Identify and Remove cluster detection algorithm (left-to-right, top-to-bottom). The maximum value in the PH transform image defines a cluster center marked by the cross-hair, determined from the centroid of a square region around the maximal response. All source image pixels whose projections fall in this bounding box are identified and their projections removed for the next iteration.*

in the transform image, identifying the contributing source pixels, followed by removing this peak allowing the next largest peak to provide the maximum response.

The *Identify and Remove* algorithm was successfully demonstrated for a low quality real image, and suggestions were given on extending this algorithm to better determine the neighborhood size and shape of of transform peaks. Since the algorithm pin-points the pixels responsible for a cluster peak, a histogram approach can identify line segments' start and end points. This was not shown in this chapter, but is done when this method is applied to the Panoramic Hough Transform.

Chapter 4

Line Segment Extraction in Panoramic Images

Feature extraction is the process of finding definitions of primitives from an image. This has been further defined in this thesis as consisting of two stages: feature detection and verification. Feature detection takes the two-dimensional pixel array, and with no a priori knowledge of the content attempts to extract a list of feature primitives. In this work, feature detection consists of finding the projections of horizontal and vertical line segments from object edges and the projections of vertical polygonal object faces.

The previous chapters introduced the *Panoramic Hough Transform* and the *Identify and Remove* algorithm. The PHT has the ability to recognize the projection of infinite horizontal line edges from non-SVP panoramic images, and the Identify and Remove Algorithm allows the identification of segments and more robust recognition of cluster peaks in cluttered and blurry parameter space images. The two are combined in this chapter to create the necessary functionality required for horizontal line segment feature detection in panoramic images.

Detecting the projection of straight vertical line segments is also outlined. A Hough transform approach was not necessary as with the horizontal segments, due to the unique differentiating quality of their projection. If the panoramic image sensor is posed vertically, all vertical lines project to straight radial lines making their detection trivial.

4.1 Feature Detection System Components

The feature detection processes can be described as a sub-system of a greater panoramic vision system, and provides features that can be used for mobile robot landmark detection, or combined with features from another camera or viewpoint for panoramic stereo 3D scene reconstruction.

The features such a sub-system reports depend on what it was designed to find. The applications targeted by this thesis are principally that of reconstructing and navigating in a man made polyhedral world consisting of horizontal and vertical polygons. This paradigm justifies the search for horizontal and vertical line segment edges as basic primitives. It was decided to find the polygon features themselves as a higher level abstraction of the segment features. Thus a the feature detection sub-system developed is restricted to finding the projections of horizontal and vertical line segment edges from non-SVP panoramic images.

Since these features are edges, they are found by assuming a model of abrupt changes in image intensity corresponding to polygon face edges. Edge detection is performed, which is a two-dimensional differentiation procedure to produce an image of edge pixels, so called *edgels*, whose magnitude is proportional to the gradient of intensity change. Since differentiation increases high frequency noise as well as high frequency edges, the image can be smoothed prior to this edge detection as part of the pre-processing. A smoothing operator is

inherent in the edge detection process used, the Sobel mask pair, as outlined in the chapter on pre-processing.

The horizontal and vertical segment features are found separately, and for increased robustness, each was divided into two opposite types according to the direction of the edge intensity derivative for a total of four primitive types. The feature detection system separates these processing stages after edge detection, and recombines the results into one output feature list.

4.2 Separate Transform Spaces For Different Edge Directions

In Chapter 2 on the PHT, all points in the high-pass image (generated by the Sobel edge template pair) are projected onto a single PHT parameter image, independent of edge direction. Points corresponding to vertical lines are projected along with those from horizontal lines, providing extra unnecessary calculations and noise. Also horizontal edges that represent the top or bottom of a lighter colored object are indistinguishable from each other.

The first improvement to this method is to filter the edgels according to edge angle and send them to one of four algorithms, two for horizontal lines and two for vertical. One horizontal algorithm collects edge pixels that represent increases in image intensity as radius increases, the other represent intensity decreases. Likewise one vertical algorithm processes edge pixels that correspond to increasing intensity in a clockwise direction, the other for decreasing intensity. This is shown in the middle stages of the flow of operations depicted in Figure 4.1.

4.3 Identify and Remove Cluster Location Algorithm for Horizontal Edges

This section presents the processing for edge points classified according to edge angle as being candidate pixels for one of the two types of horizontal lines (increasing and decreasing intensity with increasing radius). The PHT is used for the horizontal edges only.

Although the PHT shows good grouping in parameter space, automatically isolating these cluster peaks successfully in cluttered or noisy images is problematic. Improvements were needed for robust automatic detection. Two methods to improve this are: 1-Find connected groups of edge pixels and project only the points in a connected group to the PHT and vertical histogram spaces, and then detect clusters within this subset of the entire

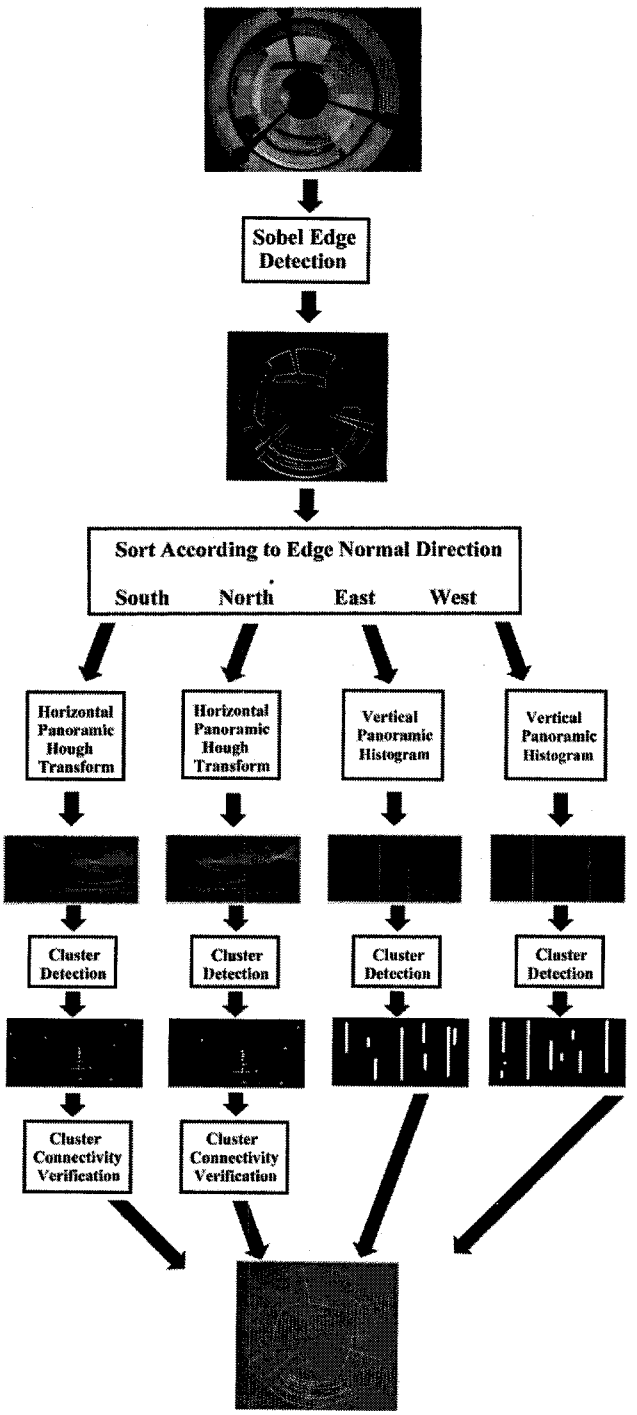


Figure 4.1: *Data flow from panoramic image to recognized horizontal and vertical features (lines and regions). The image captured by the panoramic image sensor is edge-detected, and the angle of the edge (relative to the radial line) causes the edge pixel to be sent to one of four Panoramic processing parameter images. Either a two-dimensional Hough transform to detect the curved loci of horizontal lines is performed, or a two-dimensional histogram to find straight radial lines is created. Clusters are located in these 4 spaces and feature lines extracted.*

image pixels. 2-Project all candidate edge points in the source image to their respective PHT or vertical segment detection spaces and remove all projections from identified peaks, making other lesser peaks visible and removed in turn.

To avoid having to make potentially erroneous and computationally expensive connectivity searches, the second method, the *Identify and Remove* cluster detection method, was adopted. This involves creating special data structures when performing the initial transform so that once a peak is identified the projections of all source image points that lead to that peak can be removed and thus reveal other peaks. The clusters of maximal response are iteratively selected with the source image pixels identified and their projections removed from the PHT image. This is done successively until the maximum peak in the transform image falls below a given threshold.

A real image captured from a panoramic image sensor is shown in Figure 4.2. The horizontal PHT image for the first six iterations of the *Identify and Remove* procedure for edge points with radially increasing intensity, is shown in Figs. 4.6a-f. Note that the transform images are scaled to the value of the maximum peak and so lesser clusters get brighter as more dominant clusters are removed. The data structure that allows the identification of source image pixels also allows the determination of start and end points of line segments that lie along that line as opposed to just identifying the line definition as in Chapter 3.

Figs. 4.6a-f show both the PHT transform space and a histogram of pixel frequency according to azimuth angle. Also shown is the filtered version of this histogram and the extracted start and end points for segments.

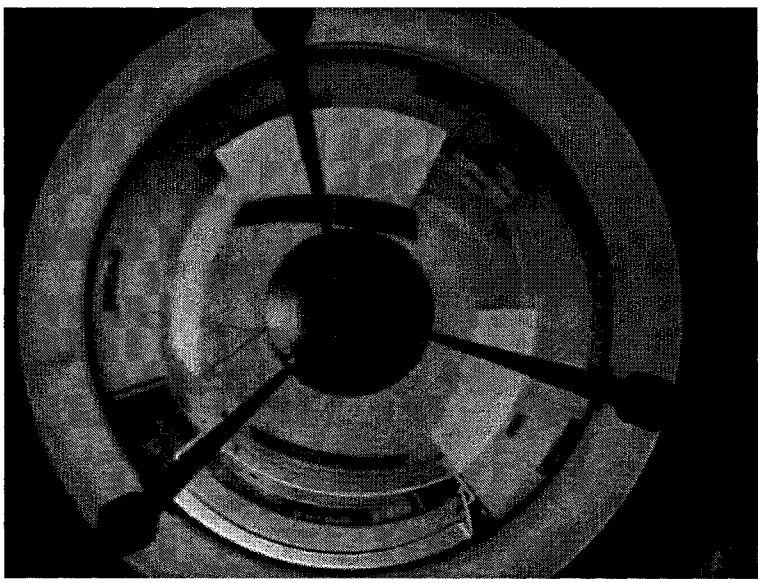


Figure 4.2: *Original image captured by a catadioptric single-lobed image sensor. Aspect ratio compensated image from a Sony 999 NTSC camera with a 15cm diameter spherical mirror.*

The data structures used when creating the PHT is shown in Figure 4.3. One linear

list, one two-dimensional list and two types of linked lists provide a circular structure for associating points in both directions between the source image and the PHT space.

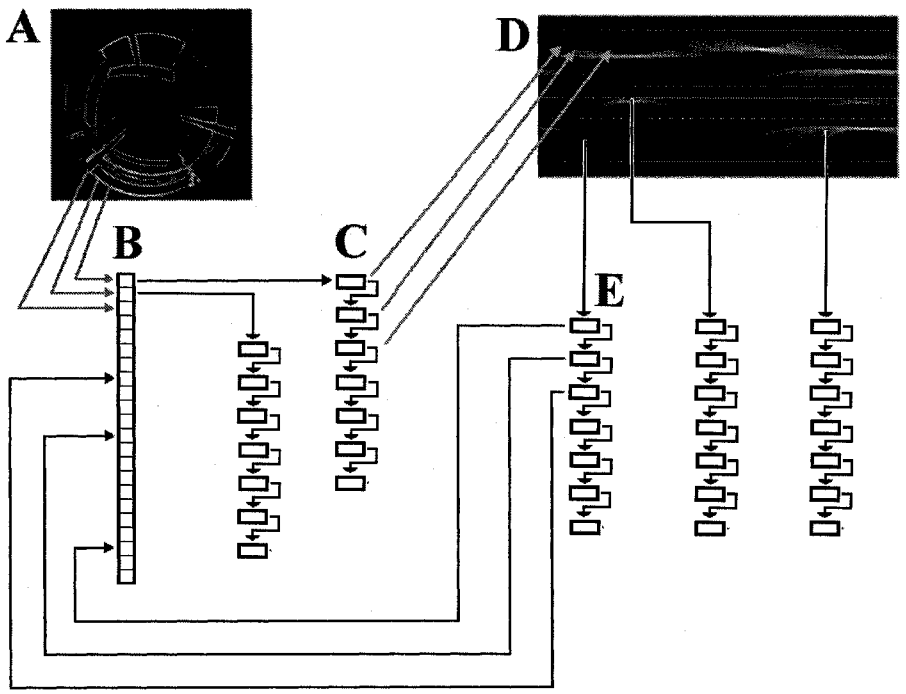


Figure 4.3: *Horizontal cluster detection algorithm (in either the north or south facing space). The purpose of this data structure is to detect clusters in the horizontal PHT parameter image D. Edge points are selected from image A and those whose angle fall in the desired range create an entry in linear array B. B is indexed by an arbitrary pixel number p^i and each entry contains the edge magnitude and pointer to the beginning of the linked list C that contains all the u, v locations in the PHT parameter image D that the edge point maps to. The edge value is added to the PHT parameter image D at each of the u, v locations. Each pixel in D has a pointer to the beginning of another linked list E that contains the pixel numbers p^i for all source image pixels that project onto this PHT image pixel, i.e. each link in E points to a pixel in linear array B. This allows all the source pixels to be identified that project onto a given point in the PHT image. A cluster is identified and all pixels responsible for that peak are removed in order to find the next cluster. Thus the clusters are identified and removed in sequence.*

One linear list of candidate edge points contain the edge intensity and a pointer to a linked list of all the coordinates in PHT space that this pixel projects to. The PHT image has an associated two-dimensional array (one entry for each transform image pixel), each entry of which points to a second linked list of pixel numbers (index for first linear list) that projected to this point.

Hence these data structures are created along with the PHT transform image and are used to locate all the source image pixels that correspond to an identified cluster peak. The cluster center is chosen not just as the maximum value point in the PHT image but as the

centroid of a region of transform pixels around this point since degrading effects of noise and quantization will cause the pixels from a single horizontal edge to not project perfectly to one point in the PHT space. The statistics of width and height of this cluster allow the determination of confidence in the existence and location of this edge. The clusters are typically more uncertain in their central azimuth angle θ_{main} than in R_{main} and so the clusters are typically wider in the θ direction (X-axis in the transform images). Also as expected, the clusters widen as R_{main} approached the horizon line as that the central angle becomes more indeterminate on the horizon.

The clusters detected using the *Identify and Remove* stage are written out to a database with each feature entry containing the line description θ_{main}, R_{main}, and the start,end points $(\theta_{begin}, \theta_{end})$ detected of segments along this line. The confidence and matching aid statistics of cluster spread (width of bounding box in PHT space), number of pixels contributing to this edge and the total edge strength are also provided in this output feature list.

4.4 Cluster Connectivity Verification

One drawback of the *Identify and Remove* cluster detection method is that connectivity information is not used and clusters that are detected first can falsely take possession of pixels belonging to other lines. If those lines have image points whose projections pass through the bounding box of a previously detected cluster, they will be erroneously detected as part of the first cluster and removed. Thus the "stronger" clusters can take pixels from "weaker" clusters. The histogram of the stronger cluster will have an extra start and stop angle (segment) with a corresponding hole broken in the histogram of the weaker cluster. This is shown in Figure 4.4.

The solution applied was to detect when this possibility could arise by creating a function $FO()$ (find overlap, Eqn. 4.1).

$$(\theta_{start}, \theta_{end}) = FO(\theta_1, R_1, \theta_2, R_2) \tag{4.1}$$

This provides a range of overlap between two horizontal line projections if the two lines do indeed overlap.

The database of clusters detected using the *Identify and Remove* stage are searched for potential pixel mis-assignment by checking each cluster with all latter detected clusters for line segments coincident with the potential overlap region. If this stronger cluster indeed has a stretch of pixels that coincides with the overlap region, and is between or at the end of line segments in the weaker cluster, it is assumed that these pixels were improperly assigned.

In this case the segment is taken from the stronger cluster and combined with the segments in the weaker cluster. Figure 4.4 also shows an example after such a correction. In images tested, it was found that this phenomenon occurred frequently and that the *Identify and Remove* cluster detection algorithm needed such verification and correction processing.

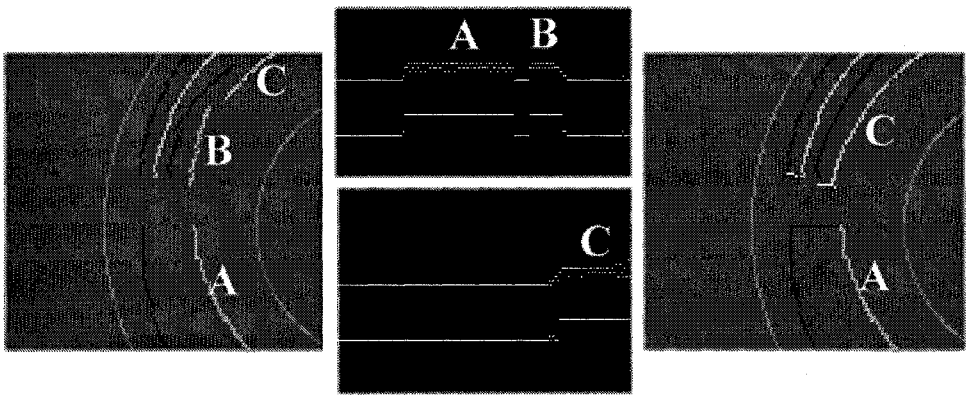


Figure 4.4: *Demonstration of how recognizing cluster solely on PHT transform image can lead to false assignment of pixels. In this example the lines for cluster 1 and 2 cross and overlap at B, causing pixel group B to be falsely assigned to Cluster 1. The left image shows the reconstruction after cluster detection, the center images show the cluster edge histogram for Cluster 1 (top) and Cluster 2 (bottom). The right image shows the corrected group assignment after group B is joined with C and moved from Cluster 1 to Cluster 2.*

4.5 Vertical Line Detection

Vertical lines are feature edges parallel to the main camera/mirror axis assuming the previously defined geometry. They can be detected in a more simple manner than the horizontal lines by simply finding connected edge pixels along radial lines that share an edge of similar angle. Detecting these features is less involved than the horizontal line search due to the non-interference of one line with another.

For locating these connected radial segments corresponding to vertical lines, only pixels satisfying an edge direction and magnitude constraint are considered. Pixels are chosen whose edge angle is within 45° of the radial direction, with others rejected as likely being potential horizontal line pixels.

This allows vertical edge segments of two types to be detected separately, those whose intensity increases or decreases with increasing azimuth angle.

Vertical line segment projections are located by searching for connected edge pixels of each of the two vertical pixel types. A method chosen *ad hoc* was to create two quasi-cylindrical warp images, one for edge pixels with a positive derivative with increasing az-

imuth angle, and a second for those with a negative derivative. These quasi-cylindrical warp images are drawn graphically in Fig. 4.1 with the X-axis repesenting azimuth angle ($0 - 360°$) and the Y-axis represents radius. Vertical line segment projections were found by finding vertically connected runs of pixels in these two images.

4.6 Discussion: Results of Horizontal and Vertical Line Feature Extraction

After the application of the Sobel edge detection template pair, the classification of edge points according to edge direction and processing by either the PHT or vertical line detection algorithm, a database of horizontal and vertical line projections is provided. The automatically detected line features from Figure 4.2 are redrawn in Figure 4.5.

The results are quite robust given the noise and low resolution of the input image (from a NTSC greyscale video camera) and the lack of any camera calibration other than the calculation of the aspect ratio. The three parameters of camera focal length, mirror radius and distance between the camera focal point and mirror center were measured very roughly yet the results came out quite good. The scene objects however, were very distinct and of uniform shading and perhaps of greater contrast to the background and one another than would naturally occur in the targeted application for this vision system: mobile robots inside a building. No false positives were created in our experiment but to detect more subtle edges one might have to lower the thresholds for edge pixel magnitude. To detect smaller feature sizes, the minimum segment length threshold might have to be also reduced. This may lead to the production of many false positives.

When this method is used for panoramic stereo vision, it is expected that any falsely detected features would not find a match and not make it into the final model.

4.7 Conclusions

The *Panoramic Hough transform* has shown to be a useful and robust method for detecting the projection of horizontal line edges for catadioptric panoramic image sensors with a non-SVP (Single Viewpoint) mirror profile. However it cannot provide reliable automatic detection alone, and needs to be combined with methods to help distinguish individual cluster peaks in the parameter space that the PHT provides. Also the PHT alone can only indicate the presence of horizontal edge lines and not locate the start and end points of segments.

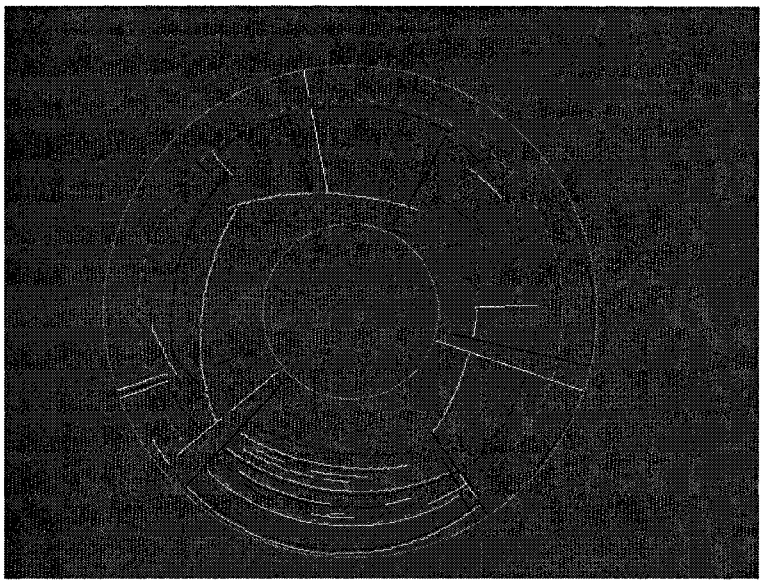


Figure 4.5: *Reconstruction of detected projections of vertical and horizontal line edges from Figure 4.2.*

The filtering of edge pixels into two different PHT spaces allow the PHT parameter spaces to be less cluttered and less noisy than if all edge pixels, horizontal and vertical, are put into on PHT parameter space. Also closely spaced edges of increasing and decreasing intensity, as around a thin horizontal object, can be discriminated as that they are mapped into different spaces. The *Identify and Remove* method allow for closely spaced clusters in this parameter space to be separately identified and line segment information to be found to augment the line detection. Failure artifacts that this method generates (false assignment of line segments) were described and a solution was provided that successfully corrects this intrinsic behavior.

A quasi-cylindrical warp approach for finding vertical line segments was also introduced.

The methods proposed allow for robust feature extraction of horizontal and vertical line segments from panoramic images with non-SVP mirror profiles, enabling such panoramic catadioptric image sensors to be useful for modeling and machine vision tasks.

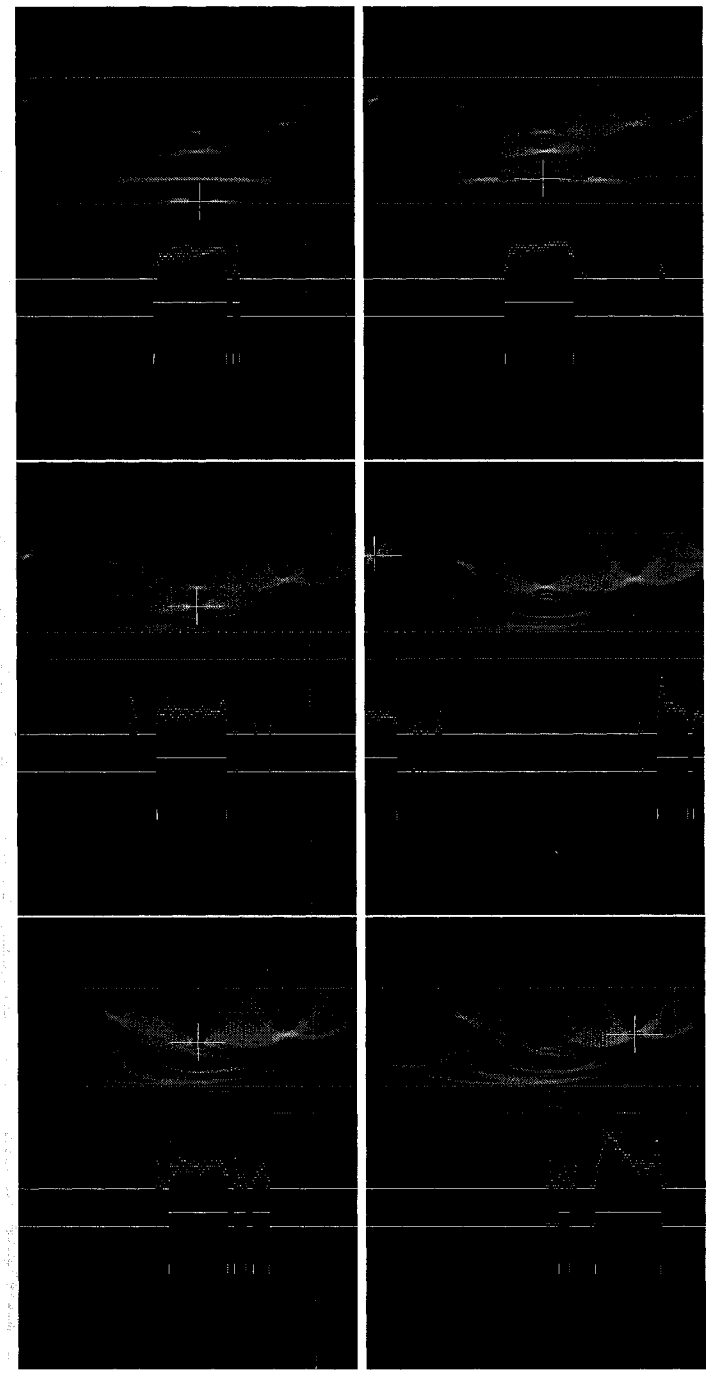


Figure 4.6: *First 6 successive stages of the Identify and Remove cluster detection algorithm (left-to-right, top-to-bottom). The maximum value in the PHT image defines a cluster center marked by the cross-hair, from which the width and height (marked by a black bounding box) are determined. All source image pixels whose projections fall in this bounding box are identified and their projections removed for the next iteration. Below the PHT image is the raw and processed histogram of edge pixel frequency according to angular position. At the bottom of each image is the determined start and end point of that line segment(s) according to azimuth angle θ. shown as vertical bars.*

Chapter 5

Stages of Feature Extraction for a Single Lobe

Each annular region of the captured image in processed separately, analogous to separately captured image in classic stereo. Each lobe reflection is analyzed for the presence of three types of primitives useful for modeling a rectilinear polyhedral world: projections of horizontal line segments, projections of vertical line segments and projections of rectangular faces. The resulting feature set for both lobes are then matched and 3D scene edges and faces reconstructed.

The initial image is presented along with parameters to define the geometry and useful image area. The geometry is expressed by the three parameters of the dioptric camera: focal length (in pixels), distance from the dioptric camera focal point to the mirror center, and the radius of the mirror lobe. The remaining two parameters, the inner and outer radii in the image, define the annular regions that have overlapping and therefore stereo views of the scene. Together these five parameters are used to extract the three primitive feature types from the image. These parameters are used by both the feature detection and verification stages described below. The same image is then re-processed with a different parameter set corresponding to the other lobe.

5.1 Hypothesize and Verify Paradigm

The feature extraction is based on a hypothesize and verify paradigm, various processes infer the probable presence of a feature which is confirmed and fine-tuned by tracking stages. For clarity, the terms *feature extraction* and *feature detection* are given different meanings in this thesis. Feature extraction is defined herein as the overall process, including both hypothesis and verification stages. Feature detection is a hypothesis step, and in this system comprises processing of the edge detected image to find projections of horizontal and vertical lines using the PHT and quasi-cylindrical warp methods. The stages falling under the hypothesis category function on a higher level of abstraction than pixels, two stages process feature list data and insert new segment features and create rectangular projection primitives from segment features.

The verification stages confirm the presence and fine-tune the parameters of segment features through inspection in the original image. This is performed by the tracking stages, which are the same as used by the mobile robot application for adjusting location of detected line segment landmarks.

The original feature detection is reasonably accurate, and in many other system may be sufficient in itself. However, the feedback of a tracking stage was required due to the sensitivity to feature errors in a stereo system with a small effective baseline. Errors in the data provided by the feature extraction can be grouped into four main types; false

positives, false negatives, slight error in line definition $(R_{main}, \theta_{main})$, and error in line segment endpoints. The effect of three of the error types, excluding correction of false negatives, can be mitigated with the feedback nature of this hypothesize-verify methodology.

5.2 Hypothesis Stage: Feature Detection of Line Segment Projection Features

The bulk of the processing as far as computational cost and system complexity is the feature detection that attempts to find the projection of the horizontal and vertical line edges of objects in the scene. Pixels on image edges are identified and categorized using the Panoramic Hough Transform and radial line detection

Typically the first stage in computer vision systems is pre-processing. With experiments performed with real imagery, the images from NTSC cameras required smoothing whereas the high-resolution digital camera did not. Edge detection using the Sobel mask pair provides an edgel (an edge pixel), composed of edge magnitude and direction, for each pixel position. Edgels with a magnitude below a threshold were discarded.

The thresholded edge magnitude image was processed as per Chapter 4 by classifying each edgel into one of four types according to edge angle in polar coordinates (i.e., angle relative to the radial direction of the edge from the image center). Pixels are classified into belonging to one of two types of horizontal edges or one of two types of vertical edges. The projection of a horizontal edge can have a positive or negative intensity change with increasing radius, and projections of vertical edges (radial lines in our geometry) can similarly be classified as going from dark to light or light to dark as one traverses the edge in a clockwise direction. The horizontal edgels are processed in separate PHT spaces, and the image is warped to two pseudo-cylindrical spaces with only vertical edgels of each type in each space. The PHT transform space images and the automatically detected cluster locations for the image in Fig. 5.2 is shown in the subsequent images.

The original feature detection process is repeated for several image slices, with the results combined. A tracking procedure is applied to confirm the existence, and fine-tune position and end points of all the features. A rectangle hypothesis stage looks for incomplete rectangular shapes and adds features which are verified by a second tracking stage. The rectangular region feature primitives are then found by determining sets of closed line segment features. Finally the feature list is pruned to remove small or inconsistent features to provide the final feature list for each image lobe. The process is shown in Fig. 5.1.

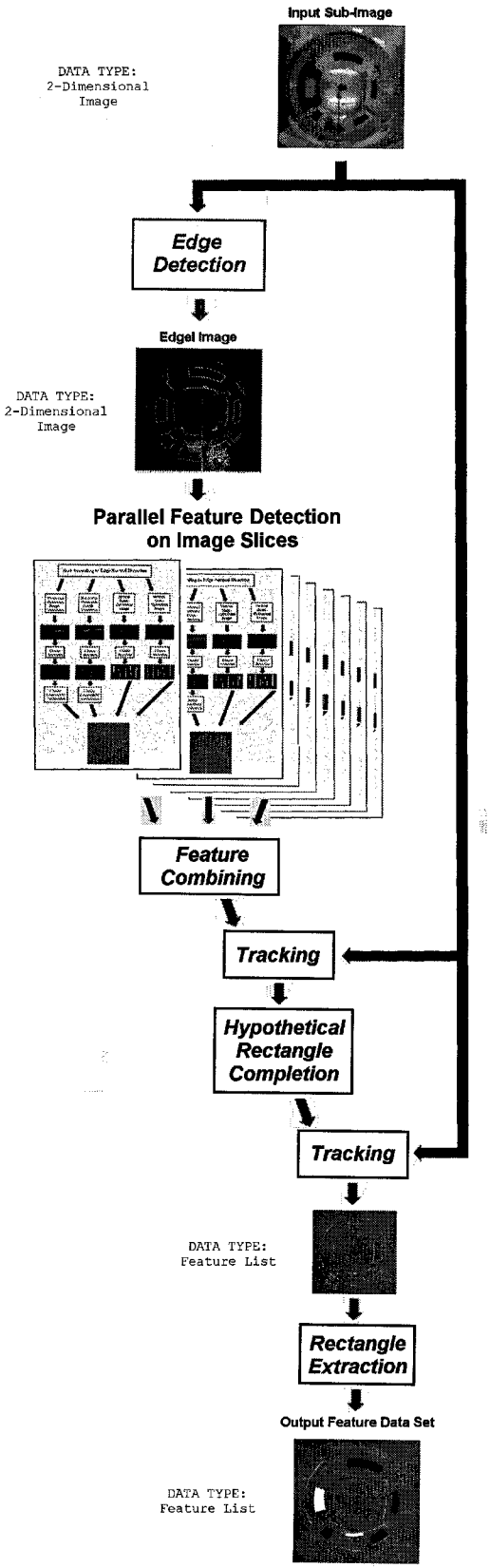


Figure 5.1: *Feature extraction stages for one lobe. Detection and hypothesis stages propose features which are verified in the source image by tracking stages. The initial feature detection is divided into radial slice sub-images for processing efficiency and resultant features combined.*

5.3 Preprocessing and Edge Detection

The images captured from the Canon D30 (the dioptric component of our experimental sensor), when the focus and aperture were set correctly, were of good sharp quality and and pre-processing was not required. This differs from the NTSC experiments where image noise required the use of an averaging filter (3x3) to be applied first to smooth out shot noise.

The image was edge detected using the 3x3 Sobel mask pair, and the magnitude determined from the euclidean of the two correlation mask responses. The edge angle information was retained for use in pixel sorting as per Chapter 4. Edge pixels (edgels) below a threshold were removed to reduce processing time and data structure size in the *Identify and Remove* operations.

The inner lobe of the image in experiment No.1 (Chapter 8) is shown below in Fig. 5.2, with the corresponding edge image section.

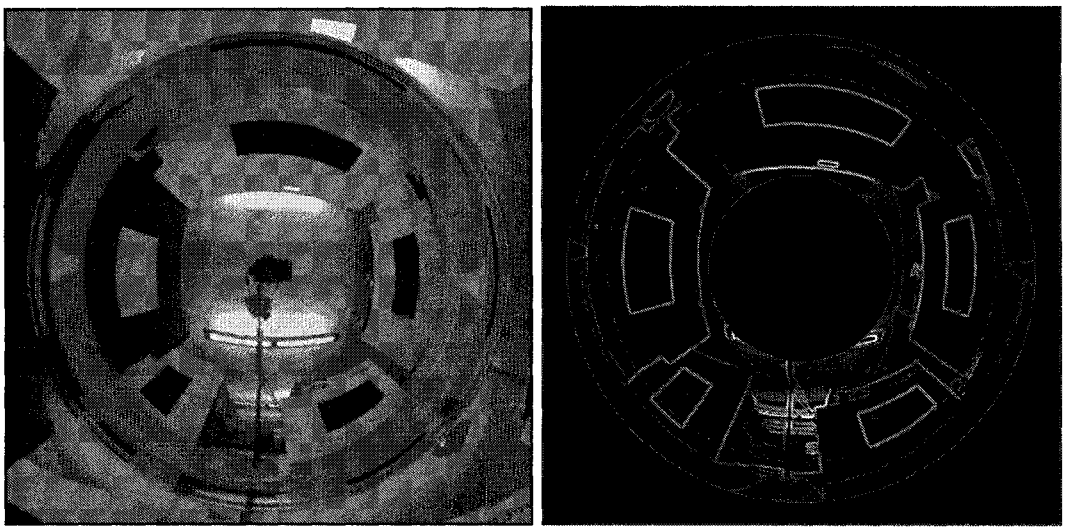


Figure 5.2: *Processing inner lobe: Original image fragment (left) Thresholded magnitude edge detected image (right).*

5.4 Splitting and Combining

With real images of the large size captured by our system, it was found that it was necessary to divide the image into sections for separate processing, with the resultant features combined. This was done for two reason, first to reduce the memory requirements, and second to improve the detection of small features. The memory requirements of the large data structures generated by the *Identify and Remove* algorithm required processing on a

computer with RAM exceeding 64 Megabytes thus the algorithm is not suitable for mobile robotics applications. The feature detection was improved since many features were not global to the whole image and in cluttered scenes could project onto one another. The latter problem was not great, but the small improvements in line segment extraction performance gained by this split and combine approach allowed for a more robust rectangle detection.

The inner lobe sub-image was divided into 8 octants, eight radial sections of 45^o each. Likewise the outer lobe sub-image was divided into 16 slices of 22.5^o each. The combining process needs only to join horizontal line segments, and does so by examining those features that meet at these octant or slice boundaries. A match likelihood estimator decides whether to join features according to their R_{main}, θ_{main} differences and relation between the spread and feature length.

The feature detection within each image slice uses the methods of Chapter 4, and is shown in Fig. 5.3. Edge detection produces edgels which are sorted and processed in one of four parallel streams, for the two types of horizontal line projection feature detection, and two for detecting vertical lines.

Images from the processing of the eight octants for Fig. 5.2 follow in Figs. (5.4-5.11), with the combined results shown in Fig. 5.12.

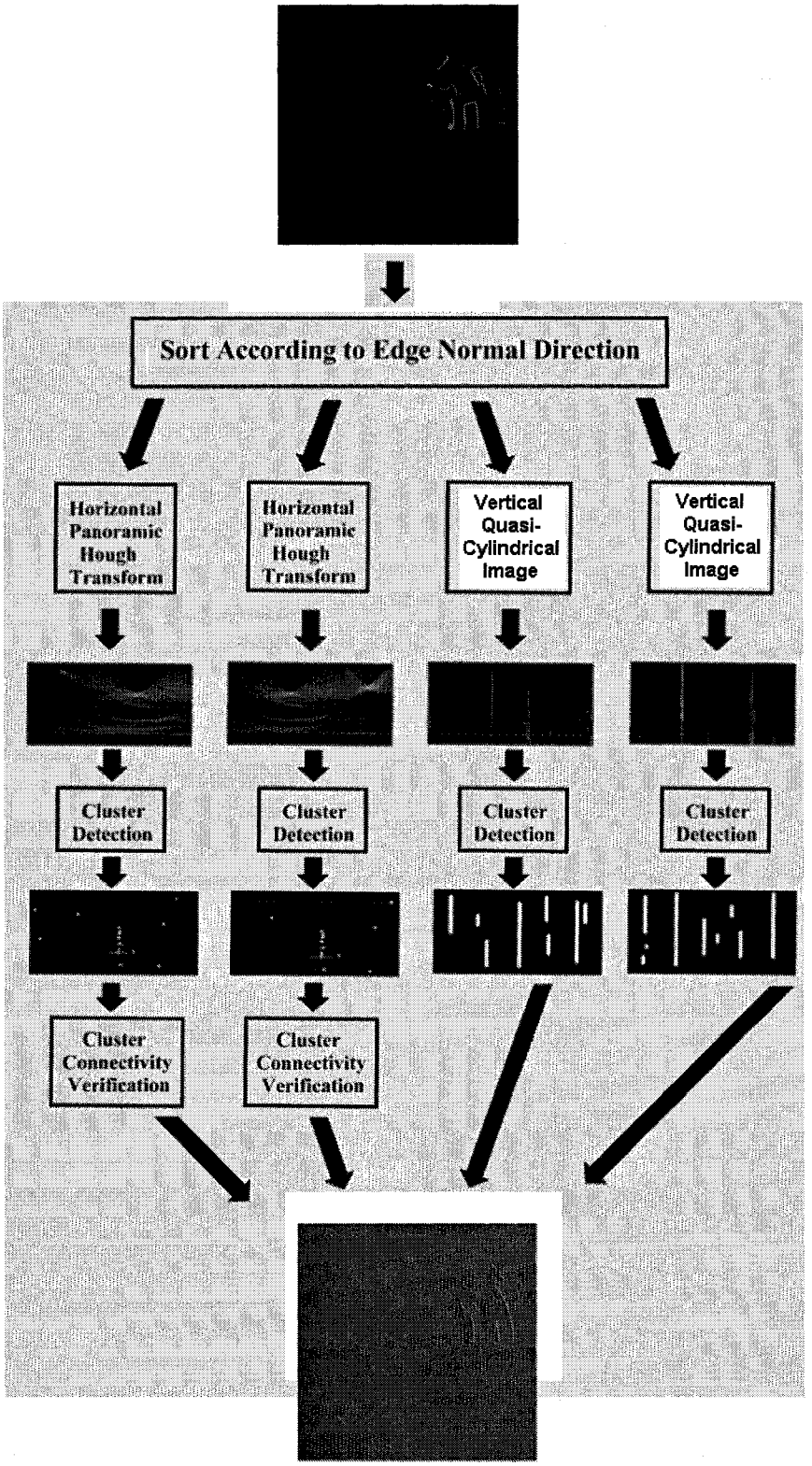


Figure 5.3: *Feature detection processing sub-stages for an image slice.*

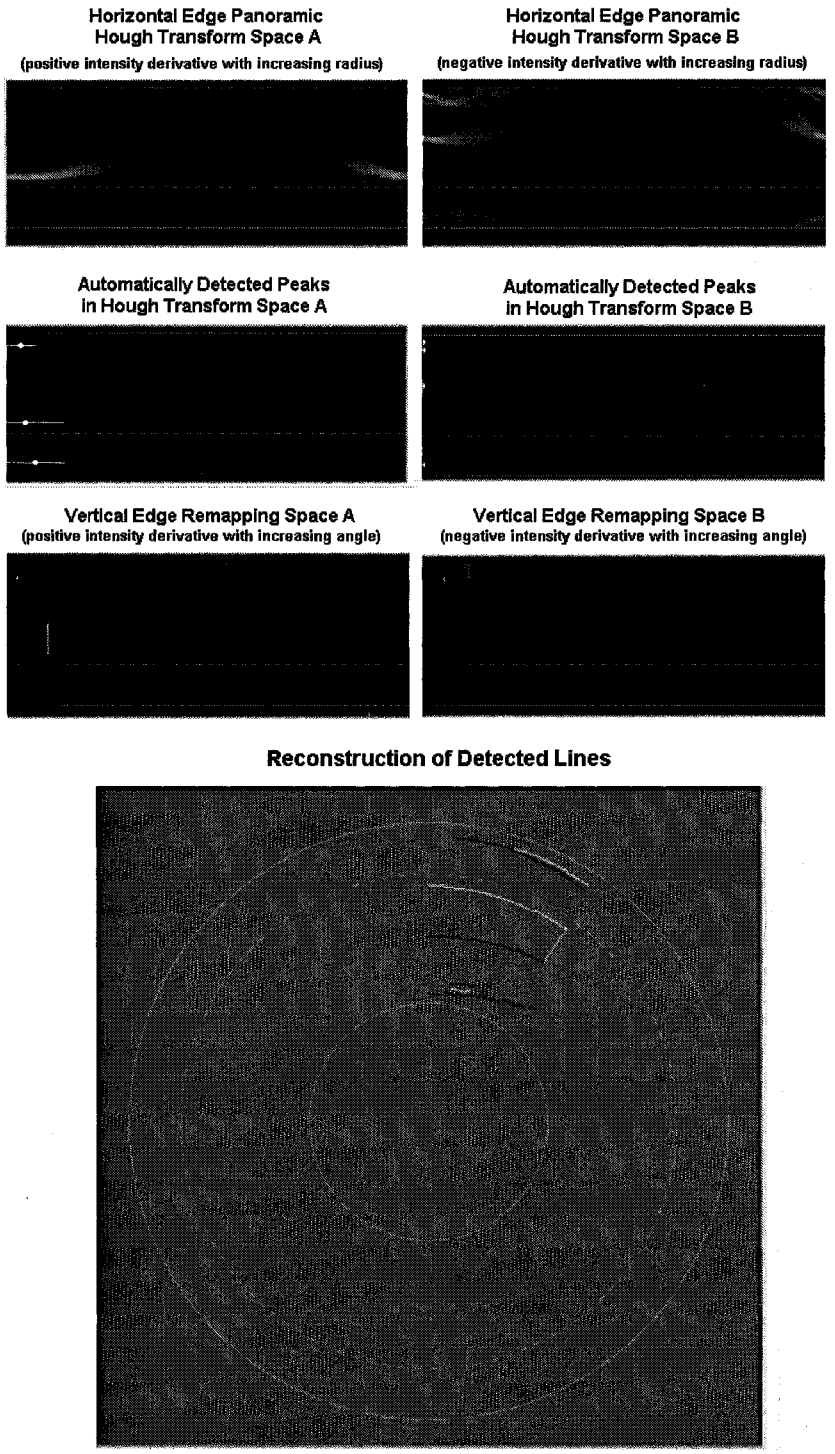


Figure 5.4: *Line segment extraction for the first octant (0-45 degrees). The 2 parameter spaces for horizontal projections and two warped cylindrical spaces for vertical projections (radial lines) have degrees on the X-axis and radius on the Y.*

The data types used through the various stages of the system start as two-dimensional arrays but become feature lists in the earlier stages. Note that the result of the feature extraction is a feature list, and images such as Figs. (5.4-5.11 bottom) and 5.12 are merely a plotted form of this data for demonstration purposes, only the list itself is used by the system. In the case of our experimental implementation, different stages were being performed by different programs, with the feature database transferred with text files. For example Fig. 5.4 (right) is the feature list file extracted from the first octant ($0^o - 45^o$). The file contains other information allowing the image to be reconstructed from this list. More on the details of the working of the actual implementation used in this thesis can be found in Chapter 4.

5.5 Line Segment Tracking

After features are extracted, it is necessary to perform further processing on them to increase robustness and accuracy of the system. Feature tracking is performed to address four recognized error modes: slight position inaccuracy, segment endpoint error, false positives and false negatives. Regarding the detected features as "theories" and reinforcing them with some verification in the image is a good image segmentation technique.

The features extracted and combined so far are the result of finding cluster peaks in the horizontal Panoramic Hough Transform spaces and linear clump detection in the pseudo-cylindrical space images. These are not infallible, and they rely on an ideal camera/mirror model and are likely to have some error. And as is detailed in the error analysis section, small errors in feature position (especially with horizontal line projections) can lead to a very large three-dimensional reconstruction error due to the small baseline. Tracking can fine-tune the location of a horizontal projection line segment or vertical line segment as shown in Fig. 5.13.

The second error mode noticed is that of a line segment being detected, but of an incorrect length. Sometimes the detected segment extends part the way along an edge, 25 percent of the correct length for example. For horizontal line projections, this can be due to the line definition of this feature line extending into a detected cluster elsewhere, and so the pixels can be falsely claimed by the cluster representing another line. This is partly addressed and corrected by heuristics based on predicted error modes of the Panoramic Hough Transform (or indeed any Hough transform technique) as detailed in Chapter 4. An example of this is the bottom line of the rectangle in Fig. 5.13.

As well as inaccuracies in feature location, false positives and negatives can occur. A false positive in this context is where a line segment was reported where an edge does not exist at all. This false feature is where a peak was found in the parameter space that does

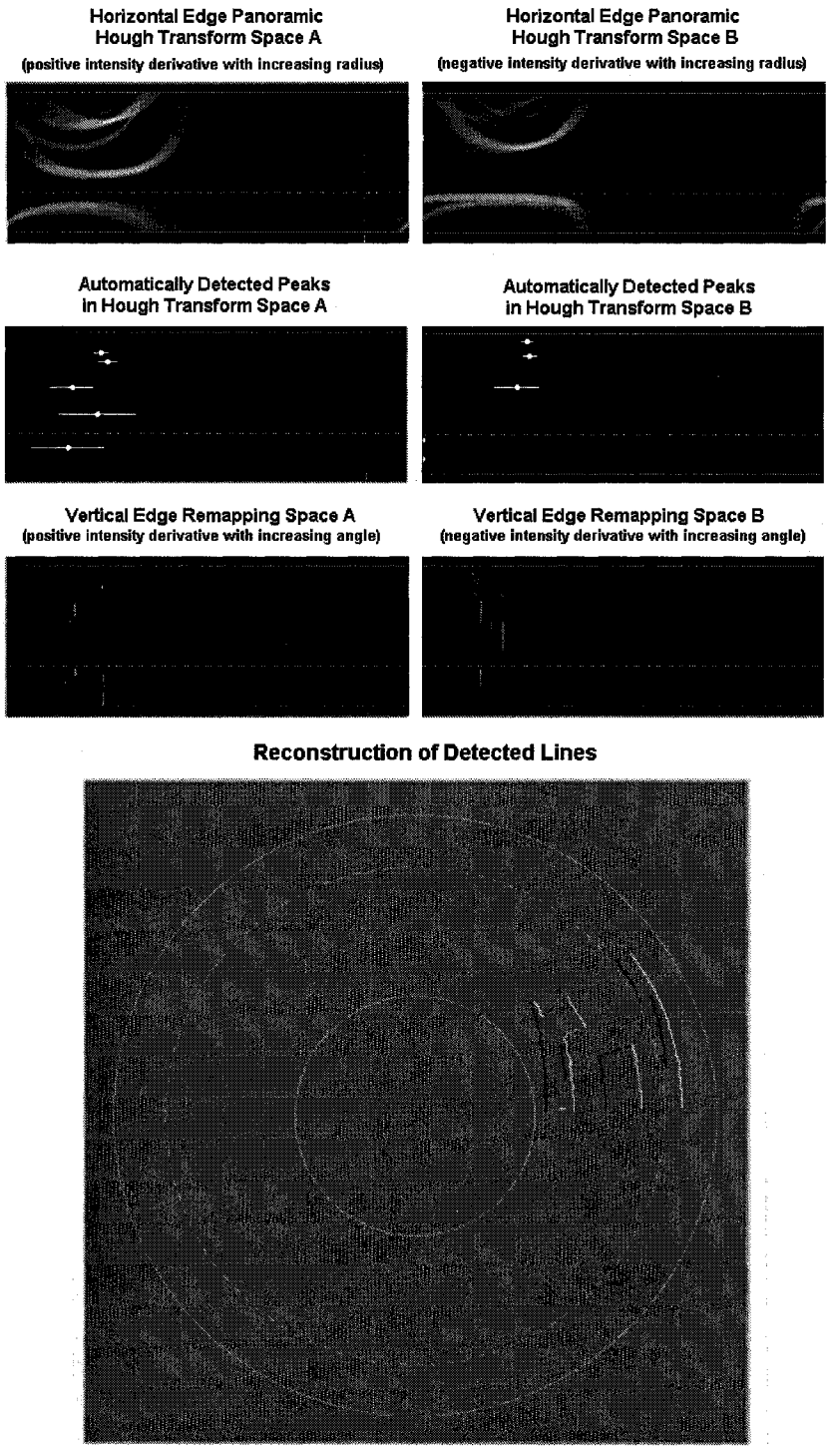


Figure 5.5: *Line segment extraction for the second octant (45-90 degrees). The 2 parameter spaces for horizontal projections and two warped cylindrical spaces for vertical projections (radial lines) have degrees on the X-axis and radius on the Y.*

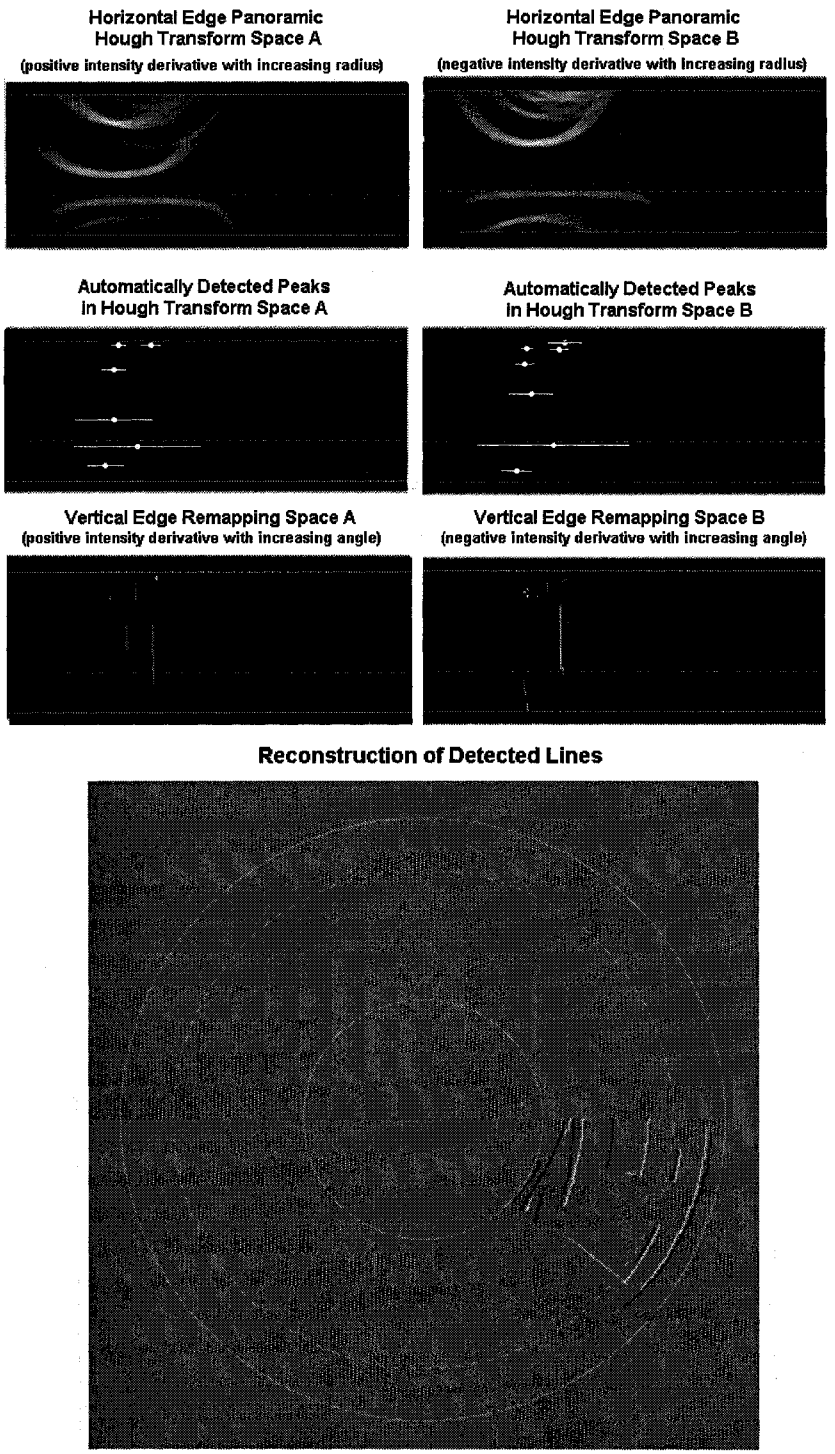


Figure 5.6: *Line segment extraction for the third octant (90-135 degrees). The 2 parameter spaces for horizontal projections and two warped cylindrical spaces for vertical projections (radial lines) have degrees on the X-axis and radius on the Y.*

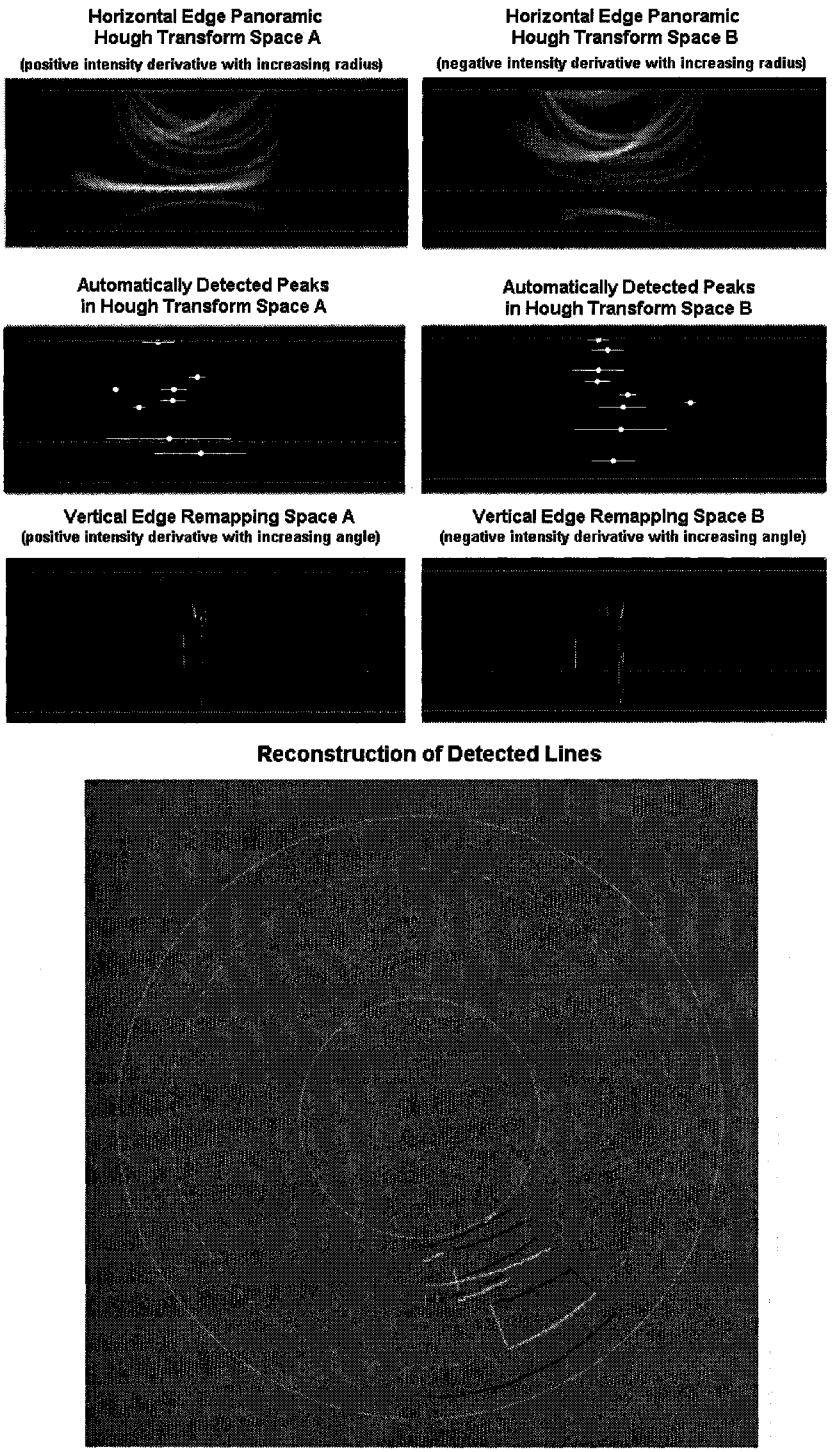


Figure 5.7: *Line segment extraction for the fourth octant (135-180 degrees). The 2 parameter spaces for horizontal projections and two warped cylindrical spaces for vertical projections (radial lines) have degrees on the X-axis and radius on the Y.*

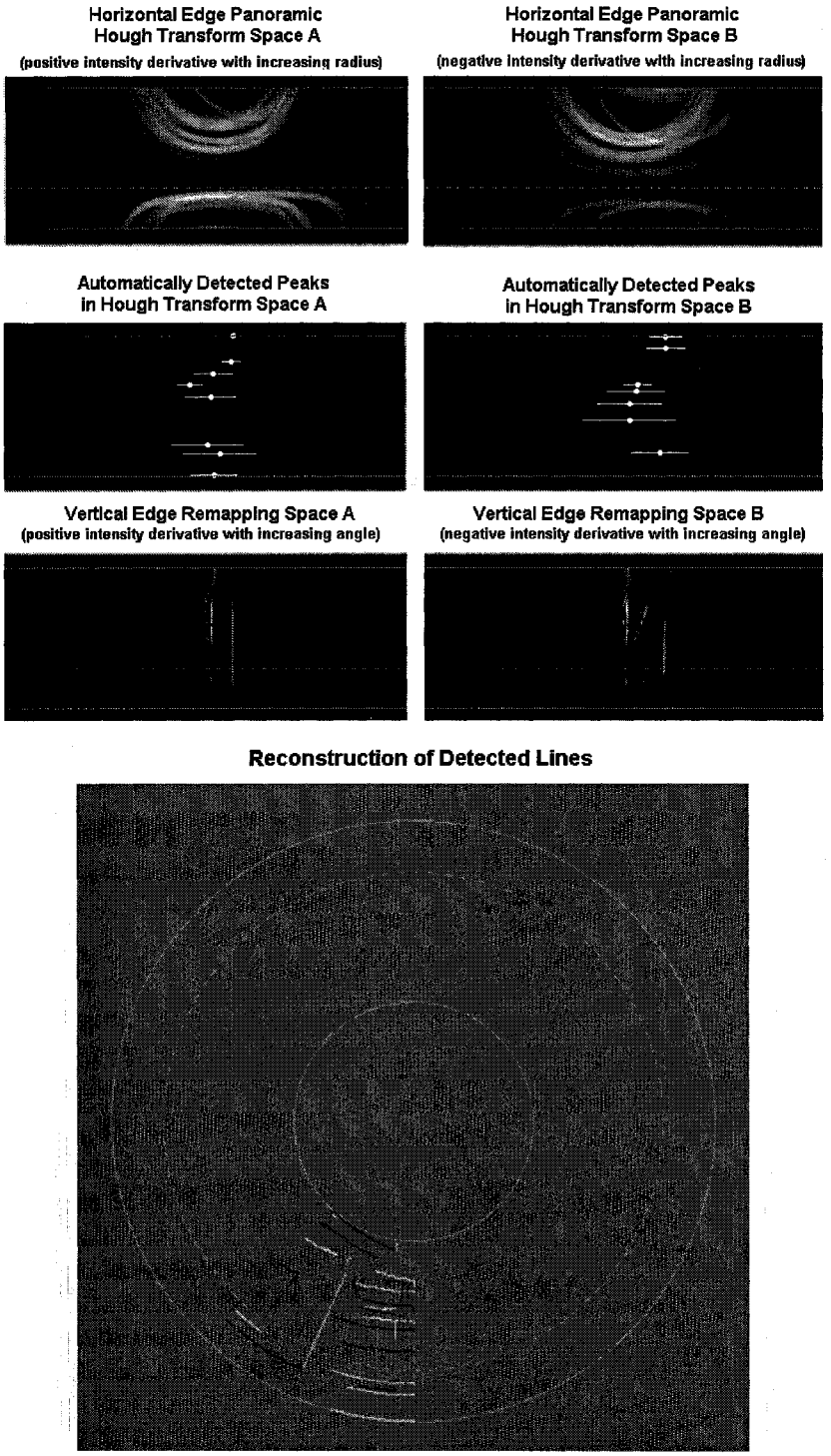


Figure 5.8: *Line segment extraction for the fifth octant (180-225 degrees). The 2 parameter spaces for horizontal projections and two warped cylindrical spaces for vertical projections (radial lines) have degrees on the X-axis and radius on the Y.*

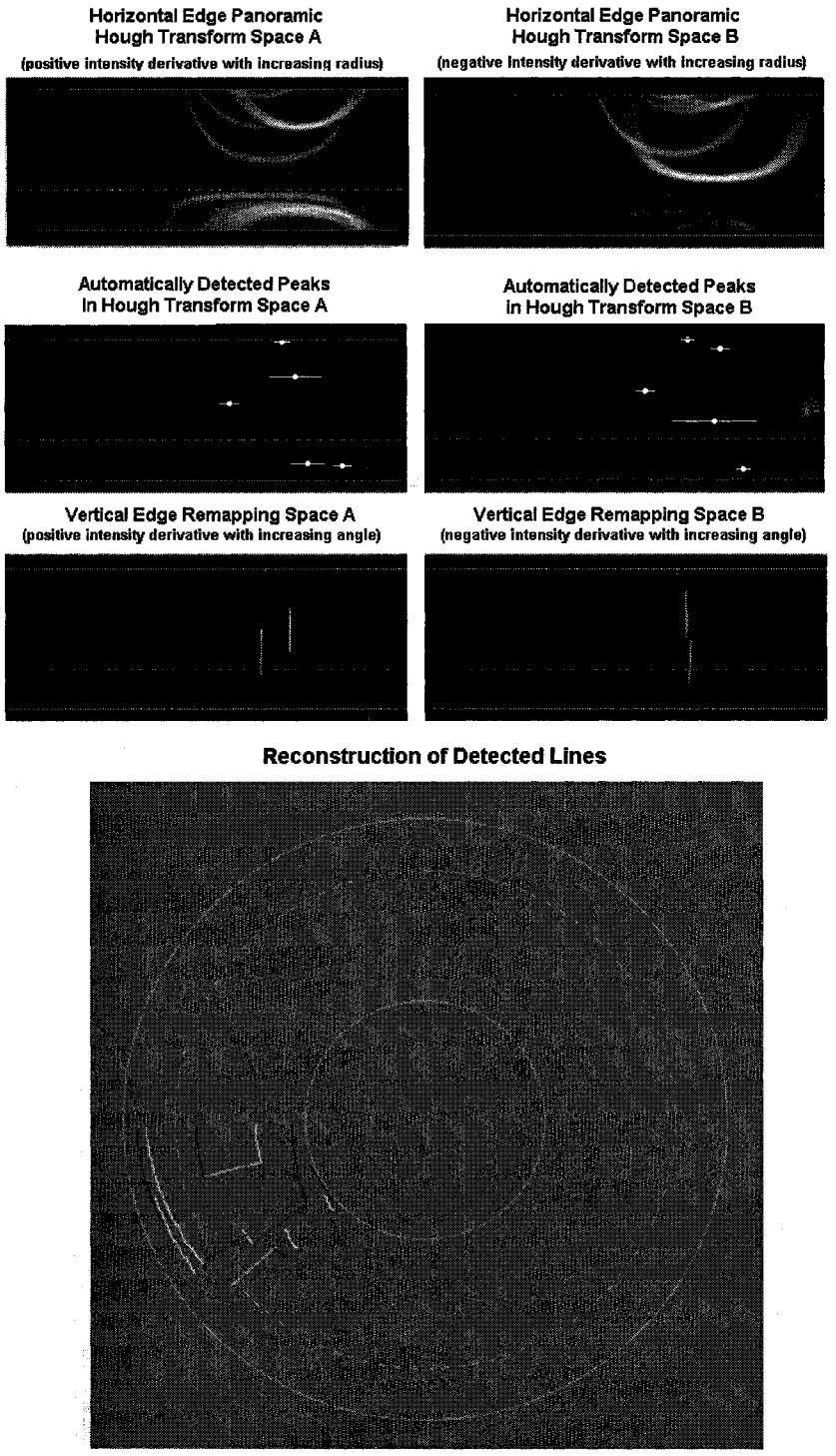


Figure 5.9: *Line segment extraction for the sixth octant (225-270 degrees). The 2 parameter spaces for horizontal projections and two warped cylindrical spaces for vertical projections (radial lines) have degrees on the X-axis and radius on the Y.*

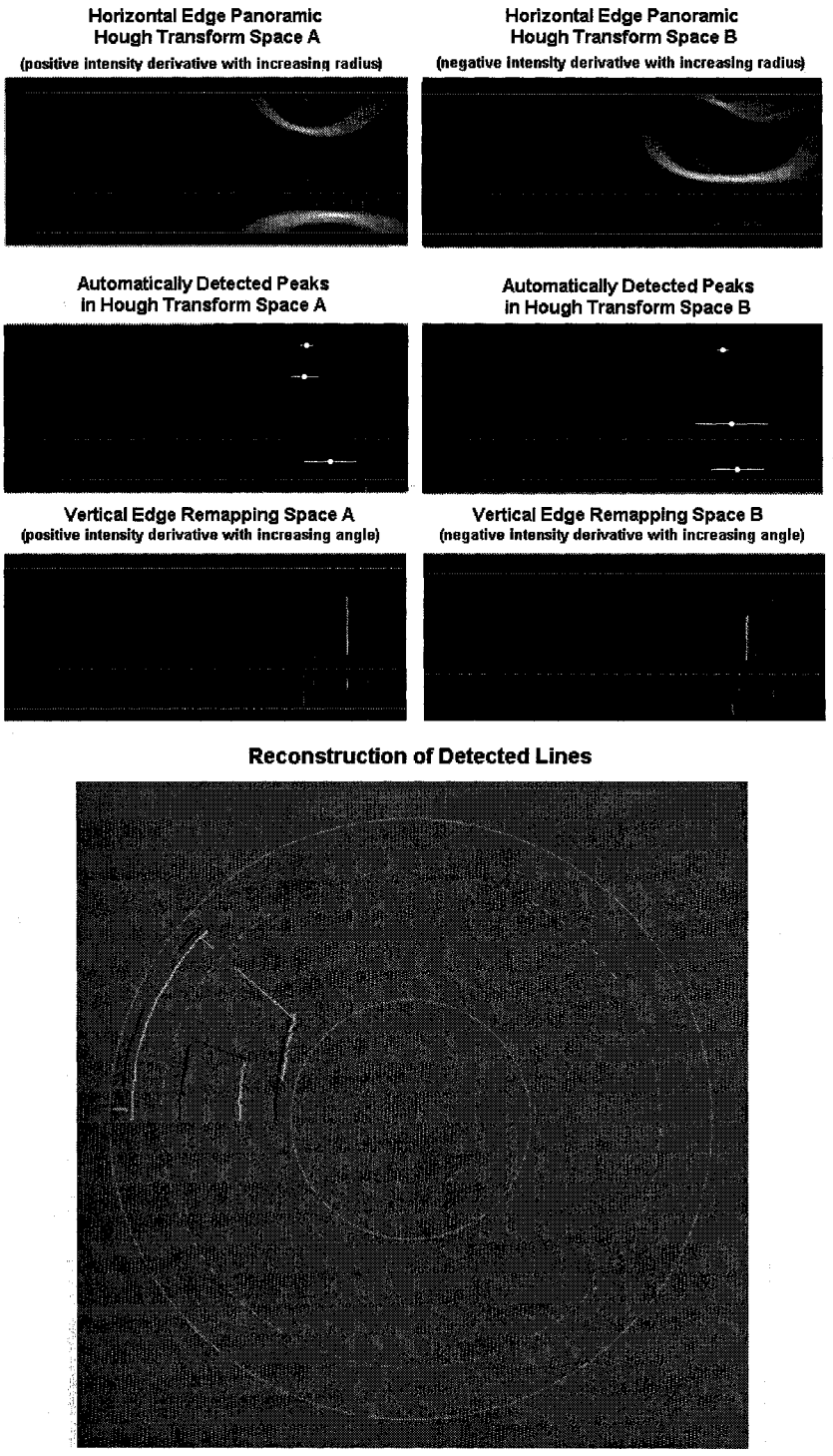


Figure 5.10: *Line segment extraction for the seventh octant (270-315 degrees). The 2 parameter spaces for horizontal projections and two warped cylindrical spaces for vertical projections (radial lines) have degrees on the X-axis and radius on the Y.*

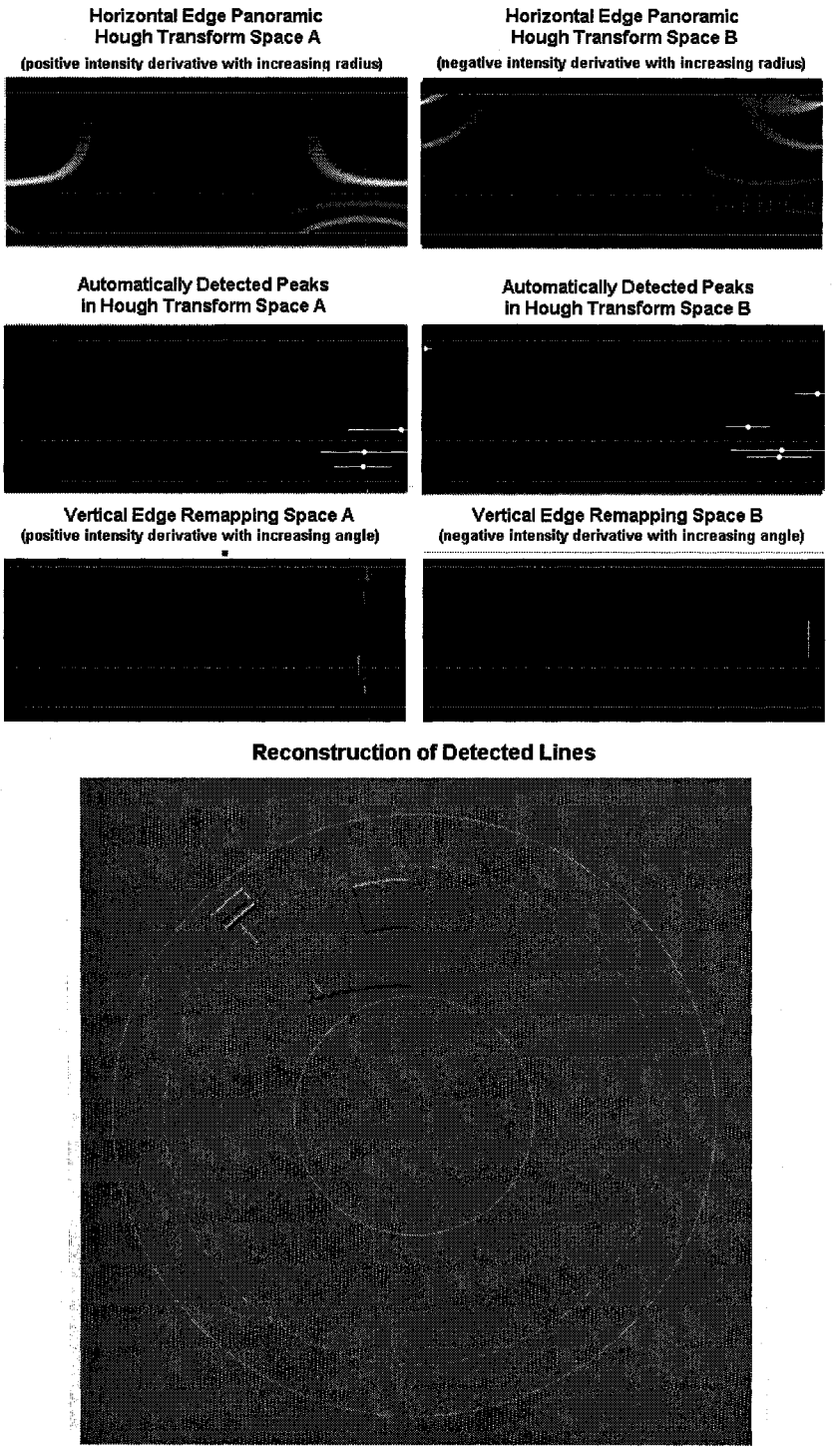
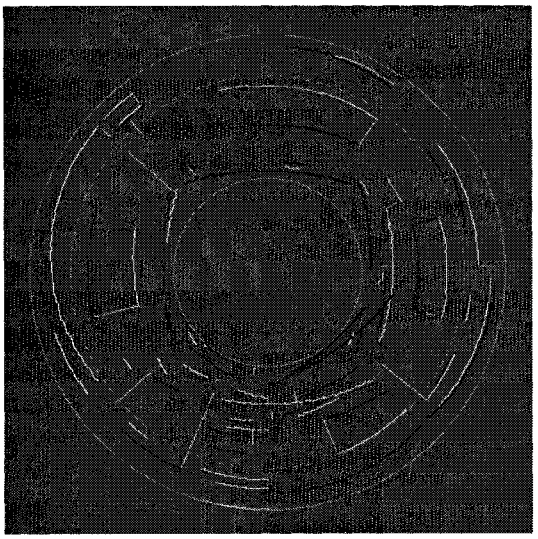


Figure 5.11: *Line segment extraction for the eighth octant (316-360 degrees). The 2 parameter spaces for horizontal projections and two warped cylindrical spaces for vertical projections (radial lines) have degrees on the X-axis and radius on the Y.*



```
//Horizontal and Vertical line segments produced by PANO_SEG.C
//Input image = <in.pgm>, lookup table file = <first_inner_lobe.lookup>
//-------------------------------
width 2160 height 1440
image_centerx 1061 image_centery 744
min_radius 83 max_radius 209 horizon 173
image in.pgm
lookup first_inner_lobe.lookup
//-------------------------------
//--------HA edges - horizontal edges (dI/dR pos)
//ha_seg center_theta spread radius | begin_theta end_theta num_pixels avg_edge
ha_seg 18 35 163 | 0 36 408 34
ha_seg 12 14 94 | 10 18 45 61
ha_seg 25 27 199 | 18 34 138 17
//--------HB edges - horizontal edges (dI/dR neg)
//hb_seg center_theta spread radius | begin_theta end_theta num_pixels avg_edge
hb_seg 0 2 128 | 0 36 346 71
hb_seg 0 2 89 | 0 43 352 60
hb_seg 0 2 96 | 9 19 67 120
hb_seg 0 2 199 | 5 34 281 24
//--------VA edges - vertical edges (dI/dT pos) cw
//va_seg theta begin_radius end_radius | num_pixels avg_edge
va_seg 36 136 165 | 0 0
//--------VB edges - vertical edges (dI/dT neg) cw
//vb_seg theta begin_radius end_radius num_pixels avg_edge
```

Figure 5.12: *(left) The result of merging the feature data sets from the 8 radial sub-images. Projections of horizontal line segments are combined if they end on the sub-image boundary and satisfy some likelihood criteria. (right) A sample .fea file for the first octant's extracted features which are redrawn in the right side of Fig. 5.4.*

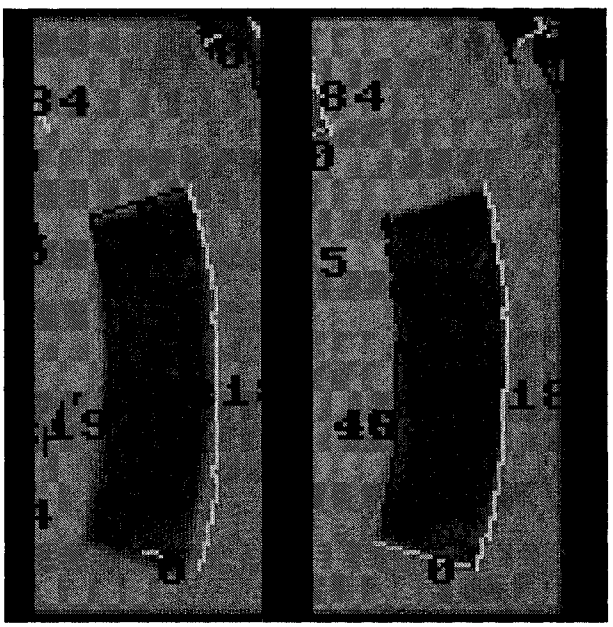


Figure 5.13: *Example of using tracking to fine-tune inaccuracies in features extracted by the transform methods. Before (left), after (right). Note the borders of the the black rectangle's projection before and after. The top, left and right sides demonstrate the first error mode, that of correct identification but inaccurate position. The bottom line demonstrates the correction of the second error mode, incomplete segment length.*

not originate from a real feature. There are several ways this can occur, the coincidental convergence of the the skirts of several clusters, or more likely the removal of two small a region of parameter space points when a cluster is blurred due to bad calibration or mirror imperfections. False positives can be corrected since the tracking algorithm will find no such edge and it will be eliminated from the list all together.

A clear parameter space cluster may not be recognized, or noise might obscure a vertical line projection in the quasi-cylindrical vertical line space causing a legitimate feature to not be reported. Tracking cannot fix the failure mode of false negatives.

The operation of the tracking procedure is shown briefly in Fig. 5.14 (left) for one "type-B" horizontal edge (intensity decreases with increasing radius corresponding to a horizontal edge darker at a lower elevation). Linear samples are taken along the predicted edge location. These samples are processed to find the step position of the correct polarity, and the position updated with the PHT. The endpoints are fine-tuned in the image by iteratively halving the distance between the last detected and undetected step edge location. This procedure is described in full in section 9.6.1. This tracking is performed for all features.

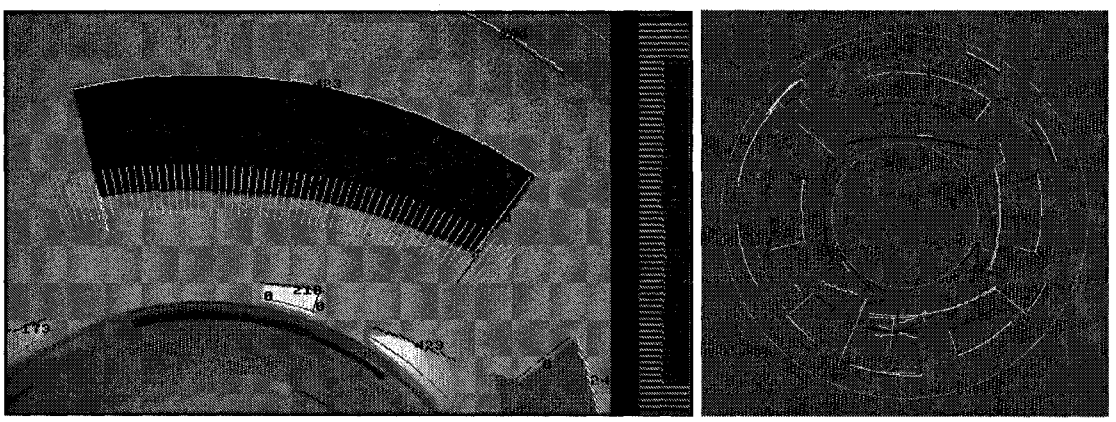


Figure 5.14: *Demonstration of edge tracking on one horizontal edge projection feature (left). The feature is defined by the horizontal line projection parameters* $(R_{main}, \theta_{main})$*, and the beginning and end angles* $(\theta_{begin}, \theta_{end})$*. All four of these are updated with tracking. Linear radial samples are taken at points along the predicted locus (the linear samples are redrawn at the right edge of the left image). These linear samples are examined for presence and location of a step edge of the correct polarity. These allow the recalculation of the horizontal line projection parameters* $(R_{main}, \theta_{main})$*. New samples are placed at positions of iteratively halved locations between samples that report the presence and absence of step edges, finding the new start and end angles (shown with a prominent white and black radial line marker for the start and end angle respectively). The resultant features after this tracking are redrawn (right image).*

5.6 Addressing False Negatives: Hypothetical Rectangle Completion

The previous tracking procedure repairs much of the incorrect feature reporting from the feature detection for three of the four error modes. The false negative error mode was not addressed and some line segment projections that exist in the image are not reported. These errors are corrected in some cases by means of creating additional hypothetical line segments to close rectangular region projections that are partially defined by the existing segments. This is done by examining the feature list, and looking for cases where two or three line segment projections indicate a possible closed shape. A set of topologically motivated rules assuming a polyhedral world are applied to the list of features. Where two parallel segment features exist with compatible endpoints, features corresponding to the two open sides of an imaginary rectangle are added to the output feature list. Likewise when three line segment projections meet of correct types (lighter or darker inside), and the open endpoints match within a threshold, the final side is proposed and the hypothetical missing feature line added to the output feature list.

The stage of hypothetical feature creation is followed by a second tracking stage, which removes the incorrect hypotheses. Most of the hypothetical features will not be found in the image and hence removed. The ones remaining correct mistakes of the forth error mode, false negatives. This is demonstrated in Fig. 5.15.

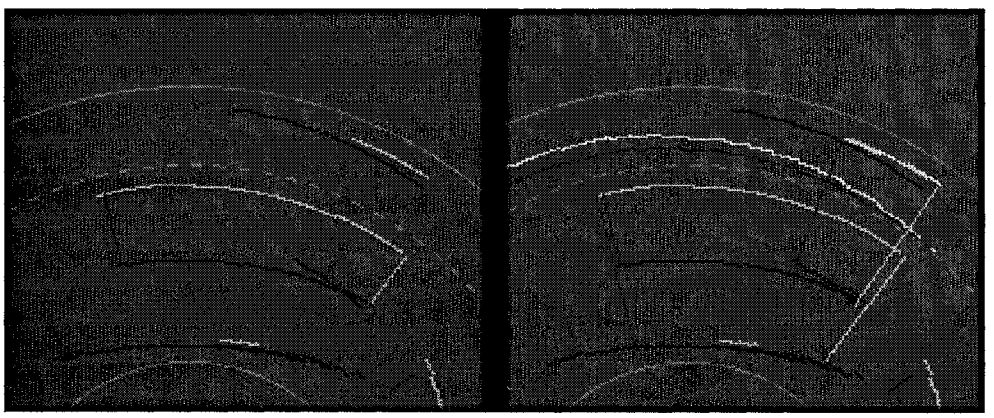


Figure 5.15: *Hypothetical features added to close potential rectangles. This step assumes a rectilinear polyhedral world and attempts to address the false negative feature detection error mode. Most hypothetical lines will not exist and will be removed in the subsequent tracking stage.*

Note that the hypothetical line segments are proposed by examination of the feature list only, and the actual image is used just for verification. In this way a conceptual parallel can be drawn between the original transform based feature extraction and this hypothetical

rectangle completion. In both cases a set of suggested features is proposed by examining some sort of database, and verified by examination on the actual image (tracking). The first uses the PHT and vertical segment algorithm processing output from sorted edgels to make feature conjectures while the latter uses the existing feature list.

5.7 Rectangular Region Detection

At this point the feature list is examined for closed rectangular regions, determined by examining feature end-points. The functionality is very similar to the hypothetical feature topology rule based procedure. If a set of horizontal and vertical line segment projections of correct types can be found to meet in a closed chape, they are assumed to be the projection of a vertical rectangular polyhedral face. The four segment features are removed and replaced with a single vertical rectangle feature.

The vertical rectangle projection is a new feature type, and is defined by five main parameters: the perpendicular angle of the plane containing this rectangle (θ_{main}), the inside and outside radii (r_{inner}, r_{outer}) and the beginning and end angles ($\theta_{begin}, \theta_{end}$). Additional parameters of the spread (uncertainty) of θ_{main} and the total edgel count of the border are reported in the feature list also. The parameter θ_{main} for this new rectangle is found from the θ_{main} parameters of the two horizontal line segment projections by taking into account their respective spreads (uncertainties) as in Eqn. 5.1 below. The spread parameter of the contributing segment feature is derived from the width of the original cluster peak, a larger spread indicates a larger uncertainty in the segment's θ_{main} value. Eqn. 5.2 defines the spread parameter of the new rectangle. Neither Eqns. 5.1 nor 5.2 are justified theoretically herein, they are proposed based on the desire to use the contributing information proportional to their confidence, and combine the confidence values in a way that roughly defines an estimate for the resultant confidence.

A less accurate but more economical alternate strategy to Eqn. 5.1 is simply to take the θ_{main} and spread value from the horizontal segment projection feature with the smaller spread value, i.e. taking the more accurate of two observations.

$$\theta_{main-rect} = \theta_{main_1} \frac{spread_2}{spread_1 + spread_2} + \theta_{main_2} \frac{spread_1}{spread_1 + spread_2} \tag{5.1}$$

$$spread_{rect} = spread_1 \frac{spread_2}{spread_1 + spread_2} + spread_2 \frac{spread_1}{spread_1 + spread_2} \tag{5.2}$$

The start and end angles of the combined feature are taken from the vertical line projection features θ. Since this angle is calculated from the average of many linear samples

taken orthogonally to this edge, a more accurate angle can be obtained. Indeed because of this averaging of many readings, sub-pixel accuracy is possible.

Combining the θ_{main} parameters in this way creates an increased average confidence value for the rectangle features than the average of the segment features themselves since the result will benefit from the more accurate one. A typical example is a polyhedral face, the bottom edge of which is close to the horizon line and hence almost indeterminate θ_{main}, and the top edge with a more distinct, tighter cluster peak. Eqn. 5.1 allows the rectangle to be defined mostly by the top, more accurate edge. The feature lists of the second phase of feature tracking (after testing the hypotheses), and that of the vertical rectangle detection are redrawn below in Fig. 5.16.

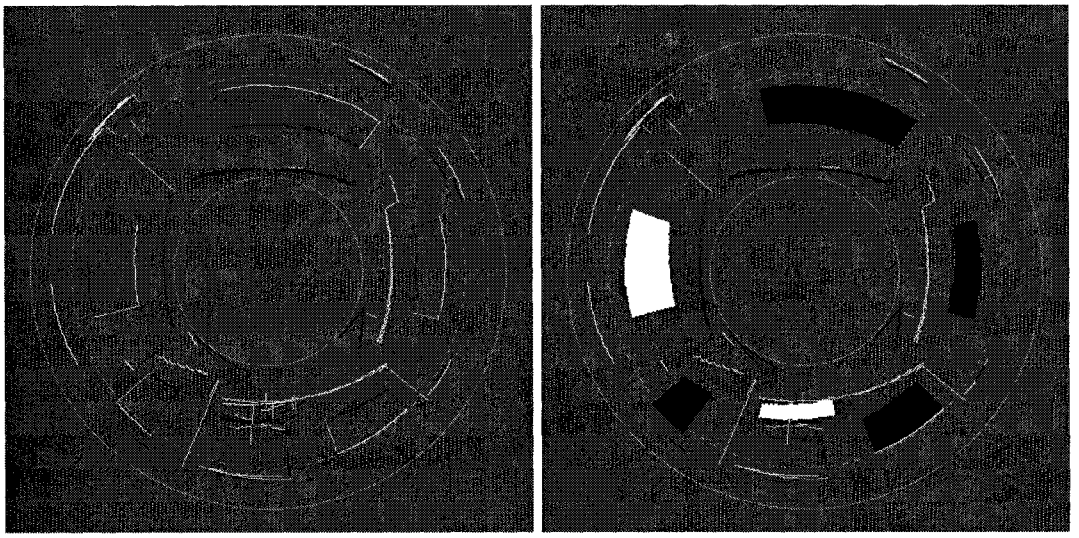


Figure 5.16: *Result of testing rectangle hypotheses (left). Subsequent detection of rectangular regions (right).*

False positives for the new feature type (vertical rectangle projections) can occur, the smaller white rectangle at the bottom of Fig. 5.16 is erroneous and declared because of the presence of four bordering line segment features. To be consistent, this feature should be verified on the image itself, perhaps examining the pixel intensities in the proposed region for uniformity. The implementation in this research did not perform this verification stage, since the objective is stereo reconstruction from two lobe views and it is presumed that a matching false positive rectangle feature will not be found in the other lobe. Possible future work could increase system robustness by performing the above check.

5.8 Feature Pruning

The process flow depicted in Fig. 5.1 does not show the two stages of feature pruning. Features are removed from the list both in the combining stage, and as a separate stage after the rectangular region detection. Pruning a feature set refers to examining and removing features which do not meet certain requirements. The pruning stage in the feature combining removes segments shorter than a preset threshold length.

The pruning stage after the rectangle detection removes all segment features whose length is below a larger threshold unless one of the endpoints meets a perpendicular segment. In this way small corners are preserved but floating short segments are pruned. This is based on the philosophy of only considering features that are consistent with a polyhedral world.

5.9 Summary

This chapter stepped through the phases of feature extraction from one viewpoint, the reflection from one mirror lobe. The process started with an image, a two-dimensional data set of intensity values, and ended with a list of features. These features are the estimated projections of horizontal line segments, vertical line segments, and vertical rectangular faces. For stereo reconstruction, this feature list is computed for two mirror lobes. A three-dimensional model is created by matching these feature sets.

Chapter 6

Reconstruction Accuracy and System Calibration for Classic Stereo

After features have been identified and matched between different views in a stereo vision system, the final step in model making is to reconstruct three-dimensional points and assign some measure of confidence to them. Much attention to detail must be paid to this step otherwise the resultant reconstruction can be fraught with error. Accurate camera calibration is required posing practical challenges. Even if perfect calibration could be achieved, the nature of the geometry and image quantization sets limits on the accuracy of 3D point recovery.

The classic perspective projection uses the concept of a focal point, a single viewpoint (SVP), a point through which all rays must pass. Reconstruction and calibration issues are first discussed in this chapter for binocular stereo using standard dioptric cameras that approximate a pinhole perspective projection. Chapter 7 extends this analysis to the non-SVP panoramic sensors.

6.1 Traditional SVP Multi-Camera Stereo Geometry

Reconstruction is the task of finding a 3D scene point from a set of image points. For a binocular stereo system (using two cameras), reconstruction is a function that provides the coordinates of a point (X, Y, Z) from two image points (U_1, V_1), (U_2, V_2), each from a separate image. The image points (U_1, V_1), (U_2, V_2) are deemed to be projections of the same scene point as seen by the different cameras, Camera 1 and Camera 2 respectively. The four numbers defining the two image points are mapped to only three numbers for the 3D coordinates, there is a redundancy and not all possible sets of (U_1, V_1), (U_2, V_2) can define a 3D point. This can be more clearly expressed by noting that each image point (U, V) defines a ray in space passing through the camera focal point and the point on the image plane. This ray has two degrees of freedom. All 3D points along this ray are projected onto this image point. With two cameras the rays associated with an image point on each must intersect if they are indeed imaging a real world 3D point. Considering a given point (U_1, V_1) and the ray this represents, only a subset of rays from Camera 2 can intersect this ray, reducing the freedom for (U_2, V_2) to one dimension. This is known as the *epi-polar constraint* and is useful for matching image points. As seen in Fig. 6.1 if a feature point is found at (U_1, V_1), it must exist (if it is not occluded) somewhere on the epipolar line in the second image. Zhang and others [118] demonstrate a system for finding the epi-polar matching from two cameras of unknown poses through iterative numerical means. In practice, due to quantization and finite accuracy, the rays from points (U_1, V_1) and (U_2, V_2) will not meet exactly and so reconstruction involves finding the 3D point closest to both rays.

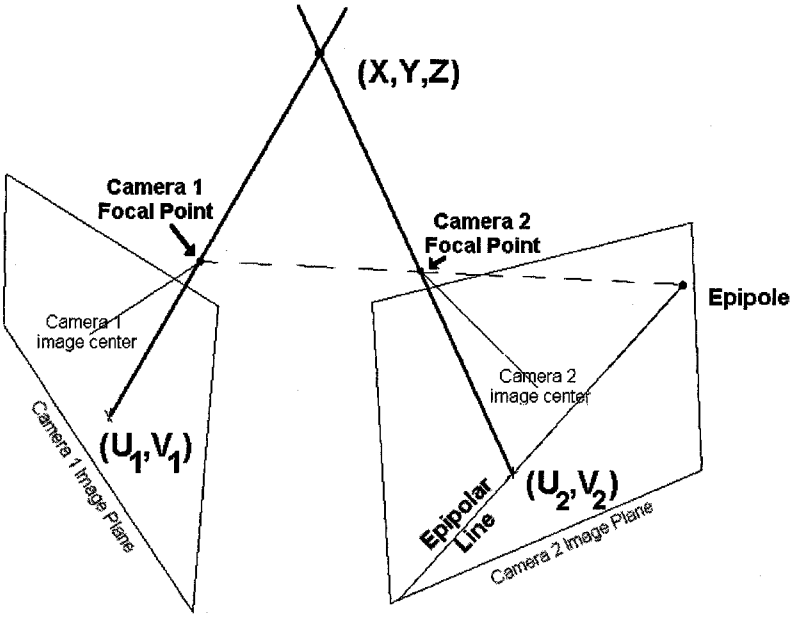


Figure 6.1: *Classic two-camera stereo geometry. A set of matching image plane points (U,V) define rays which intersect at the 3D scene point (X,Y,Z). The epipolar constraint is shown, useful for matching.*

Fig. 6.1 demonstrates the general case, with a *non-rectified* geometry, and in many cases the images are re-projected so that new virtual image planes are coplanar. This moves the epi-poles to infinity (projections of each focal point in the other camera's image plane) and the epipolar lines become the same rows between the images. In all cases the distance between focal points, i.e. viewpoints is called the *baseline*. In *rectified* geometry, the angle between the main camera axis becomes parallel and the axis become perpendicular to the baseline. Typically the rectified image U axis is defined as being parallel to the baseline. In *rectified* images [91] the depth is defined as the distance of a point from the baseline. The depth can be determined from the *disparity* which is a measure of the image distance a feature 'moves' between the two rectified images. If the depth is infinite, the disparity is zero. Assuming identical camera focal lengths F, camera baseline B Eqn. 6.1 defines the depth as a function of disparity D.

$$depth = B\frac{F}{D} \tag{6.1}$$

The reconstruction of the 3D point coordinates is however only half the battle since if this information is to be used in a practical manner, the uncertainty of this reconstructed position should be found. This accuracy information is important when combining multiple scenes into a larger model, such as if this sensor was places on a mobile robot for example.

The largest error for stereo vision on mobile robots is usually in range, (in this case the distance from the baseline) since the distance between cameras on a mobile robot is typically smaller than the scale of objects in the environment. The distance error, expressed as a percentage of the correct range is given the most attention in this and the next chapter.

Fig. 6.2 and Eqn. 6.2 describe the percentage error d_{error} for an error $d\theta$ in the angle of one of the two rays. The range error can either be calculated with $d\theta$ towards or away from the other camera's main axis, the latter gives a large error measure and so it used in these calculations.

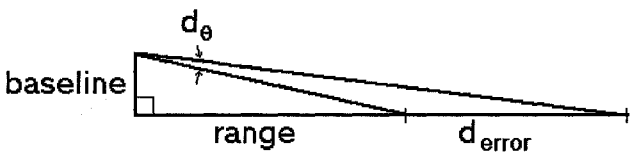


Figure 6.2: *Simplified distance error in stereo.*

$$r = \frac{range}{baseline}$$

$$d_{error} = 100\% \cdot \left(\frac{tan(tan^{-1}(r) + d_\theta)}{r} - 1\right) \tag{6.2}$$

The effect of a small $d\theta$ on the range has two ramifications, the first being that there is a potentially large area of uncertainty in range, even with correctly identified image points (U, V), and the second being a large error if the feature extraction is even a few pixels off. Due to the quantization nature of digital images, a pixel coordinate describes not a ray but a thin volume of 3D space. The intersection of the volumes from two views with a narrow *vergence angle* (the angle between focal points as seen from the 3d world point, ideally 90°) is a long, narrow volume of space.

The angle a pixel subtends is a function of the image resolution and focal length. If the Canon D30 digital camera (Chapter 7) with a 50 mm lens was used without a cataptric component (mirror), the horizontal resolution of 2160 pixels and the focal length of 6050 pixels, yields a $d\theta$ of 0.009°. Fig. 6.3 (left) shows the percentage distance error as a function of the range/baseline ratio if we used two of these camera/lens assemblies in a classic binocular stereo system. With a 10 cm baseline and a range of 1 metre, we would have a depth uncertainty of 1.5% = 1.5 cm. At a distance of 50 metres, the uncertainty would be 6.7% = 3.3 metres. This highlights the weakness of stereo vision. A small error in feature detection and matching can yield a large error in position estimation, even with a very high image resolution and narrow field of view. Fig. 6.3 (left) plots this relationship for a (U_1, V_1), (U_2, V_2) mismatch of 1,2 and 3 pixels for this camera with an image resolution

of 2160 pixels on a side. Fig. 6.3 (right) repeats this for a stereo pair with the same field of view (zoom), but with an image resolution of 1024 pixels on a side. The error for 1 pixel can also be interpreted as the length of the range of uncertainty, anywhere in that range a point feature will project onto the same pixel in the image plane.

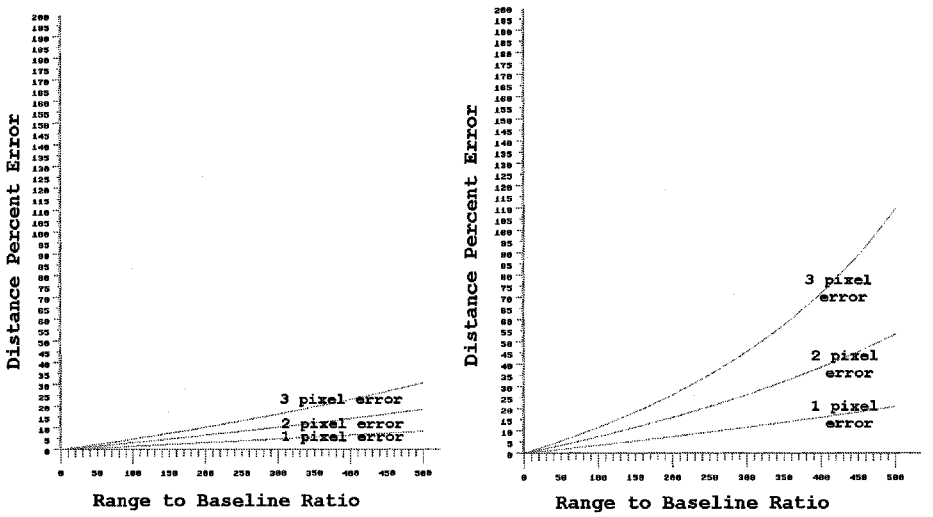


Figure 6.3: *Simplified distance error in stereo. Left: 2160 x 2160 resolution, right: 1024 x 1024 resolution.*

Fig. 6.4 provide plots for an image resolution of 640x640 pixels (left) and 320x320 pixels (right). If a binocular stereo system, with a resolution of 320 pixels, had a baseline of 2.5 cm, it would have a potential error of 51% (1.3 metres) at a range of 2.5 metres, if the feature extraction and matching was incorrect by 3 pixels. And all of the above discussion is with a narrow field of view of 19.6°, the situation would be worse for a camera with a smaller zoom factor.

As will be shown later, the stereo panoramic sensor built for the experiments in Chapter 8, has an equivalent baseline of about 2.5 cm, the effective image resolution of the inner lobe is only about 126 pixels with a field of view of more than 45° of elevation so one should not expect a highly accurate 3D reconstruction. This is the trade-off we should expect, since the panoramic stereo system provides such a sweeping panoramic stereo view with one image sensor. Because of the pixels being spread over a large section of a spherical viewing volume, the confidence of our 3D reconstruction will have modest limits even if the sensor resolution is high as in our system.

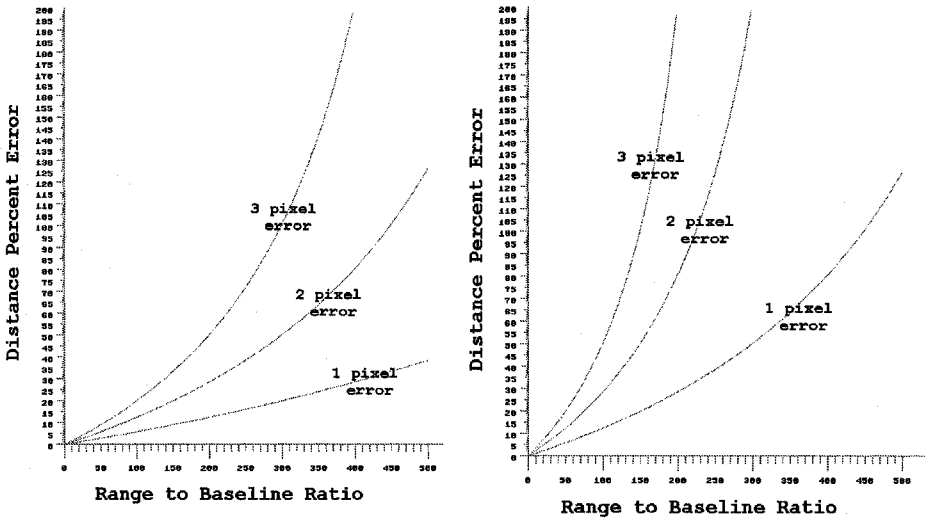


Figure 6.4: *Simplified distance error in stereo. Left: 640x 640resolution, right: 320x 320 resolution.*

6.2 Perspective Camera Calibration

Note that the accuracy discussion of the previous section assumes that we have an accurate mapping between an image point (U, V) and a ray in space. Determining the relationship between points in camera images and point in 3D space is known as *calibration*. Various methods have been used by researchers to find this calibration information, and can be classified by whether or not they rely on the pinhole perspective projection model. If this model is used, the image formation can be described by *intrinsic* and *extrinsic* parameters. Without considering *radial distortion*, represented as intrinsic parameters, the other parameters can be combined into the standard equations sometimes referred to as the *Essential Matrix*. If the image formation can be modeled in this way, the 3D ray for each camera image plane pixel can be found. Alternatively, the *two-plane* calibration scheme described below allows finding this pixel-to-ray mapping directly without relying on a perspective projection model. It however does not offer the flexibility the above methods do if the camera is often moved.

6.2.1 Perspective Camera Calibration: Intrinsic and Extrinsic Parameters

Extrinsic parameters define the pose of a camera, i.e. its position and orientation. There are six degrees of freedom for the general case. The intrinsic parameters are particular to the camera itself, and need only be found once if the focus and aperture are fixed.

The principle intrinsic parameters are the focal length, and the aspect ratio which is a

function of the image sensor pixels' width to height ratio. The Canon D30 we used has square pixels and hence this ratio is unity. The other commonly identified intrinsic features are those which measure the *radial distortion*, a phenomenon that appears when using lenses (dioptic cameras).

Most dioptric cameras, for example standard film, video and digital cameras, are all designed to approximate a pinhole perspective projection. A camera can be constructed with merely a small hole and an image plane, however not much light will be captured and the exposure time will be long. Lenses are introduced to replace the pinhole and capture more light but emulate a pinhole perspective projection. However, they are intrinsically only an approximation of a pinhole, and produce blurring at certain ranges and introduce the effect of radial distortion. Lenses introduce the concept of depth-of-field (DOF), which means that the camera has to be focuses for a specific depth unlike a perfect pinhole camera where objects at all depths are simultaneously in focus. The DOF is a range of distance at which the blurring is smaller than the minimum pixel size, or emulsion particle size in film cameras. An ideal pinhole camera would have infinite depth of field, all objects near and far would be in sharp focus, but would not capture much light due to an infinitely small pinhole!

Fig. 6.5 demonstrates some examples of images with severe radial distortion, captured from an underwater robot system built by the author for the University of Alberta in 1995. The image can be warped to approximate a correct perspective projection if these intrinsic parameters are known. The qualitative effect of radial distortion is a "fish eye" effect where pixels further out are "bent" inwards. Many image straight lines, especially around the edge of the image, become curved. Addressing radial distortion is especially important for use with low cost off-the-shelf cameras since the view is typically very curved. Radial distortion by definition is an effect by which the image is distorted by a remapping of the radius of points relative to a central image point. If an image point's coordinates are expressed in polar coordinates, its equivalent position in a perspective projection image can be found by modifying the radius with a one-dimensional function $r_{correct} = F(r_{distorted})$. This function is typically approximated by a 3rd order polynomial, and hence an effective perspective projection can be obtained by warping the input image with the knowledge of the center of distortion (X_{cw}, Y_{cw}) and the three polynomial coefficients. Radial distortion can be considered an image plane phenomenon, independent of object depth and so once a mapping has been found, it can be used for all images thereafter. The two corrected images in Fig. 6.5 were obtained with the same mapping.

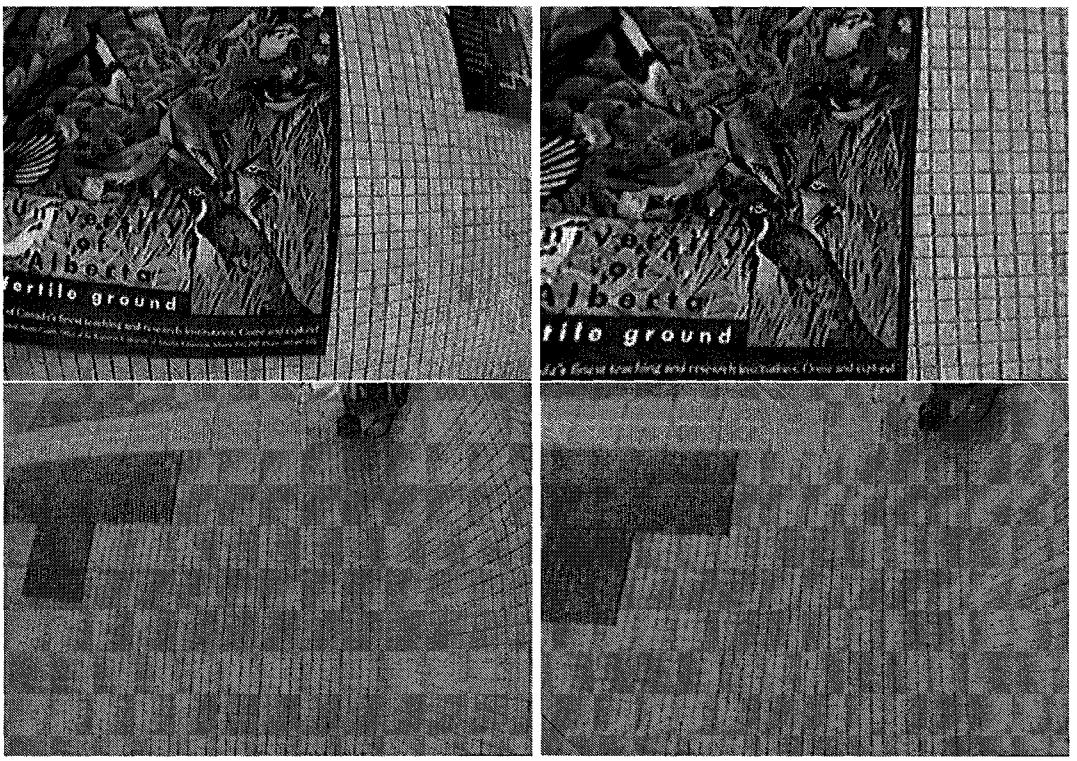


Figure 6.5: *Examples of radial distortion from images from an underwater robot where the air-water interface created a large radial distortion effect. The left images show the original images, the right show the images after radial distortion correction has been applied. Once the radial distortion parameters have been found, the same correcting warp can be applied to all images from the camera.*

6.2.2 Perspective Camera Calibration: Essential Matrix

A term used often in computer vision is the *essential matrix*, which is the mapping between a 3D point and a point on the image plane of an SVP perspective camera. It is in essence just the combination of the basic perspective equations $U = \frac{X}{Z}$, $V = \frac{Y}{Z}$, with the conversion of coordinates between camera and world coordinate spaces. The geometry below assumes a perfect camera with no radial distortion, and assumes square pixels. A camera with radial distortion can be used, if an corrective warped is first applied.

The intrinsic parameter of focal length and the extrinsic parameters of camera center and orientation are used below.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} i_{xx} & i_{xy} & i_{xz} \\ i_{yx} & i_{yy} & i_{yz} \\ i_{zx} & i_{zy} & i_{zz} \end{bmatrix} \begin{bmatrix} X_w - C_x \\ Y_w - C_y \\ Z_w - C_z \end{bmatrix}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} i_{xx} & i_{xy} & i_{xz} \\ i_{yx} & i_{yy} & i_{yz} \\ i_{zx} & i_{zy} & i_{zz} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - \begin{bmatrix} i_{xx}C_x + i_{xy}C_y + i_{xz}C_z \\ i_{yx}C_x + i_{yy}C_y + i_{yz}C_z \\ i_{zx}C_x + i_{zy}C_y + i_{zz}C_z \end{bmatrix}$$

The perspective equation then follows.

$$U = -foc\frac{X_c}{Z_c} = -foc\frac{i_{xx}X + i_{xy}Y + i_{xz}Z - (i_{xx}C_x + i_{xy}C_y + i_{xz}C_z)}{i_{zx}X + i_{zy}Y + i_{zz}Z - (i_{zx}C_x + i_{zy}C_y + i_{zz}C_z)} \quad (6.3)$$

$$V = -foc\frac{Y_c}{Z_c} = -foc\frac{i_{yx}C_x + i_{yy}C_y + i_{yz}Z - (i_{yx}C_x + i_{yy}C_y + i_{yz}C_z)}{i_{zx}C_x + i_{zy}C_y + i_{zz}Z - (i_{zx}C_x + i_{zy}C_y + i_{zz}C_z)} \quad (6.4)$$

The matrix can be written as two functions with combined coefficients as in Eqn. 6.6.

$$U = \frac{A \cdot X + B \cdot Y + C \cdot Z + D}{J \cdot X + K \cdot Y + L \cdot Z + 1} \quad (6.5)$$

$$V = \frac{E \cdot X + F \cdot Y + G \cdot Z + H}{J \cdot X + K \cdot Y + L \cdot Z + 1} \quad (6.6)$$

Another common form is the 3 x 4 matrix in Eqn. 6.7 where the image coordinates are expressed as ratios, $U = \frac{u}{w}$, $V = \frac{v}{w}$ of the *homogeneous* coordinates (u, v, w). *Projective Geometry*, homogeneous coordinate representations and the essential matrix are described in the texts of Faugeras [37] and Hartley [54].

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ J & K & L & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (6.7)$$

A common calibration approach is to solve for $A, B, ...H$ directly with known image point to 3D point correspondences, without any attempt to separate the parameters. This

finds directly the mapping from scene points and image points. In the reverse mapping, finding a 3D ray from an image point, the equations can be rewritten as two planes, with the intersection defining the 3D line.

6.3 Perspective Camera Calibration: Two-Plane Method

A very elegant and fundamental calibration method which bypasses all attempts to model any intrinsic or extrinsic parameters and go straight to the final result of mapping a 3D ray to an image plane pixel is the work of Wei and Ma [113]. Their *two plane* technique involves finding correspondences between two images of a calibration pattern taken at two depths. Each pixel can interpolate its position between identified markers on the calibration plane in both images and find a 3D start point and vector to define the ray corresponding to that pixel.

This method could be used for our panoramic sensors, two cages of measured points could be lowered around the sensor, carefully aligned and a set of two 3D points found or interpolated for every image point. This would allow a 3D ray to be determined from the line passing through both points.

For ease of implementation, the radial symmetry was assumed exact and the procedure applied to a linear calibration pattern presented to the sensor at two depths, thus producing two 2-D points of range and height for each radius. This one-dimensional adaptation of the two-plane method for the panoramic stereo sensor was implemented to improve the accuracy over a modeled approach of the catadioptric system, and is detailed in Chapter 7.

6.4 Calibration of Dioptric Component of Panoramic Sensor

The automatic calibration technique from Appendix B was applied to find the intrinsic parameters of the dioptric camera used in the stereo panoramic sensor. The Canon D30 was measured to have a focal length of 6050 pixels, which was verified with a manual approach. The radial distortion was found to be negligible, unlike all the low cost video cameras and digital cameras used previously in the author's research which suffered from extensive radial distortion. Radial distortion was not detectable with the Canon D30. Hence no image pre-warping was required to produce a proper perspective projection of the camera's view.

6.5 SVP Camera Calibration Summary

Calibration is a process whose goal is determining a 3D ray in space corresponding to each image pixel. Calibration can be achieved by dividing the process into determining internal, intrinsic parameters and external extrinsic parameters defining position and orientation. Alternatively this can be performed by simply noting the position of many points of a calibration pattern at two distances, and constructing a lookup table of rays according to image position. Proper calibration of any stereo system is important to obtain reasonably accuracy, and becomes crucial with large range to baseline ratios as our panoramic stereo system will have. This chapter demonstrated that even with perfect calibration, the image resolution and geometry still set minimum bounds for 3D reconstruction accuracy.

Chapter 7

Reconstruction Accuracy and System Calibration for Panoramic Stereo

Theoretical stereo reconstruction accuracy and calibration are two different topics that are closely linked and are discussed together in this chapter.

The stereo sensor design allows the benefits of panoramic vision with the need for only one image plane, however the confidence in reconstructed scene coordinates needs characterizing. The same limitations of object range versus baseline (the distance between viewpoints) apply as described for classical non-panoramic stereo, the general trend being that position accuracy degrades with a large distance to baseline ratio.

As well as characterizing what errors can occur with an ideal model, calibration needs to be performed and evaluated since inaccuracies therein will add more error. Three-dimensional polyhedral reconstruction using our catadioptric panoramic stereo sensor requires calibration as conventional dioptric stereo rigs. This is performed, both to find parameters for feature extraction, and to recover 3D points given a pixel location in each view.

The chapter deals with some of the practical concerns when attempting to reconstruct 3D models with the novel catadioptric panoramic stereo sensor. The system used for this thesis utilized spherical mirrors, mirrors with a circular radial profile. However, when in the laboratory, it was determined that this ideal model could not be used for reconstruction because of mirror imperfections. The experiments and results that led to this are detailed.

7.1 Canon D30 Digital Camera as Dioptric Component

The dioptric component of the system, the Canon EOS D30 digital camera (Fig. 7.1) and 50mm lens assembly, gave a high quality image of 2160 x 1440 square pixels with negligible radial distortion. The camera was chosen because of its ability to mount different standard lenses. It is a digital camera in a SLR camera body. This camera is more expensive than several competing brands which have built-in, unchangeable lenses. However, it was found that the depth of field with any of these less expensive cameras was not sufficient to focus on both mirror lobes simultaneously. Hence it was necessary to obtain a camera to which standard professional camera lenses could be attached. The Canon D30 was chosen for this reason and for its ability to capture images remotely from a computer. The camera was interfaced with via a USB interface to a PC running the image aquisition software on the Windows 2000 platform.

The camera uses a CMOS image sensor of size 2,160 x 1,440 pixels. The sensor itself is quite large, 22.0 x 14.9 mm giving a pixel width of approximately 10.1 microns. The large size of the CMOS sensor imaging area allows more light to be captured, increasing image quality. For comparison, lower end consumer video cameras have a CCD sensing area

of about 5.5 mm, and standard 35 mm film (by definition) is 35.0 mm x 23.3 mm. The Sony 1/1.8 CCD (used in the Nikon Coolpix 990, Sony DSC-S70, Olympus C-3030Z) has comparable resolution but only an imaging area of 5.52 x 4.14 pixels [7]. This gives the Canon D30 a sensor area per pixel 15 times greater, resulting in 15 times as much light being captured per pixel. Image sensor area is one considerations of importance in our system because of the small aperture setting. Wetzel and Frosini [114] explain how the shot noise at low levels is an inverse function of imaging element size. The electrical noise level is roughly constant, whereas the captured signal is proportional to the pixel area. For a given irradiance, the signal-to-noise ratio (SNR) will be higher (rises linearly with pixel size) and hence create a cleaner image.

The aperture was set to 9.5 to get suitable depth of field to allow clear focus on both mirror lobes simultaneously. This results in a shutter time of 1.2 seconds, which is quite long but due to the Canon D30's large pixel size results in a low noise image. After focusing, the focal length was measured to be 6050 pixels as described in Chapter 6. With a 50 mm (focal length) lens, the Canon D30's "focal length multiplier" of 1.6 and a pixel size of 10.1 microns, a focal length of 7920 pixels was expected, reinforcing that manufacturers' specifications are not always accurate.

The dioptric component of the system, Canon D30 digital camera and 50mm lens assembly, gave a high quality image of 2160 x 1440 square pixels with negligible radial distortion. The aperture was set to 9.5 to get suitable depth of field to allow clear focus on both mirror lobes simultaneously. Overall, after choosing the correct lens, adjusting the focus and aperture, the Canon D30 gave good quality images, the only parameter necessary for the feature extraction and calibration was the focal length.

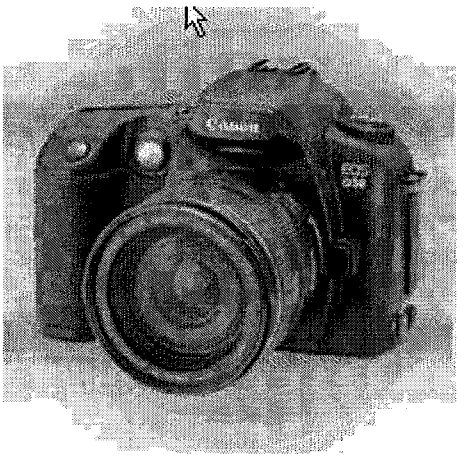


Figure 7.1: *Digital camera used for dioptric component of stereo panoramic sensor.*

7.2 Calibration Assuming Perfect Mirror Geometry

The camera and mirror were rigidly mounted in a solid frame that occluded only 2.5 azimuth degrees. The camera center was determined, and the inner and outer radii of both annular regions of useful image were found. This was determined as the area of the mirror surface that corresponded to a view common to both lobes. The inner annulus, the reflection of the inner mirror lobe, had a width of 126 pixels (starting at a radius of 83) whereas the outer annulus was chosen to be 302 pixels wide (starting at a radius of 262). Overall 28.9 percent of the image pixels were usable. The loss was due mostly to the aspect ratio of the digital camera image size being rectangular whereas only a circular region was useful. Thus, much of the image on either side of the mirror was unusable.

Assuming spherical geometry (Fig. 7.2), aligned camera and mirror lobe axis and no radial distortion, Eqns. [7.1-7.6] model the mapping between image and scene points. Considering a single lobe, there are three geometric parameters to determine: the dioptric camera focal length F, the distances between the camera focal point and each of the mirror lobe centers F_y, and the mirror radii R. After the camera's focal length was determined, the other two geometry parameters were still unknowns. Eqns. [7.1-7.6] and Fig. 7.2 show the relationship between a scene point (P_x, P_y) and an image point with a radius of U pixels.

$$M_x = U\frac{F(F_y+R) - \sqrt{F^2R^2 - U^2F_y^2 - 2F_yU^2R}}{U^2+F^2} \tag{7.1}$$

$$M_y = \sqrt{R^2 - M_x^2} \tag{7.2}$$

$$\alpha = tan^{-1}(\frac{M_x}{F_y - M_y}) \tag{7.3}$$

$$\beta = tan^{-}1(\frac{M_x}{M_y}) \tag{7.4}$$

$$\psi = 2\beta + \alpha \tag{7.5}$$

$$P_x = M_x + P_y tan(\frac{\pi}{2} - \psi) \tag{7.6}$$

If sample image points were recorded, for 3D points of known height, then Eqns. [7.1-7.6] would have only two unknowns: F_y and R. A parameter space approach was utilized to find a relationship between each mirror lobe distance F_y and radius R. This was achieved by measuring the radius of scene points at known locations, and numerically calculating a curve of F_y versus R. 22 points were used and their numerically determined relationships superimposed for the outer lobe, and 12 points were used for the inner. The curves produced from several points at different elevations were overlaid, and the resulting peak line defined a relation between F_y and R. Since F_y can be measured manually with a lower percentage error, it is taken to be correct, and R determined from this parameter space relation. Fig. 7.3 (left) shows the parameter space image for the inner lobe, and Fig. 7.3 (right) shows

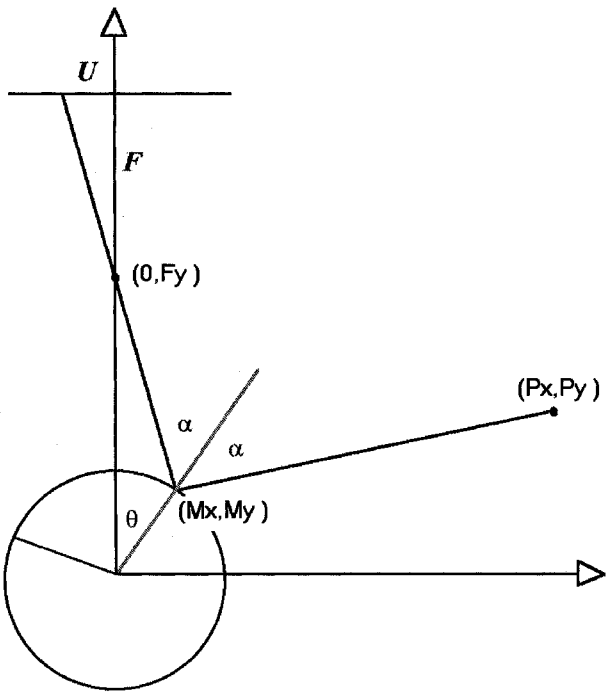


Figure 7.2: *With radial symmetry of the mirror, a two-dimensional slice through $P(x, y, z)$ models the projection of the same point (expressed in coordinates (P_x, P_y) in the axis of this vertical plane) after being reflected from a spherical mirror to a position at a radius of U on the image plane.*

the outer lobe. The images are not drawn with high accuracy axis scales, they are provided for clarity, during calibration the actual relationship was read off by counting pixels. Using a manually measured F_y, R for each lobe could be read from the graph to yield a value of much higher precision than could be measured directly. The percentage error of the F_y measurement would translate (after considering the graph slope) to a similar percentage error in R. Since F_y could be measured to millimetre accuracy over some 50 cm, the mirror radius could be estimated with an accuracy of 20 microns for the inner lobe, and 100 microns for the outer. The inner lobe was measured to have a $F_y = 48.7$ cm and $R = 1.950$ cm, and for the outer lobe $F_y = 53.9$ and $R = 5.900$ cm.

Four image space parameters also need to be found for each lobe, the image center (U, V) and the minimum and maximum radii for stereo vision. The image center for each lobe was determined simply by the intersection of the projection of vertical edges in the scene. This was done separately for the two lobes since the central mirror axis would have some error relative to the Canon D30 focal point. Objects were moved around in the scene to obtain the minimum and maximum overlapping volume to permit stereo.

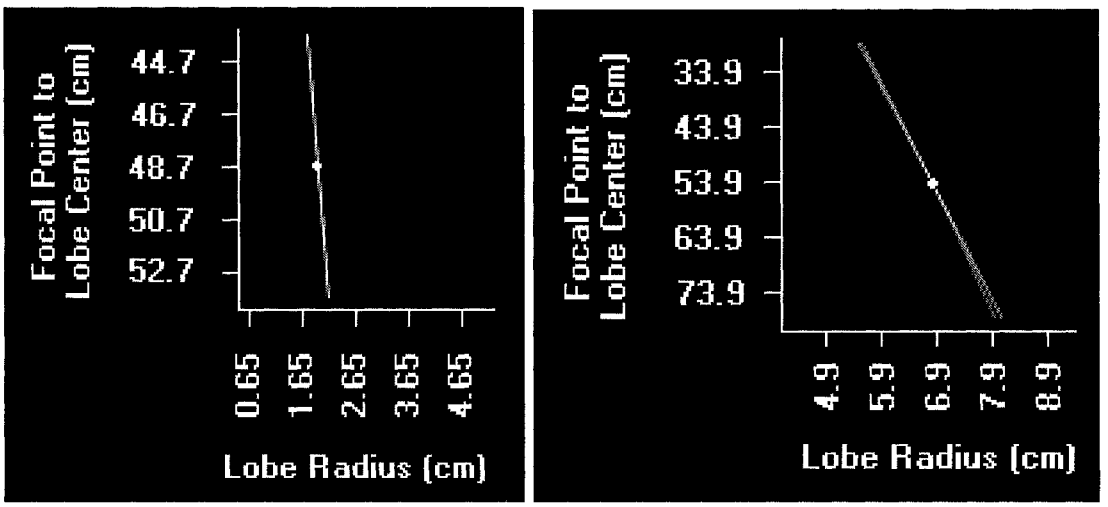


Figure 7.3: *Finding the camera height to mirror radius relation (F_y to R) via numerical means. Left image shows the relationship for the inner lobe, the right image shows the outer.*

Thus, assuming the mirrors to be perfect spherical lobes, the calibration information of the entire system consisted of the focal length (F), mirror position (F_y), mirror radius (R), R_{inner}, R_{outer} and (C_U, C_V) for each lobe. These parameters were stored in the .GEO file format in the software implementations of the systems described in this thesis (Appendix A). This proved sufficient for use with the Panoramic Hough Transform to correctly extract horizontal and vertical line segment projections.

7.3 Effective Baseline

With a non-SVP mirror, there is no common virtual convergence point of rays incident on the mirror, and hence the effective baseline between the two viewpoints of a scene point varies with the elevation in each lobe. Fig. 7.4 demonstrates this, and shows how if the panoramic sensor is placed in the orientation shown with the camera above looking down, objects with a higher elevation angle have a larger baseline, and hence better accuracy of position can be achieved.

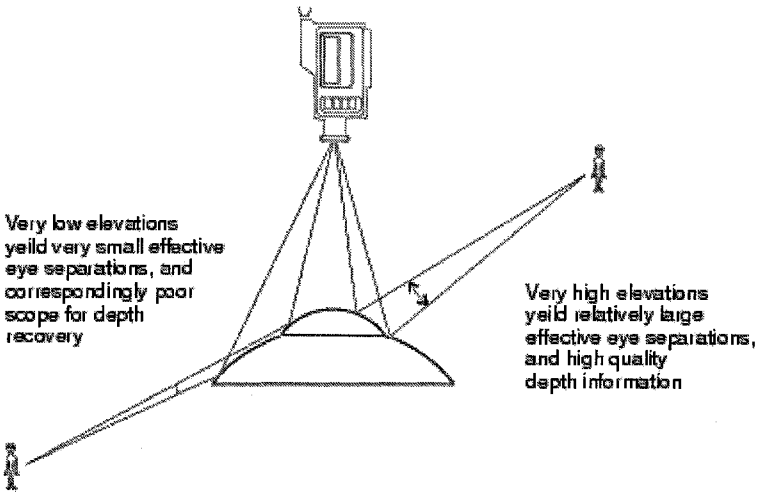


Figure 7.4: *Different elevations provide different effective baseline distances.*

Eqns. [7.1-7.6] above allow the numerical determination of the mirror reflection point (M_x, M_y) in the cross-section geometry once the F_y and R parameters were known. The calibration pattern points at a distance of 100 cm were used. Although the average baseline changes with elevation angle, the average baseline is about 2.75 cm as shown in Fig. 7.4.

7.4 Geometric Constraints on Range Accuracy

As shown in Chapter 6, the accuracy of stereo reconstruction can be well approximated as a function of the range to baseline ratio. Better accuracy could be obtained by using a larger bi-lobed mirror, or one with a larger inter-lobe distance. This then requires a greater depth of field for the dioptric camera to be able to focus on both lobes, and hence a larger lens and/or smaller aperture setting and longer exposure. Even with a dioptric camera with a large depth of field, practical constraints limit the inter-lobe distance. The effective baseline distance is unfortunately inherently small compared to the scale of the entire sensor for a practical implementation.

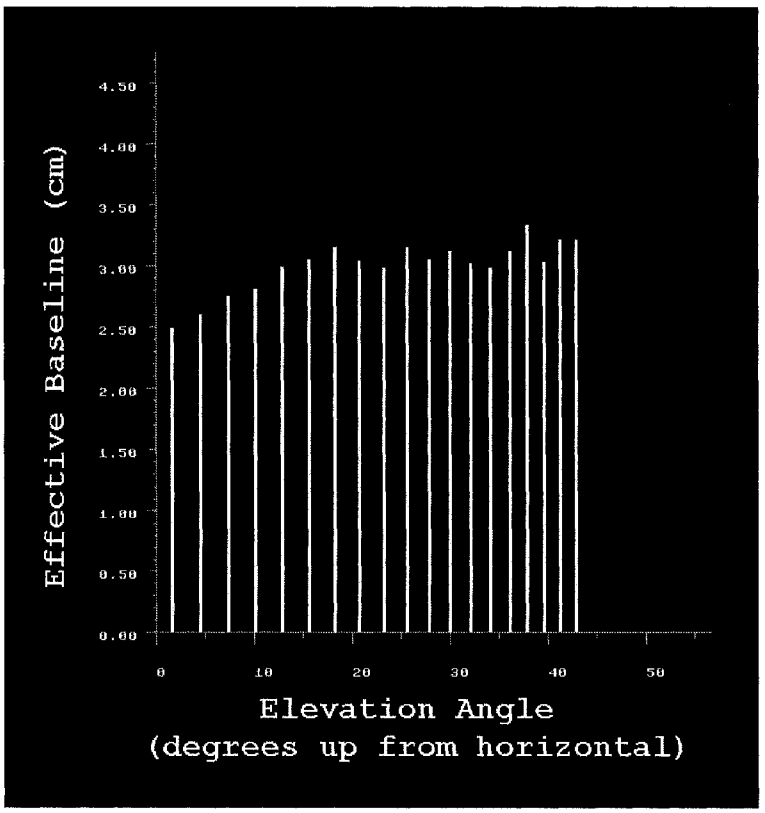


Figure 7.5: *Plot of effective baseline with respect to elevation angle (at a large distance).*

The effect of a small baseline with respect to the distance to the scene point (range, depth) is that of a long, thin region of uncertainty as shown in Fig. 7.6.

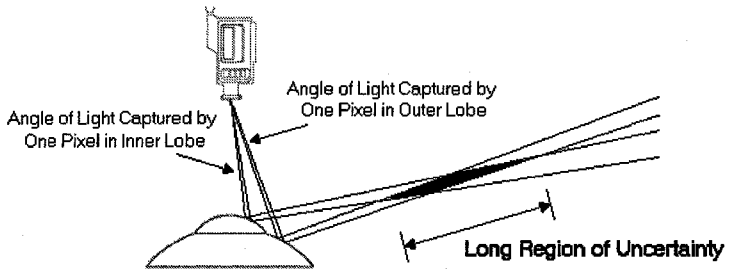


Figure 7.6: *Region of uncertainty associated with a narrow baseline.*

7.5 Experimental Calibration Verification

A series of images were taken with test markers at different measured 3D locations, and their locations in the images manually located. The test markers consisted of fiducials (Fig. 7.7) and a vertical test pattern with markings at 1 cm and 5 cm intervals. The vertical test pattern was imaged at distances from a range of 15 cm to 90 cm, in 5 cm steps (Fig. 7.8).

A set of some 250 data groups were aggregated, each data group consisted of a (U, V) image point and the corresponding 3D coordinates (X, Y, Z). The accuracy could then be tested for the two types of calibration. The 3D ray produced by the calibration data for a given image point was examined for where it reached the Z height of the 3D point, where a X_{calc} and Y_{calc} value could be calculated. The percentage distance error therefore is $100\% \cdot (\frac{\sqrt{X_{calc}^2+Y_{calc}^2}-\sqrt{X^2+Y^2}}{\sqrt{X^2+Y^2}})$.

The percentage distance error was calculated using the spherical model of Eqns. [7.1-7.6], and the empirical lookup table based calibration described in section 7.7. The results are shown in Fig. 7.9 as a function of range and it can be seen clearly that the empirical calibration provides better accuracy. Fig. 7.9 (top) details the best accuracy the spherical model could provide. The data was tested with many slightly different geometry parameters to verify the ones derived above and no better accuracy results could be obtained with the empirical data set. The conclusion is that the mirror is not perfectly spherical. Also local mirror imperfections were present, areas where the radial profile was inconsistent.

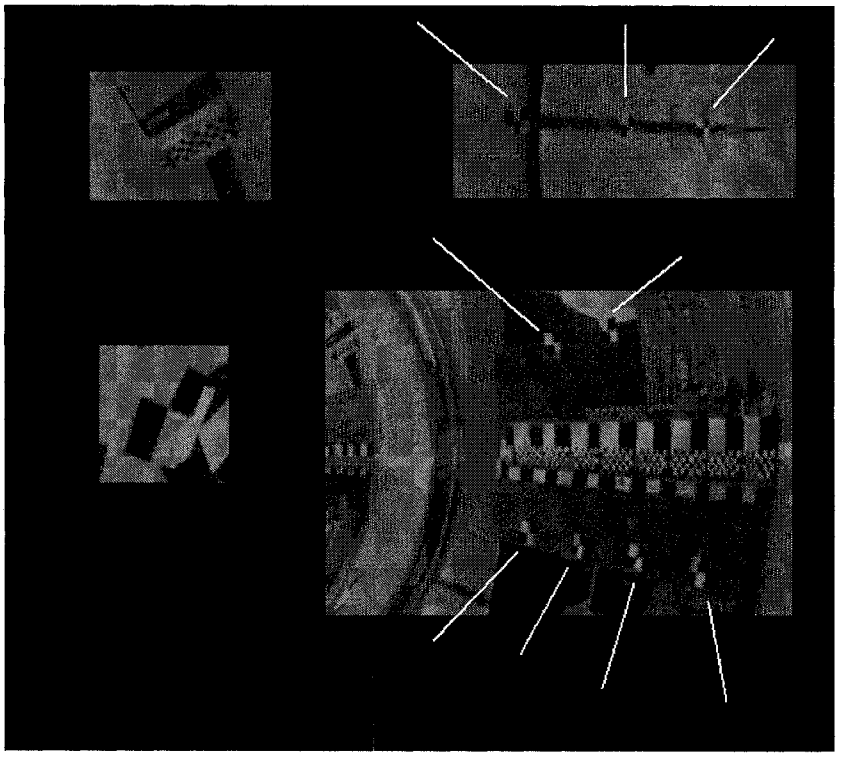


Figure 7.7: *Calibration accuracy test: Examples of image sections showing fiducials (calibration targets). The targets were mounted in the scene at different depths. The 3D location, and the manually located image coordinates were recorded.*

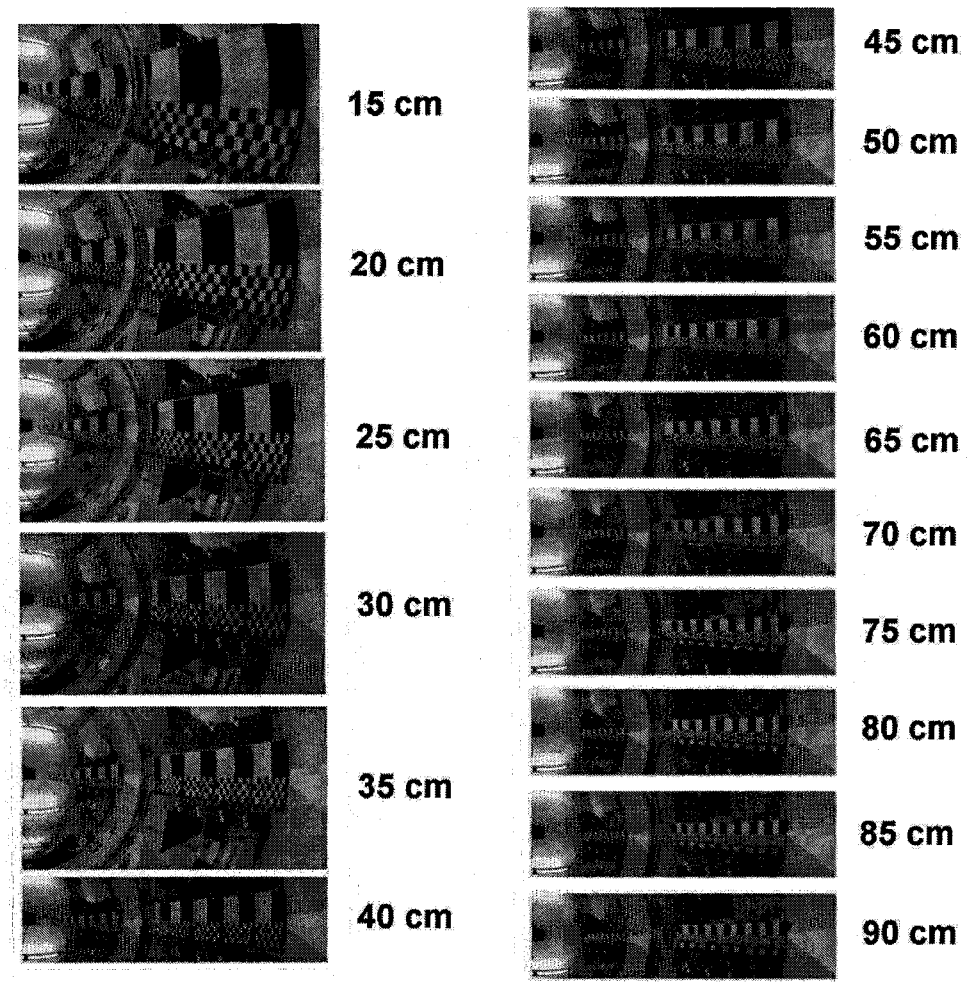


Figure 7.8: *Calibration accuracy test: Image sections used to evaluate accuracy of calibration methods. A vertical test pattern was placed at various ranges and the image coordinates recorded for all visible height markings. Note that the test pattern can be seen in both lobes. The resolution of the inner lobe was not sharp enough to see the 1 cm gradations, but the 5 cm gradations were visible.*

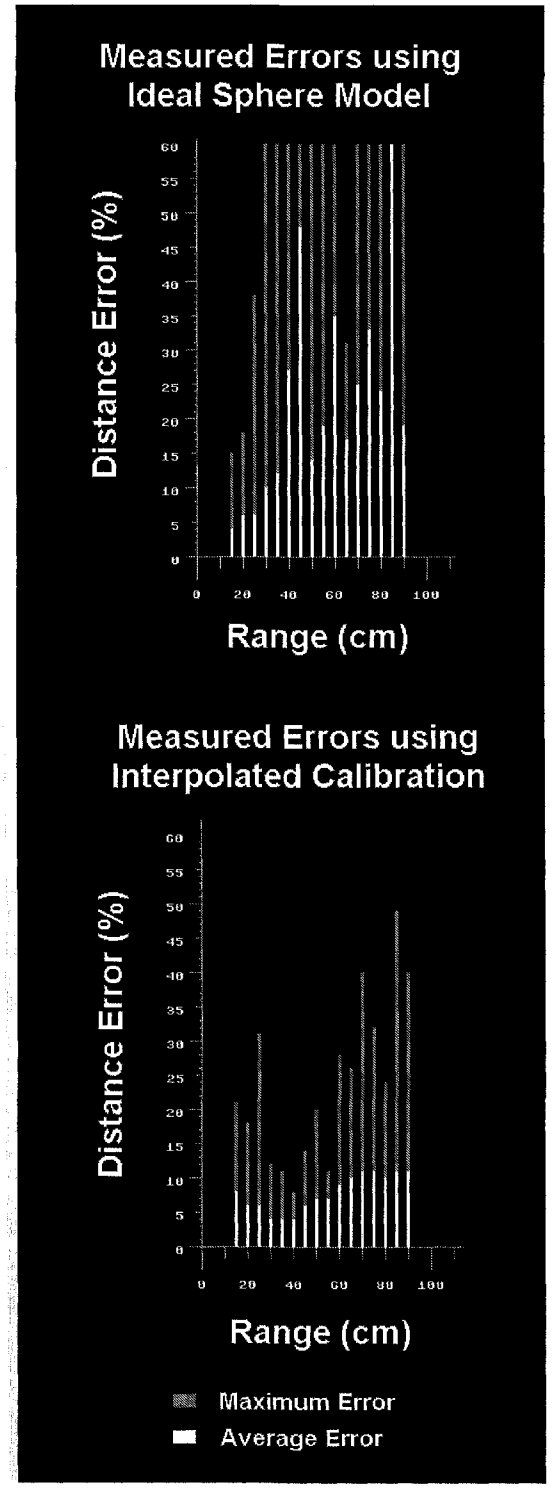


Figure 7.9: *Calibration accuracy test: Examples of image sections showing fiducials (calibration targets). The targets were mounted in the scene at different depths. The 3D location, and the manually located image coordinates were recorded.*

7.6 Mirror Imperfections

Since only finite precision is possible in the real world, imperfections are expected and indeed found in the mirrors used. Both the single and double lobed mirrors, small aberrations magnify to produce a visible error due to the spreading effect of the convex curvature. They can be seen in Fig. 7.10. These aberrations were gentle ripples in the otherwise continuous profile, and could be seen by visually inspecting the mirror at close range. These were radially symmetric as would be expected with a mirror produced on a lathe. The lookup table calibration approach could attempt to address this phenomenon due to its radial symmetry.

7.7 Alternate Calibration: Lookup Table

The ideal spherical geometry model had to be discarded to improve the calibration accuracy. For this reason, only radial symmetry was assumed, and a one-dimensional version of Wei and Ma's two-plane calibration method [113] was implemented. The radius seen in the image was recorded for points at many heights for both ranges (20 cm and 100 cm). The two images captured to generate these measurements are shown below (Figs. 7.11,7.12). The radius values in between measurements were interpolated producing a radius to height lookup table for both range values.

The 3D ray for an image point is determined by converting (U, V) to polar coordinates (R, θ), finding the start and end heights from these lookup tables, and using using θ to convert to a final 3D ray. The improved accuracy was proved on the data set, with the results compared to that of the spherical model in Fig. 7.9.

7.8 Lookup Table Calibration Accuracy Evaluation

The percentage errors obtained from the data set are shown qualitatively in Fig. 7.13 and statistics aggregated in Fig. 7.14. It can be seen that the error increases at both the maximum and minimum heights. This was due to the mirror having more aberrations near the transition between the two lobes, and the effect of rays approaching the parallel horizon lines.

Due to the near parallel rays from each lobe, the error in 3D point reconstruction can be characterized by a large error in distance and a small error in azimuth and elevation angles. This is shown in Fig. 7.14, note the small error in elevation angle compared to the

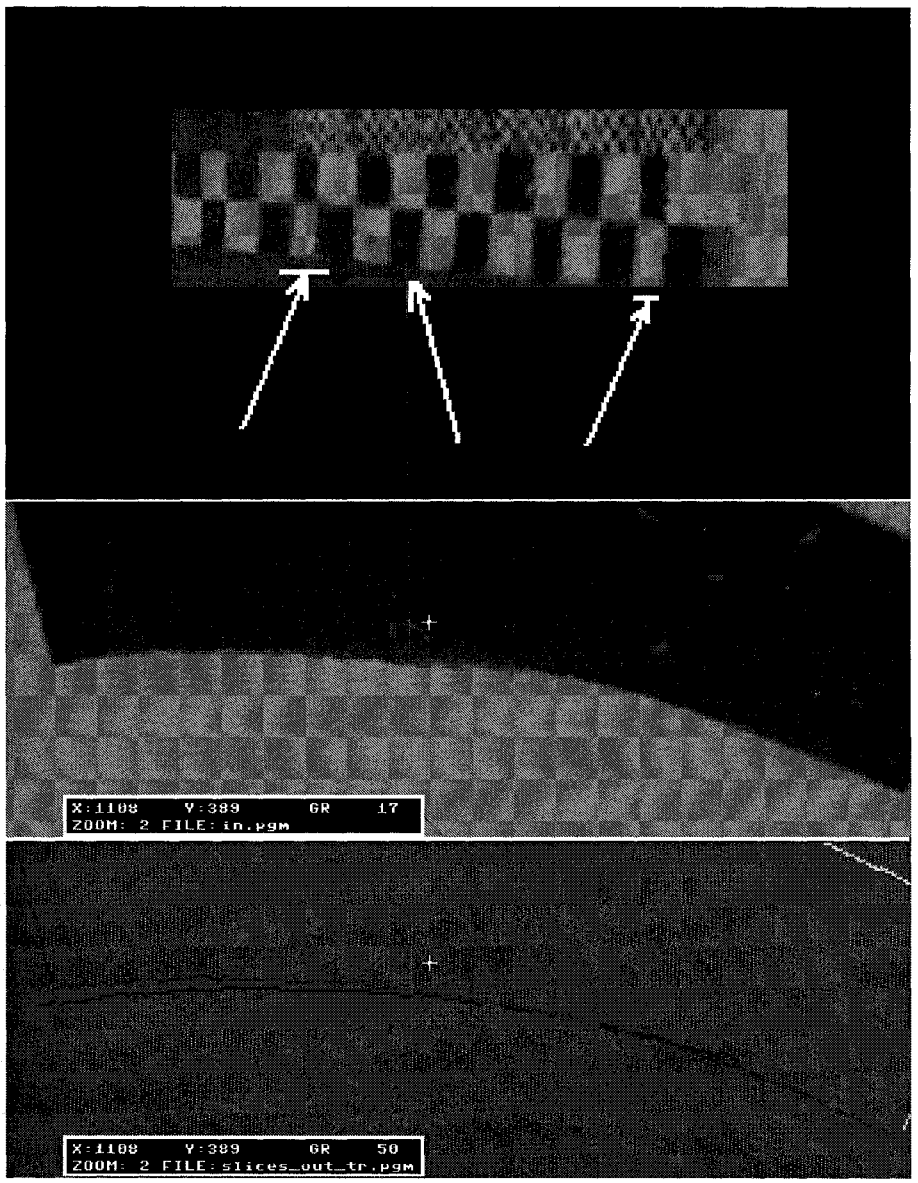


Figure 7.10: *Mirror imperfections. The top image shows inconsistencies in the reflection of the vertical test pattern, note the differences in marker widths. The bottom image is from the reconstruction experiments in Chapter 8 and shows an error that created a bulge in the polygon edge. The feature extraction was adversely affected and the polygon was not recognized.*

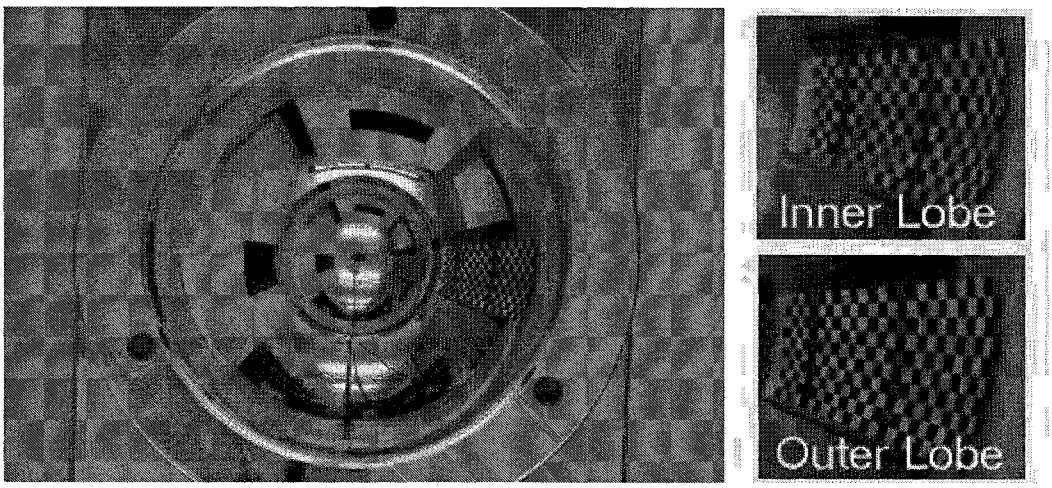


Figure 7.11: *Lookup table based empirical calibration. Image captured with vertical test pattern at range = 20 cm.*

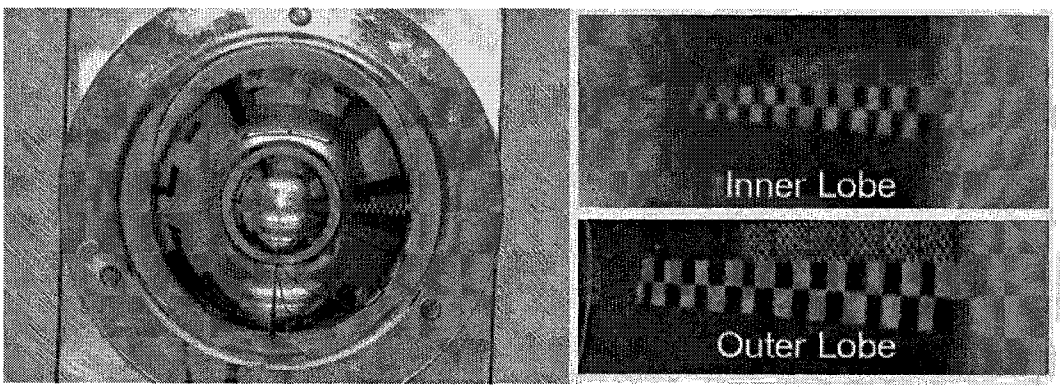


Figure 7.12: *Lookup table based empirical calibration. Image captured with vertical test pattern at range = 100 cm.*

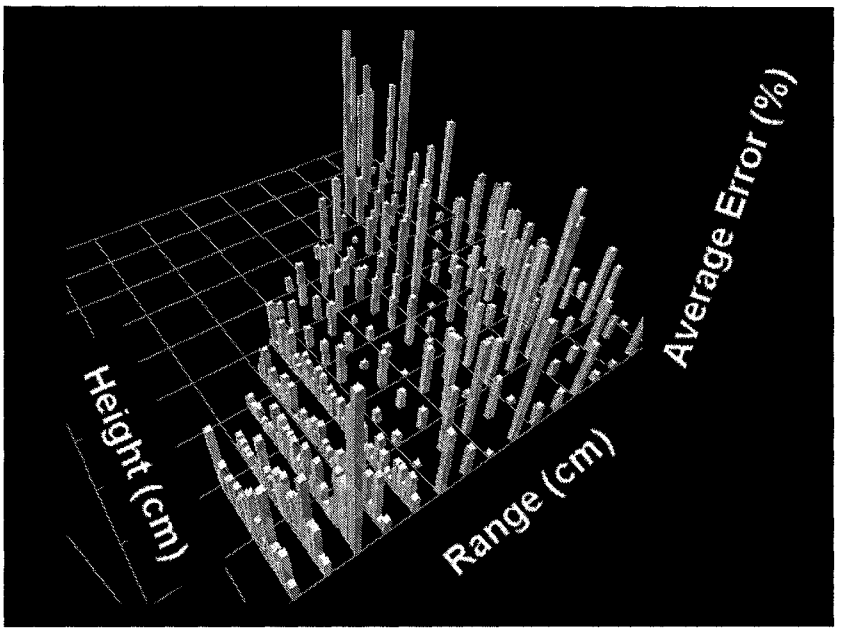


Figure 7.13: *Experimental calibration results: Distribution of distance error as a function of range and elevation (percentage distance error expressed as height).*

large error in distance. The azimuth error is not shown, but was less than 0.5 degrees. This correlates with the long, thin volume of uncertainty depicted in Fig. 7.5 above. The volume of space that a scene point can lie in (and have this point project to the same two points in the image plane) is small in the elevation and azimuth angular direction, but large in the distance out from the mirror.

7.9 Calibration Error According to First Order Error Model

The first order approximation of expected error developed in the previous chapter for classic stereo is applied to the panoramic stereo camera. Eqn. 7.7 and Fig. 7.15 is repeated from Chapter 6 showing how the error approximation is calculated. The uncertainty in distance according to this simplified diagram is estimated by solving for the error created by a one pixel error in stereo matching.

$$r = \frac{range}{baseline}$$

$$d_{error} = 100\% \cdot \left(\frac{tan(tan^{-1}(r) + d_\theta)}{r} - 1\right) \tag{7.7}$$

As with the classic stereo rig using two dioptric cameras, the image resolution and

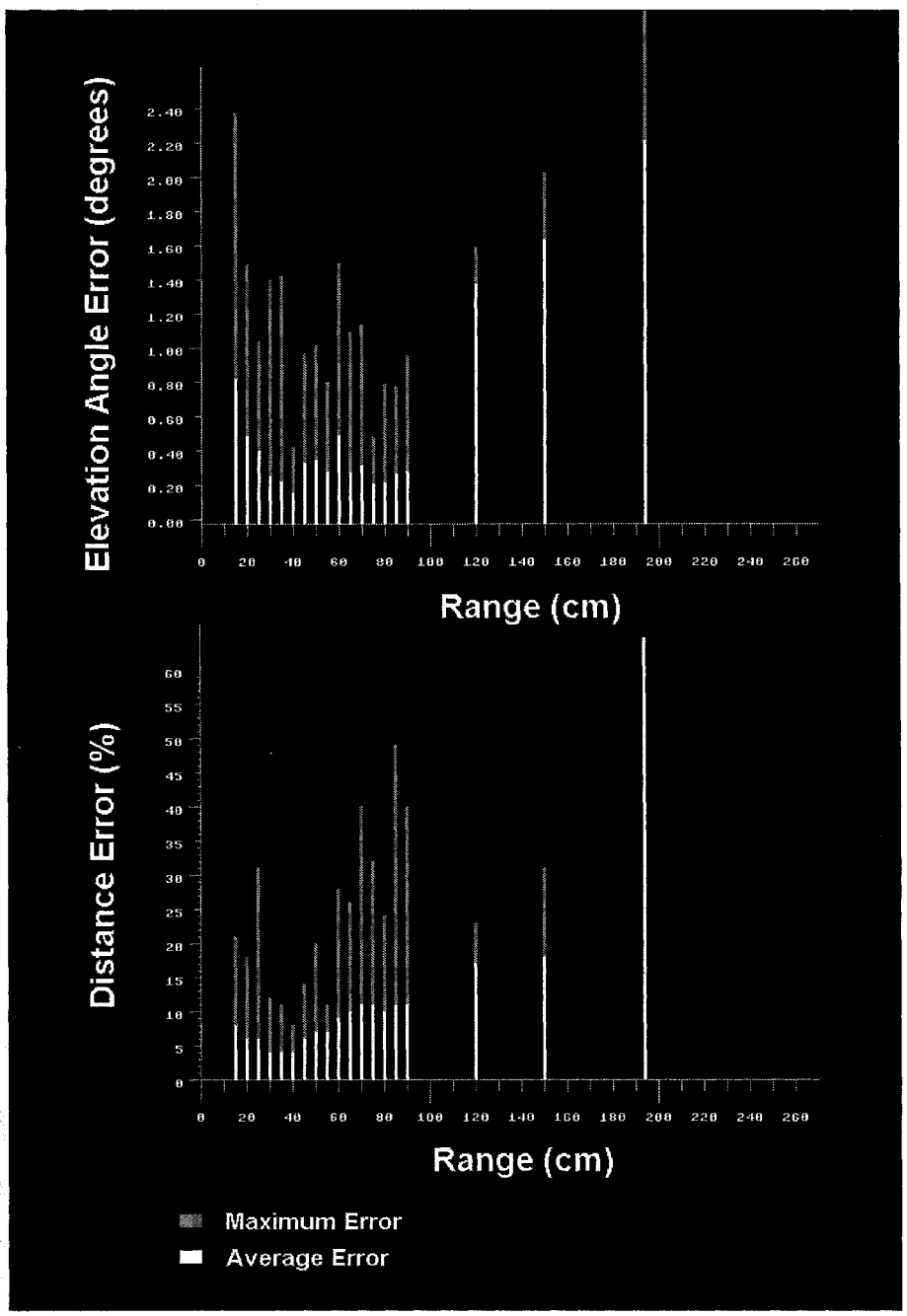


Figure 7.14: *Aggregated statistics of 208 3-D position reconstructions (points identified and matched manually). Note the small angular error, typically less than a percent, and the large distance error.*

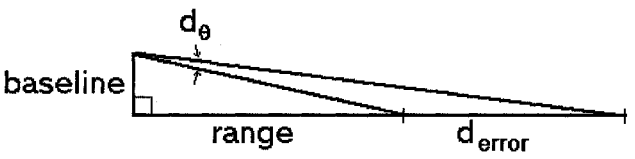


Figure 7.15: *Simplified distance error in stereo.*

baseline provide this error estimate. In our sensor, the inner lobe's annulus was 126 pixels wide, and the outer lobe's annulus was 302 pixels wide. In our sensor, the resolution of the inner lobe is the limiting factor for resolution. A crude approximation of 126 pixels divided by an elevation range of 60° yields about $\frac{1}{2}$ degree per pixel. The average of the effective baseline from Fig. 7.4 was plotted using Eqn. 6.2 with $d\theta = 0.5°$ for distances of 15 to 192 cm (Fig 7.16). This graph represents the expected percent error if a feature point is correctly identified in the outer lobe, but incorrect by only one pixel in the inner lobe. The average and maximum distance errors obtained experimentally are superimposed.

For the majority of the readings taken at the range less than one metre, the maximum errors encountered do correspond roughly to this simplified model. The model predicts the *percentage uncertainty*, i.e. the length of the volume contained by one pixel. Therefore it is expected that the average error should be about half this amount, with some measurements being in error by the full amount.

7.10 Calibration Error According to Second Order Error Model

The predicted error can be examined in more depth by considering the actual pixel to 3D ray mapping. This is done to obtain predicted distance errors using both the spherical model and lookup table based calibration. A spherical model pixel to 3D location mapping can be formed by solving for the intersection of the rays produced by applying Eqns. [7.1-7.6] for each lobe. Unfortunately Eqns. [7.1-7.6] cannot be combined into a single differentiable function to directly express this error/uncertainty volume. However the pixel to 3D ray mapping can be performed numerically and is done so for the proportions of our sensor.

The potential confidence/error volume length is calculated by finding the scene distance for integer values of image point radii in each lobe, and then comparing it to the distance obtained if the inner or outer radius is incremented. Four curves are produced, two that show the percentage distance difference between 3D points found by decrementing the inner lobe radius by one or two pixels, and two curves solving for the same by incrementing the

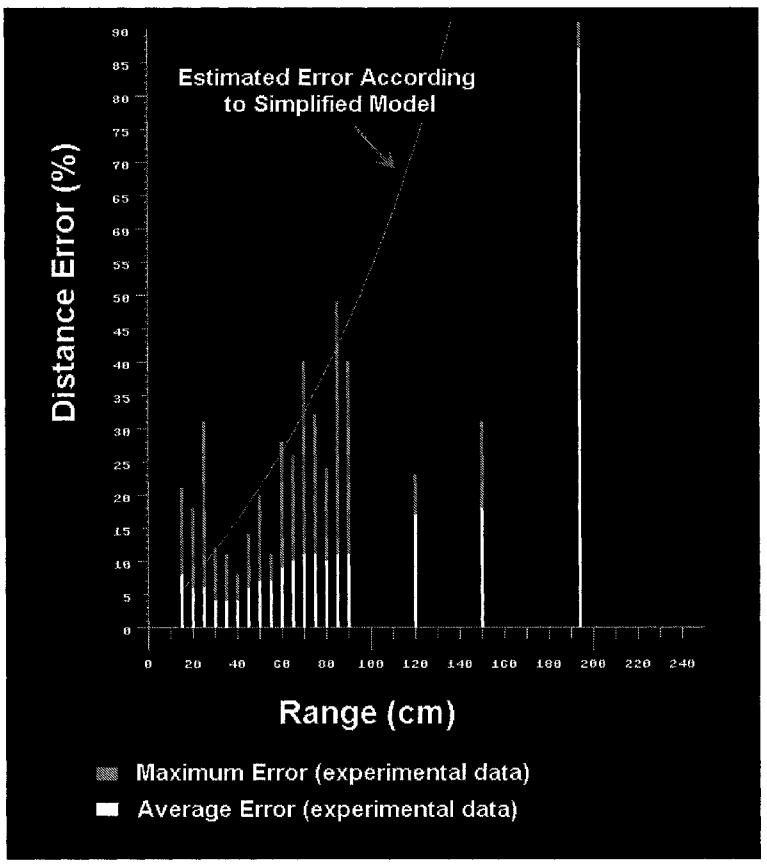


Figure 7.16: *Plot of estimated error caused by a one pixel error using the simplified model of Eqn. 6.2 and Fig. 7.16 (with $d\theta = 0.5°$ and baseline $= 2.9cm$). Experimental results of distance error are also shown (vertical bars), with both the average (white) and maximum (grey) percent error shown to compare to the simplified model.*

outer lobe radius by one and two pixels. The inner lobe radius is decremented, and the outer lobe incremented to create positive $d\theta$ angles (Eqn. 7.7) which correspond to the larger error. This is calculated for the full range of elevation angles above the horizon, and the maximum error reported. These results of the spherical model error estimation are shown in Fig. 7.17 with the average empirical error (found with the lookup technique) overlaid for comparison.

The empirical data is from the above mentioned data set, where the clearly distinct fiducials and vertical pattern markers were identified manually, so that an average error of a half pixel is expected. Since the image of the inner lobe has less resolution, it is expected to be the limiting factor. The average error can be estimated to be one half of the length of the volume of uncertainty. Examining Fig. 7.17 it appears that the average empirical error seems to roughly fit this prediction. The average error curve is about half the maximum error caused by a one pixel error in the inner lobe. Also from Fig. 7.17 it can be seen that the average error of the measurements taken at a range of 195 cm is not explained by the spherical model.

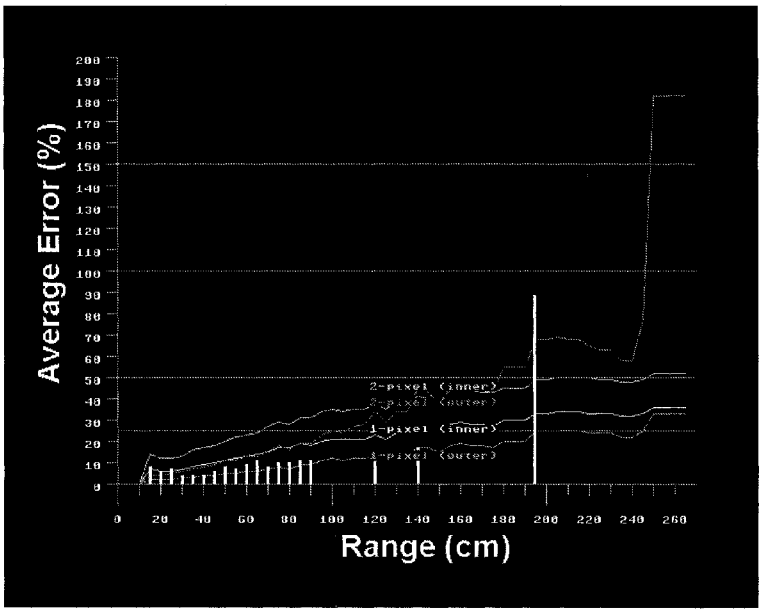


Figure 7.17: *Spherical model predicted percentage distance error. The distance error produced by being inaccurate in image radius by one or two pixels, for both lobes is shown. The average error found empirically is overlaid with vertical bars.*

The predicted error estimation is repeated for the pixel to ray mapping using the lookup table method.

Similar to the error calculations with a perfect spherical mirror, lengths of several volumes of uncertainty are calculated by finding the difference in range estimated from a given matched pair of image radii, and that found for a change in either lobe by one or two pixels.

This is calculated for the full range of elevation angles above the horizon and the maximum error reported. The full results are shown later in Figs. 7.19-7.22, but summarized in Fig. 7.18. It can be seen at first glance that this model predicts a much greater error, especially at larger ranges. This time the measurements at 195 cm are accounted for.

The reason for the drastically more pessimistic expected error is due to the effects of larger errors appearing at the lower heights, closer to the horizon lines for both mirror lobes. Since the curves in Fig. 7.18 show only the maximum error, the worst error in the range of elevation angles overrides the rest.

Another observation is the non-monotonic, almost chaotic nature of the predicted error curves. This is explained by the quantization error in digital images. As the distance increases within the range of one pixel, the radius reading is still the same so the percentage error rises. As we reach the next integer measure of image radius, the error drops as the predicted and empirical errors become closer again. The width of these "oscillations" corresponds to being in the range of one pixel.

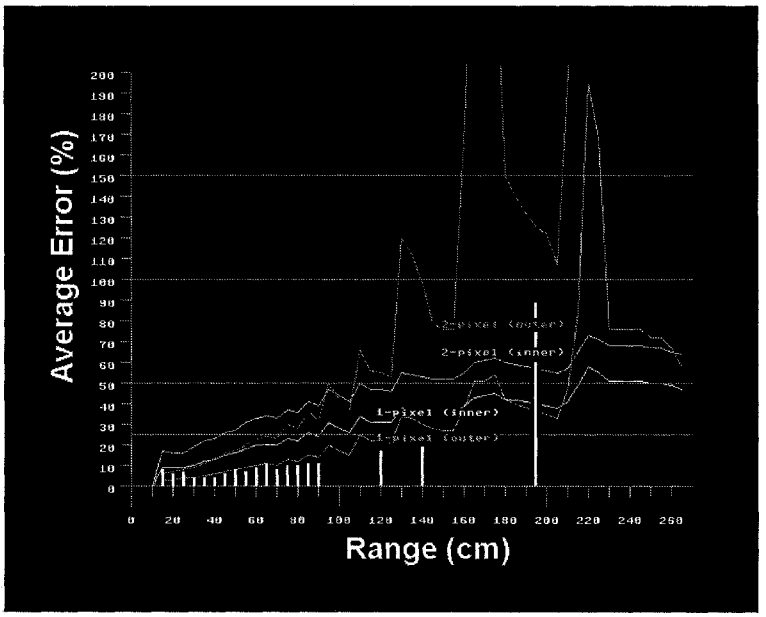


Figure 7.18: *Predicted percentage distance error using the lookup table calibration model. The distance error produced by being inaccurate in image radius by one or two pixels, for both lobes is shown. The average error found empirically is overlaid with vertical bars.*

As mentioned above, the predicted error above is the maximum error for a given range. The results that combined into Figs. 7.17-7.18 are provided below in Figs. 7.19-7.22. It can be seen why the error at the horizon was not included since it would dwarf the others. This large error occurs when both the image radii are close to the horizon line. If a one or two pixel error causes them both to fall at the horizon, or to cross over and create diverging rays then the scene distance cannot be calculated. Asymptotes in the error curves form as

the range of objects on the horizon get too large.

7.11 Calibration and Error Analysis Conclusions

The real world task of calibrating imperfect system components has to be addressed to make a working system. The parameters for the catadioptric system were determined assuming mirrors of a perfect circular profile, which were used for the feature extraction.

The error in reconstructing 3D points given a point pair $(U_{inner}, V_{inner}), (U_{outer}, V_{outer})$ was examined both from a simplified theoretical viewpoint, and with more comprehensive models. Catadioptric panoramic image sensors with practical physical dimensions suffer from a large range to baseline ratio and hence errors in distance of 50% or more were expected and observed. The errors were shown to be primarily in distance as expected, and were a function of both the range and height of the 3D scene point.

The spherical mirror model could not provide the necessary accuracy because of mirror imperfections. A one-dimensional lookup table approach similar to Wei and Ma's two plane method was employed and the error was reduced.

The practical results of the work detailed in this chapter are the ideal spherical mirror geometry parameters for use with feature extraction, and a calibration procedure for reconstructing 3D points with an estimate of their accuracy.

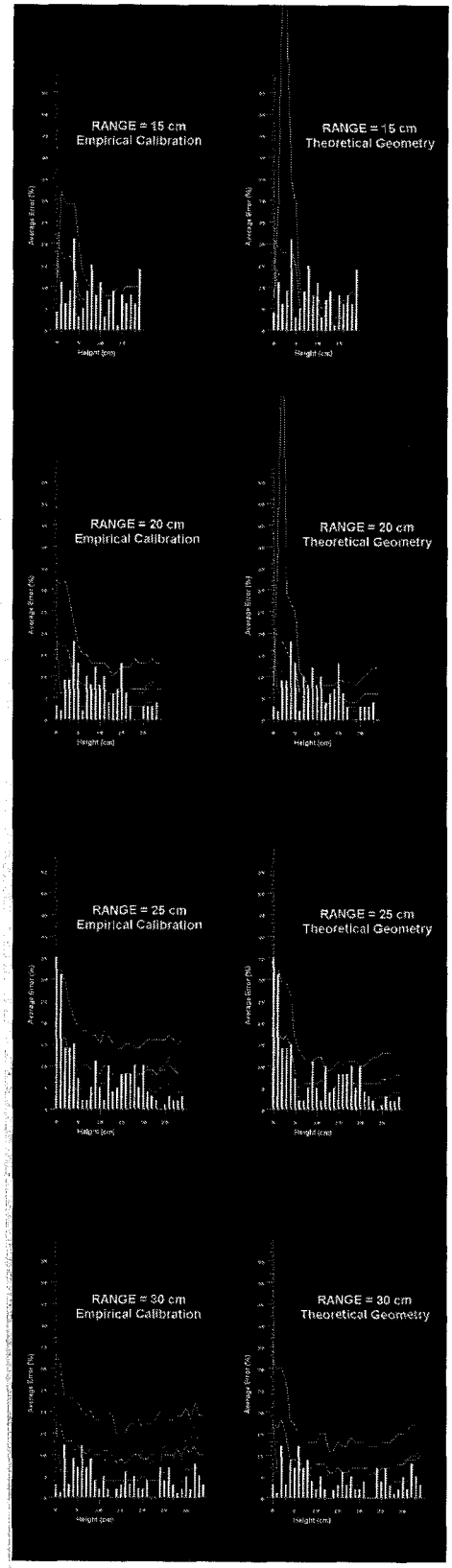


Figure 7.19: *Distance error at a range of 15 to 30 cm. Plot of expected distance error as a function of range and height for errors of one and two pixels in the inner and outer image radii.*

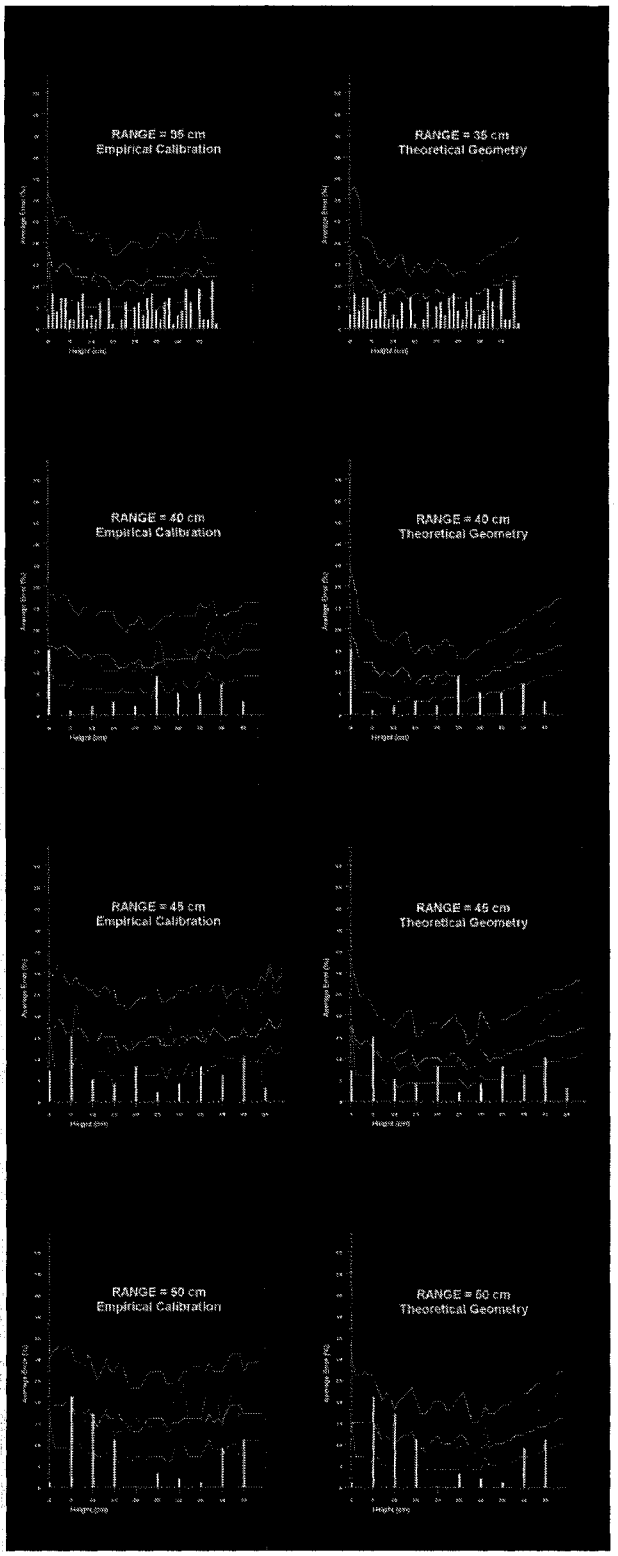


Figure 7.20: *Distance error at a range of 35 to 50 cm. Plot of expected distance error as a function of range and height for errors of one and two pixels in the inner and outer image radii.*

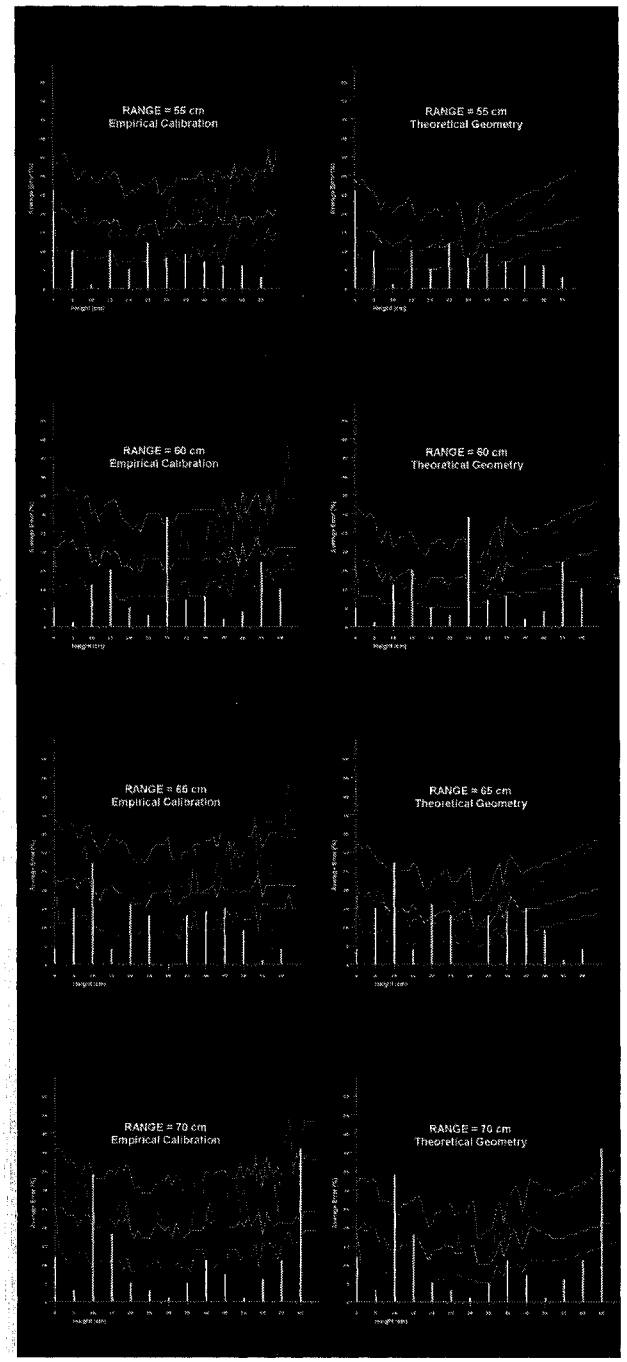


Figure 7.21: *Distance error at a range of 55 to 70 cm. Plot of expected distance error as a function of range and height for errors of one and two pixels in the inner and outer image radii.*

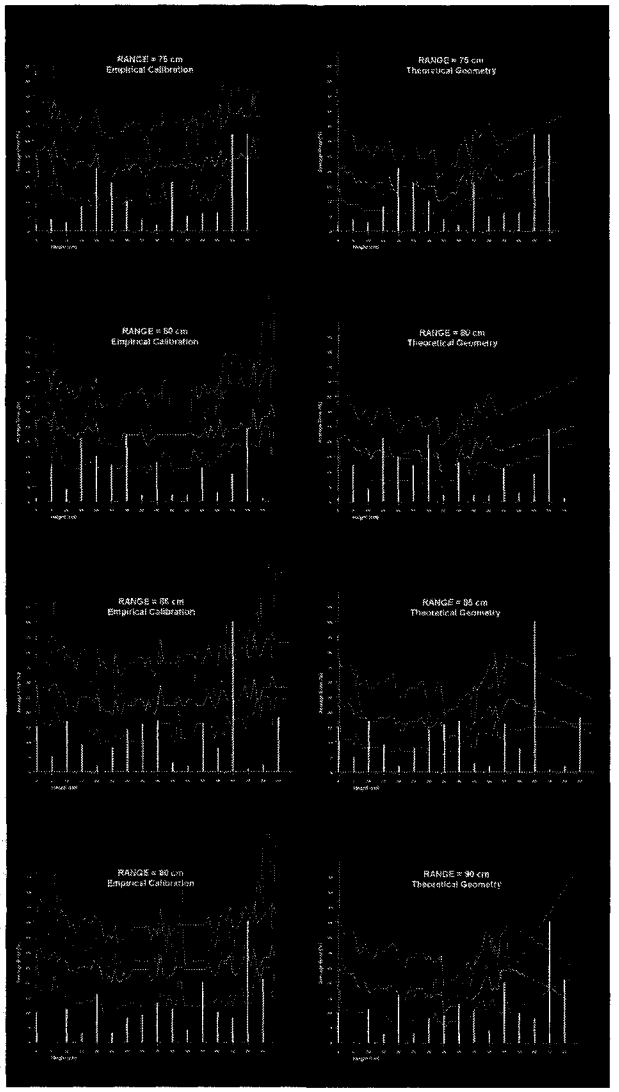


Figure 7.22: *Distance error at a range of 75 to 90 cm. Plot of expected distance error as a function of range and height for errors of one and two pixels in the inner and outer image radii.*

Chapter 8

Panoramic Stereo Reconstruction

8.1 Panoramic Stereo Reconstruction

The full panoramic stereo reconstruction system was tested with real images. An experimental panoramic catadioptric image sensor was assembled and scenes imaged and reconstructed. Results for six of these experiments are shown and discussed in this chapter.

Real imagery was captured with a Canon D-30 high resolution digital camera, a Canon EF 50mm lens and a double lobed spherical mirror. The dioptric camera and panoramic sensor system can be seen in Fig. 8.1 (right).

8.2 Experimental Catadioptric System

The camera and mirror were rigidly mounted with the Canon D30 camera directed vertically down at a bi-lobed mirror of (assumed) spherical profile. The two components were fastened to a custom made solid frame that occluded only 2.5 azimuth degrees (for the member that holds the camera above the mirror). This is an improvement over the the mount used for the mobile robot tracking that had some 25^o obscured and yet was less solid.

The Canon D30 was aligned vertically colinear with the mirror central axis. The Canon D30 (dioptric component) was positioned with its focal point 53.9 cm above the outer lobe, which had a radius measured to be 5.7 cm. The smaller inner lobe had a measured radius of 1.95 cm at a vertical distance of 48.7 cm from the Canon D30 focal point. The empirical measurement of these calibration parameters was performed in a procedure developed in Chapter 7. Because of the mirror having imperfections and not being an entirely spherical shape, these parameters were used only for feature extraction with the Panoramic Hough Transform. Unlike the synthetic image experiments, the spherical model could not be relied upon for calculating the 3D vector corresponding to each pixel. A different approach was required to give acceptable errors when converting pixel coordinates to 3D space vectors. The system is shown in Fig. 8.1 .

Within the captured images, the camera center was determined, and the inner and outer radii of both annular regions of useful image was found. This was determined as the range of the image of the mirror surfaces that corresponded to a view common to both lobes. The inner annulus, the reflection of the inner mirror lobe, had a width of 126 pixels (starting at a radius of 83) whereas the outer annulus was chosen to be 302 pixels (starting at a radius of 262). Overall 28.9 percent of the image pixels were usable. The loss was due mostly to the aspect ratio of the digital camera image size being rectangular whereas only a circular region was useful. Thus, much of the image on either side of the mirror was unusable. With radially symmetric mirrors and rectangular image capture devices, a low pixel usage rate is

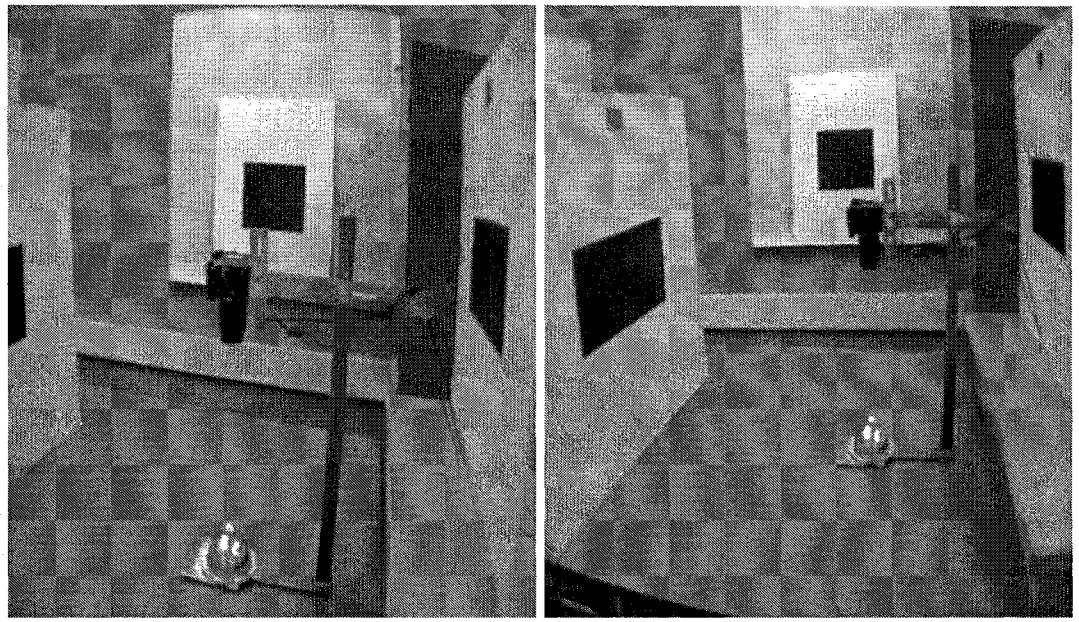


Figure 8.1: *(Left) Digital camera used for dioptric component of stereo panoramic sensor. (Right) Experimental setup.*

expected.

Fig. 8.2 shows a sample image taken from the scene in Fig. 8.1(right), extracted features and a reconstructed model.

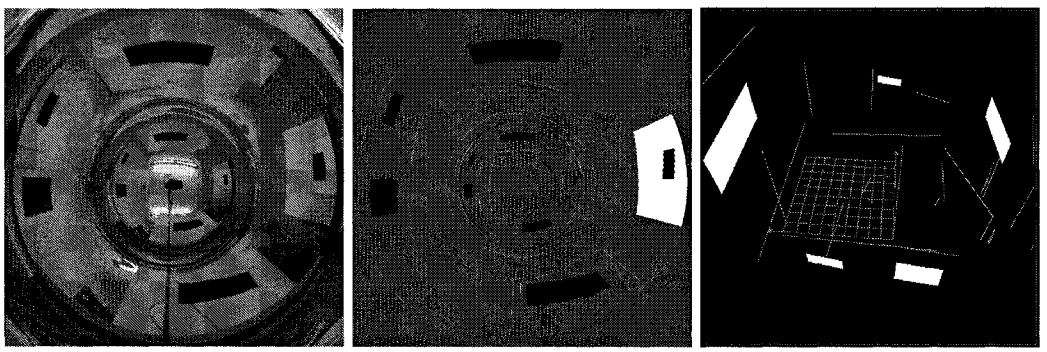


Figure 8.2: *Sample captured image (left), extracted features (middle) and reconstructed model (right).*

Extensive calibration was conducted (chap 7) with manually matched points to achieve the positional accuracy expected with the system parameters. A predicted position error function as a function of range was estimated for a pixel matching uncertainty of 1,2 and 3 pixels. The simplified error estimates underlay the collected error statistics in the graphs provided. In the aberration free sections of the mirrors this function can be used to estimate the expected error in the reconstructed 3D position.

Experiments were constructed using using black and white vertical rectangles positioned

at measured locations relative to the panoramic camera. One image was captured for each scene, and was passed twice through the feature extraction process of Chapter 5. The two feature sets (inner and outer lobe) were input to the matching and reconstruction stage. Tables (Chapter 7) were used instead of the assumed spherical geometry for reconstruction accuracy.

At this point a reconstructed model of the scene was generated automatically by the stereo panoramic sensor and subsequent vision processing Fig. 8.3. This model was compared to the manually measured model and the vertices compared. The percent distance error was computed for each vertex that was correctly identified. In the following images the reconstructed and correct model are overlaid in each 3D view for comparison, with the reconstructed rectangular faces shaded grey and the correct faces shaded white. Lines are drawn on some images between the correct and corresponding reconstructed vertices.

Six of the experiments are presented herein, chosen for their demonstration of the success and failure modes with this system. 3D line segments are extracted but the results' analysis focuses on the rectangular polygons, since the main focus of the research is to reconstruct polyhedral worlds. The original image, the extracted features, several views of the reconstruction, and a plot of percentage distance error is provided for each example. One false positive occurred in Experiment 2, a polygon is "detected" that did not exist. It was a darker region of the laboratory bordered on four sides by lighter straight edges. Since it was only found in one lobe, it did not create a false polygon in the output 3D model. This error would have been removed had a verification stage been done for rectangular projections in the same way it is performed by the tracking stages for segment projection features. False negatives did occur, where faces in the experimental scene did not make it into the model through a failure of feature extraction in one of the lobes. The experimental scenes were quite simple and relatively uncluttered, and the false negatives in these experiments were a result of a rectangular edge falling outside of the range of stereo view, or mirror imperfections.

The errors in position were as expected in that there was typically a large error in the reconstructed range of the polygon vertices, as opposed to a much smaller one in elevation and azimuth angle. In a mobile robot system utilizing this system, scene models from different viewpoints would greatly increase the accuracy. It should be noted that the error plots make the results appear worse than they are as that the largest errors for a given range are shown in the plots.

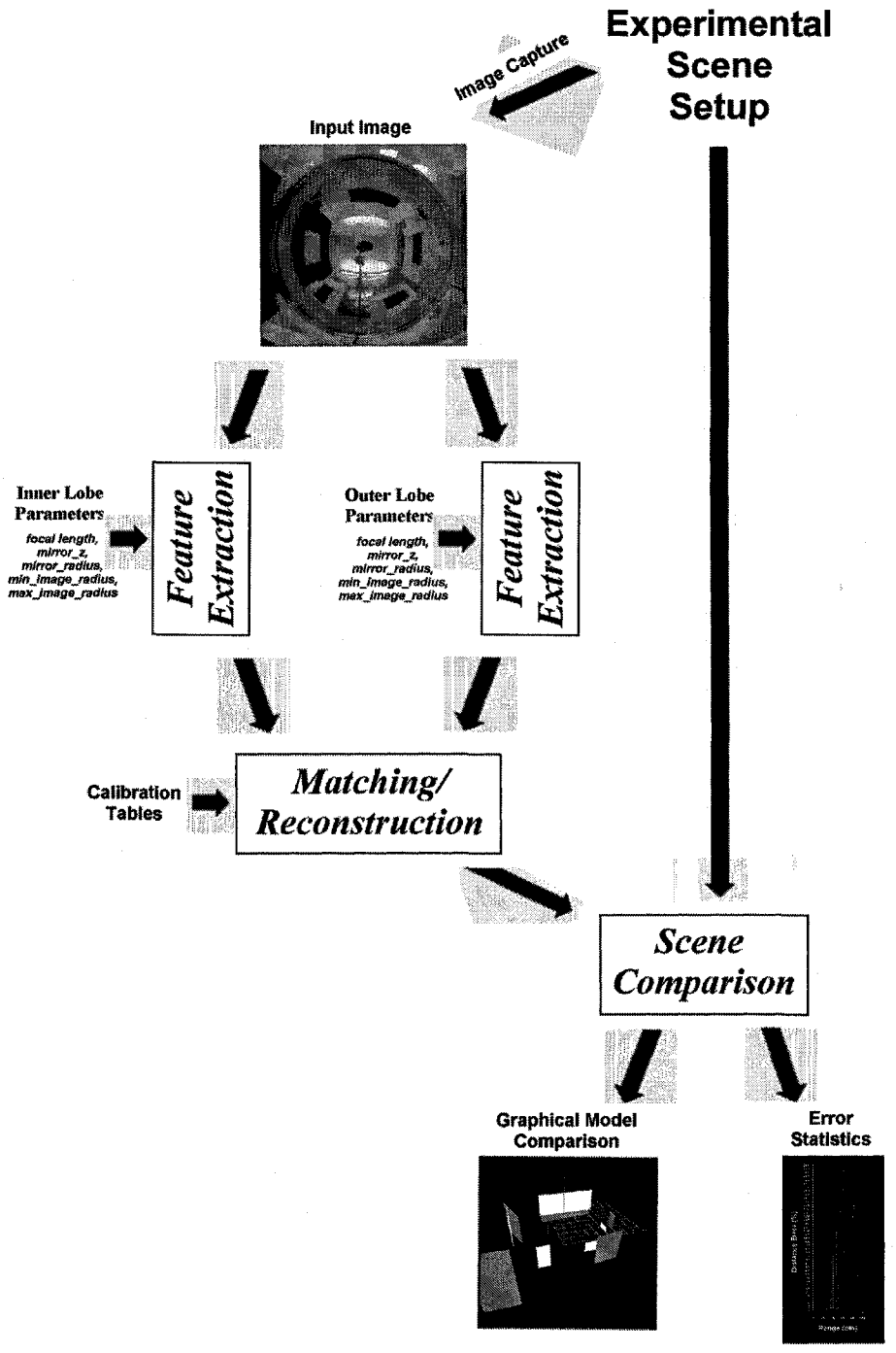


Figure 8.3: *Experimental Procedure: 3D reconstruction and verification.*

8.3 Experiment 1: Polygons at Medium Range (25-102 cm)

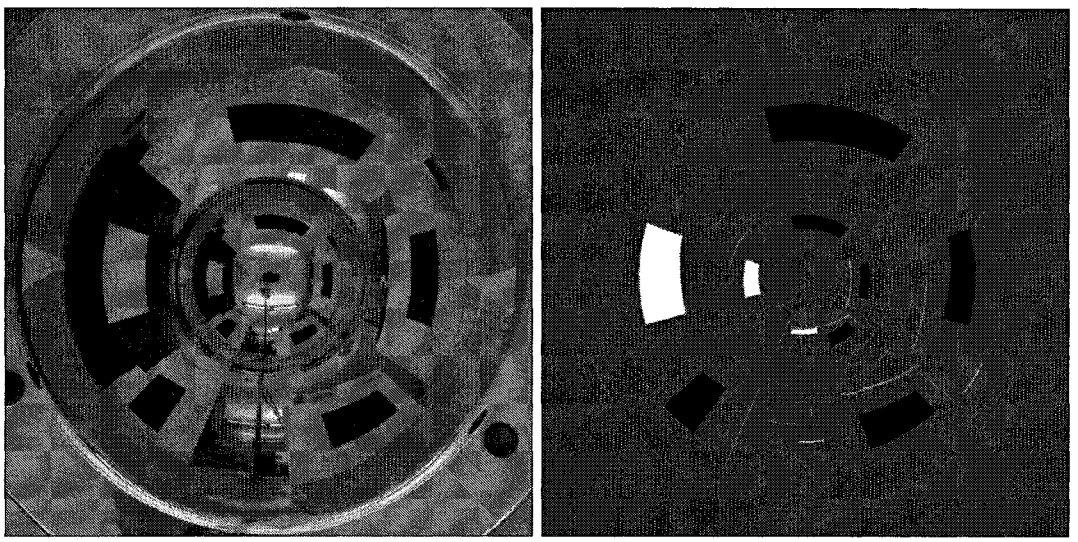


Figure 8.4: *Experiment 1: Original captured image (left) and extracted features (right).*

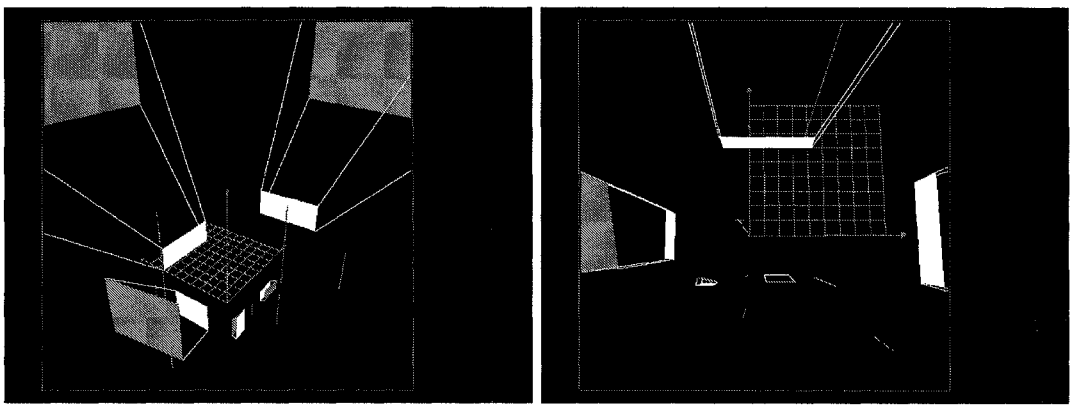


Figure 8.5: *Experiment 1: Two views of the stereo reconstruction. Correct polygons are shown in white, automatically reconstructed ones in grey. The grid squares (shown for positive X and Y values only) are 10cm wide.*

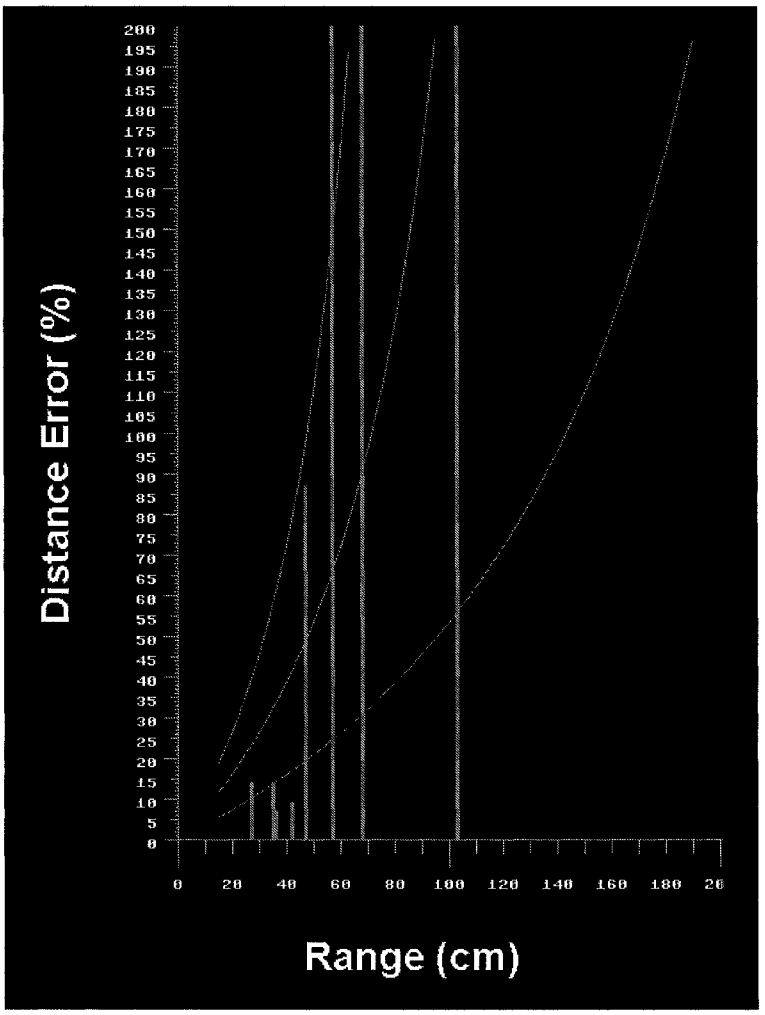


Figure 8.6: *Experiment 1: Plot of percentage error distribution. Vertical bars are error measurements, the curves from right to left are the estimated error according to a feature location error in the inner lobe of 1,2 and 3 pixels respectively.*

8.4 Experiment 2: Polygons at Close Range (25-50 cm)

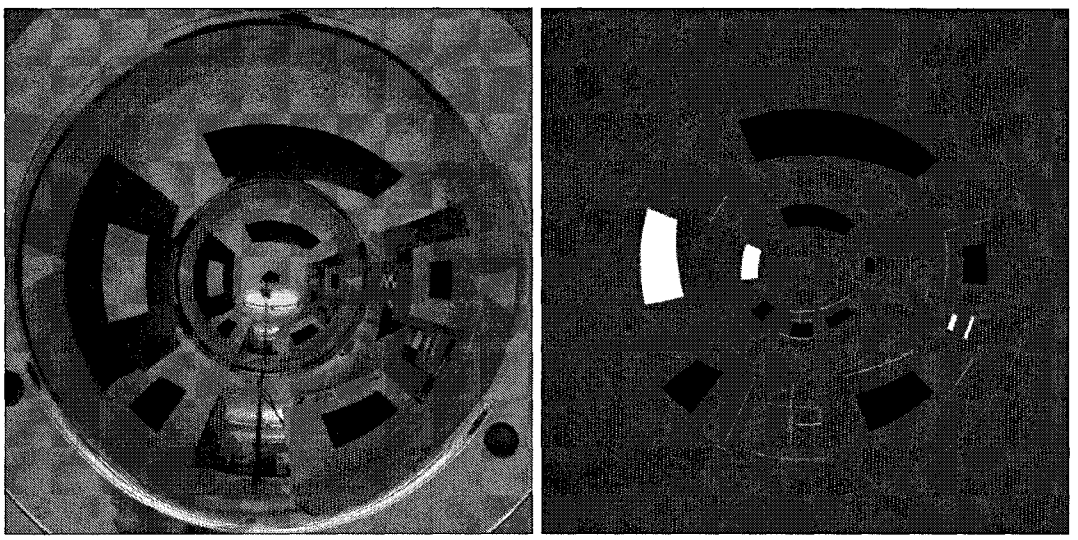


Figure 8.7: *Experiment 2: Original captured image (left) and extracted features (right).*

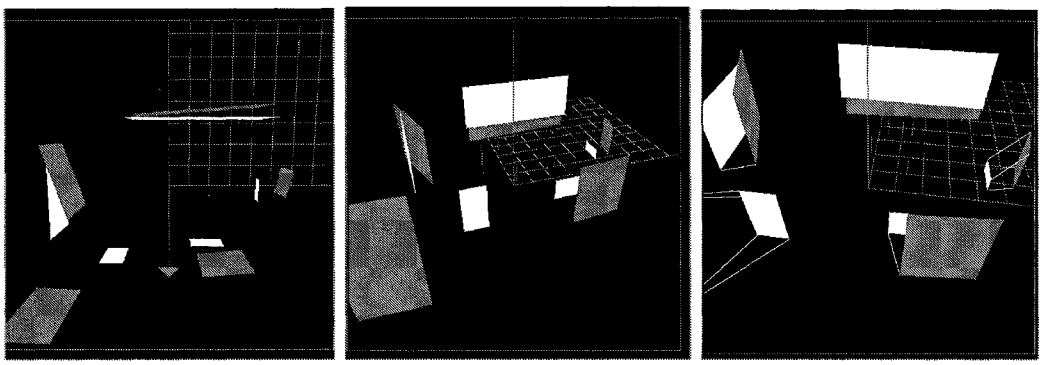


Figure 8.8: *Experiment 2: Three views of the stereo reconstruction. Correct polygons are shown in white, automatically reconstructed ones in grey. The grid squares (shown for positive X and Y values only) are 10cm wide.*

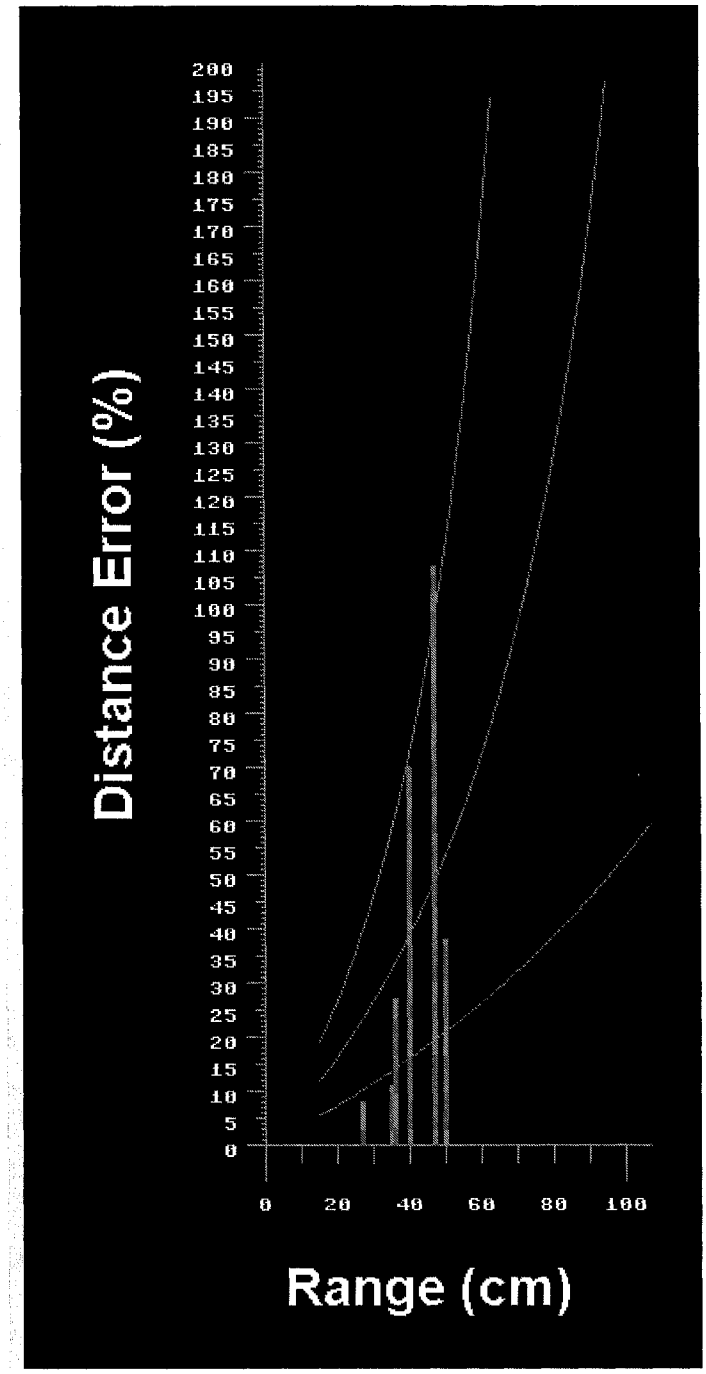


Figure 8.9: *Experiment 2: Plot of percentage error distribution. Vertical bars are error measurements, the curves from right to left are the estimated error according to a feature location error in the inner lobe of 1,2 and 3 pixels respectively.*

8.5 Experiment 3: Polygons at Close Range (25-50 cm)

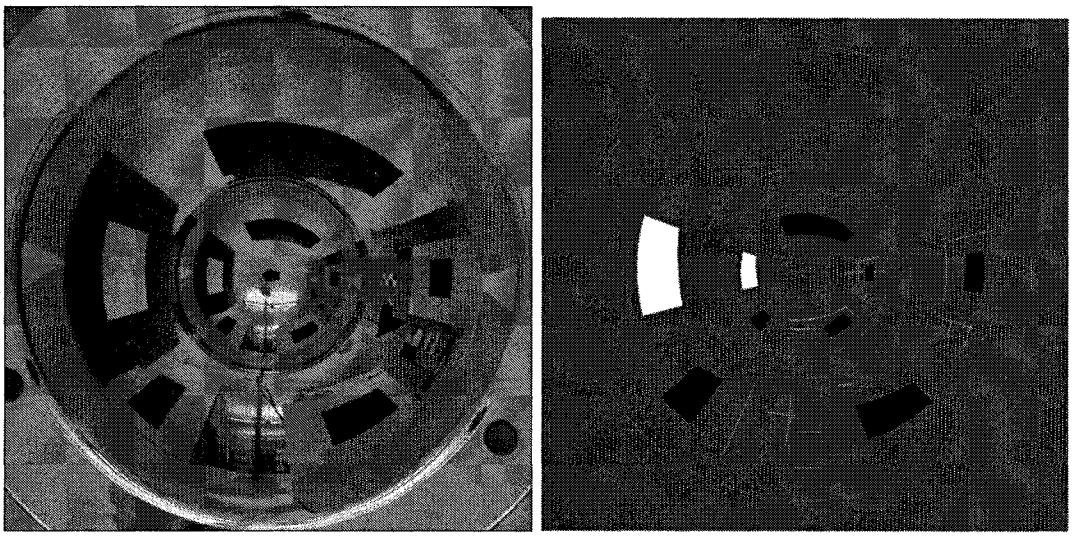


Figure 8.10: *Experiment 3: Original captured image (left) and extracted features (right).*

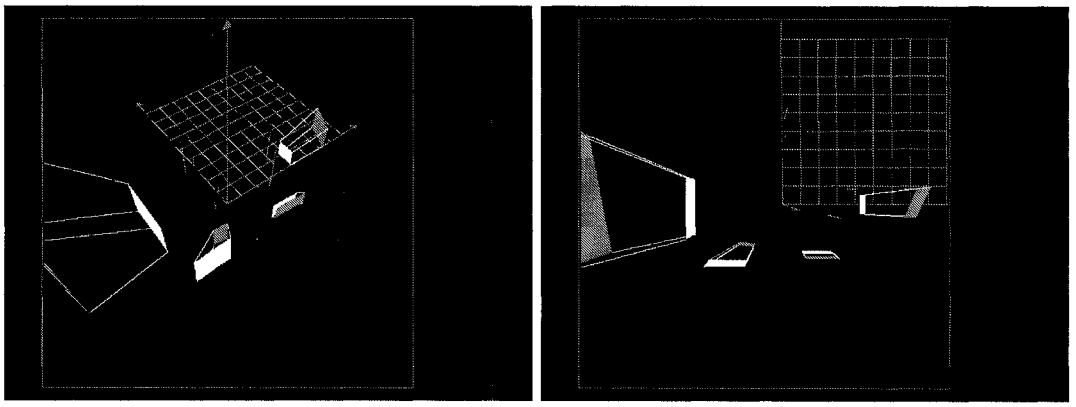


Figure 8.11: *Experiment 3: Two views of the stereo reconstruction. Correct polygons are shown in white, automatically reconstructed ones in grey.*

The reconstruction error is much better for experiment 3, primarily due to the closer range of objects. Note in the reconstruction how the error increases with range.

The reconstruction of Experiment 3's scene is missing a polygon. Fig. 8.13 shows where the failure occurred. The top edge of the vertical black rectangle was not completely inside the stereo viewing volume, it can be seen completely in the outer lobe, but the center of the top edge violates the inner minimum radius. Thus a match for the feature in the outer lobe list does not have a match, and a reconstructed polygon is not created in the final 3D model.

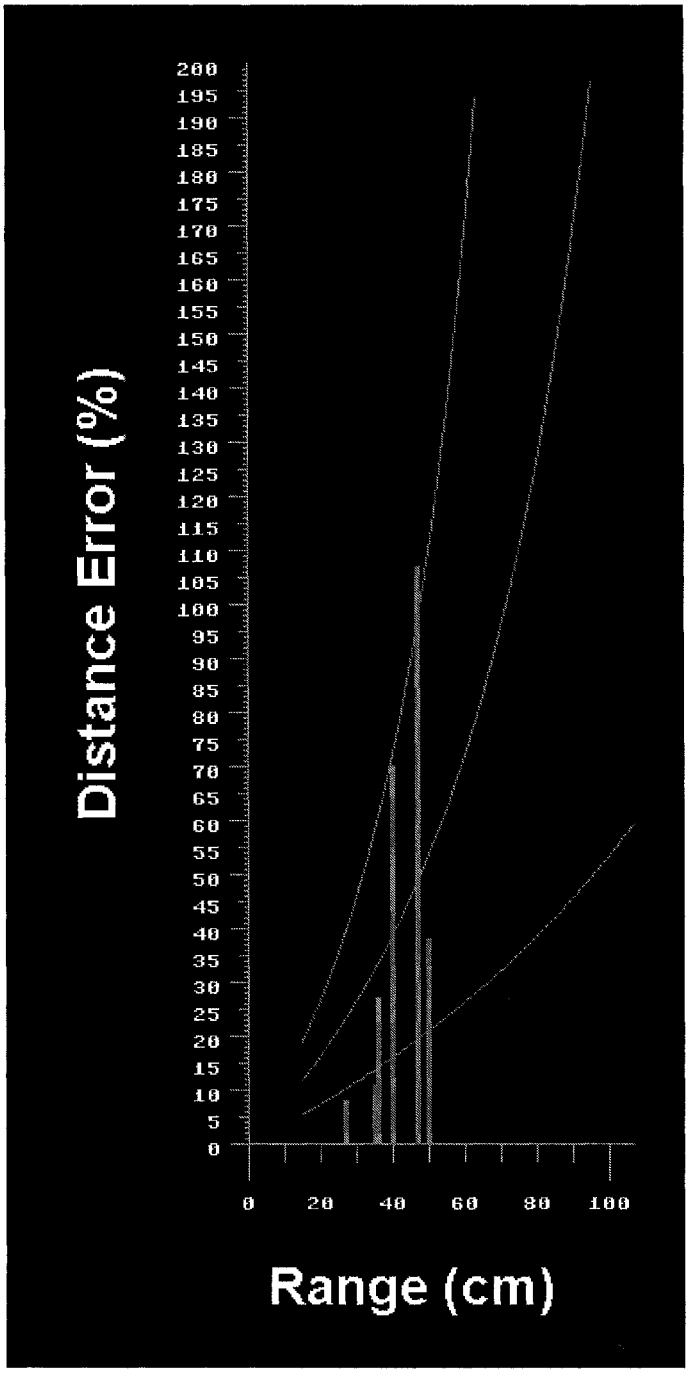


Figure 8.12: *Experiment 3: Plot of percentage error distribution. Vertical bars are error measurements, the curves from right to left are the estimated error according to a feature location error in the inner lobe of 1,2 and 3 pixels respectively.*

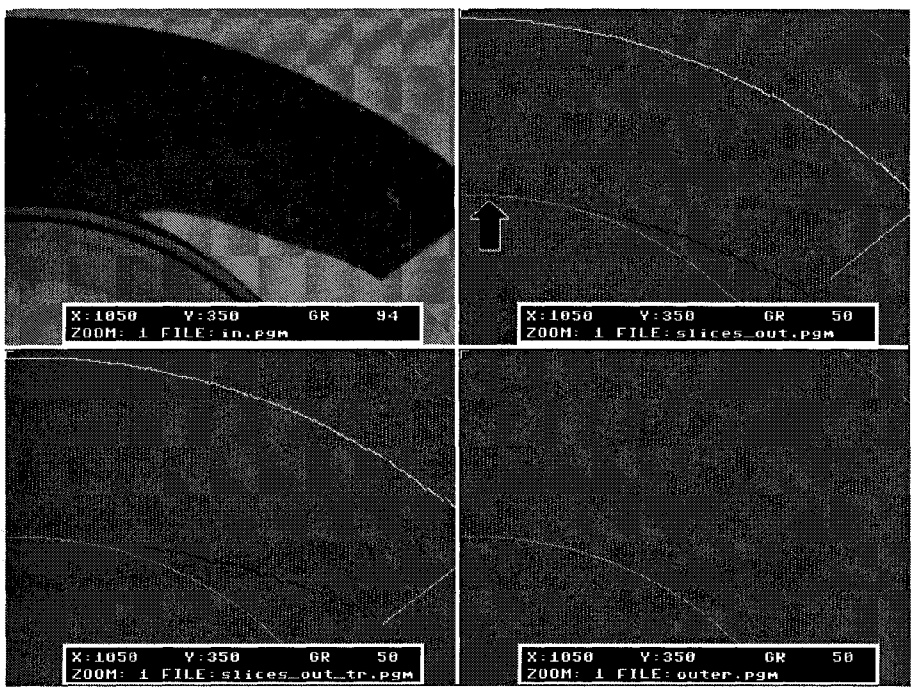


Figure 8.13: *Experiment 3: False negative detection of top vertical rectangle projection in outer lobe image.. Close-up of feature extraction (4 tiled images). One polygon is missing from the reconstruction, plot of feature extraction shows how the top edge was too high/close to the camera and so an edge is missing. The arrow in the top right image points to the circular line which represents the minimum radius parameter for this lobe/ The four tiled images: Original image (top left), output of feature detection before first stage of tracking (top right), output after both tracking stages complete (bottom left), output of rectangle extraction with no recognized rectangle projection (bottom right).*

8.6 Experiment 4: Polygons at Close Range (28-40 cm)

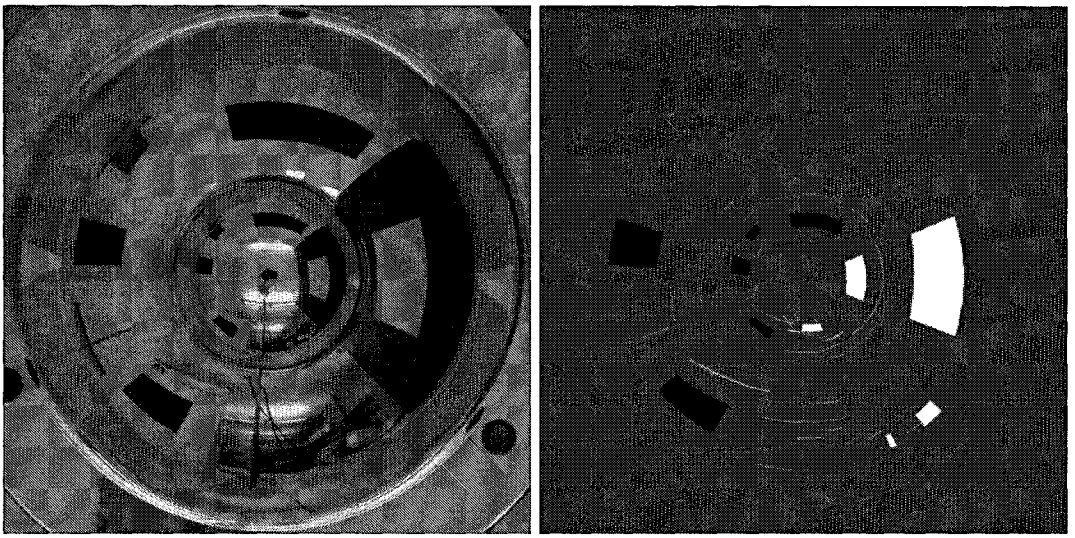


Figure 8.14: *Experiment3: Original captured image (left) and extracted features (right).*

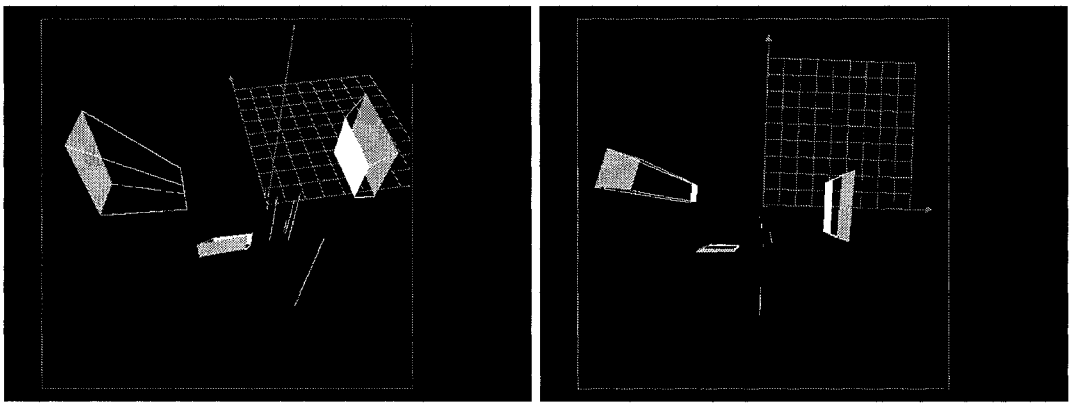


Figure 8.15: *Experiment 4: Two views of the stereo reconstruction. Correct polygons are shown in white, automatically reconstructed ones in grey.*

Experiment 4 shows three false positives for the white rectangles, but since none of them find a supporting match in the other lobe, they do not create a false positive in the final 3D model.

Also of note is the undetected top black polygon, the rectangle feature is not detected in the outer lobe. The reason is shown in Fig. 8.16 where a mirror aberration causes a bulge-like error in the top edge (lower edge in the image). The Panoramic Hough Transform stage attempts to fit these edge pixels and produces two similar, partially overlapping feature definitions. Fig. 8.16 (right) shows how the tracking cannot repair the damage since the shape simple does not correspond to a curve expected according to the non-SVP horizontal

line projection theory.

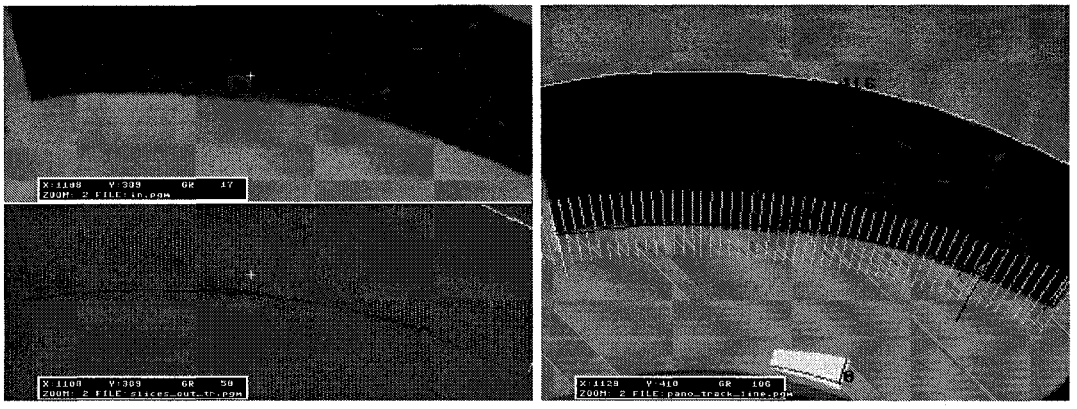


Figure 8.16: *Section of captured image showing a bulging of the polygon edge due to a mirror imperfection(left top). Feature detection cannot map a single horizontal projection curve to the edge, two incomplete overlapping segment features are produced (left bottom). Tracking procedure does not follow edge to correct end, black and white markers show endpoints of the tracked feature (right).*

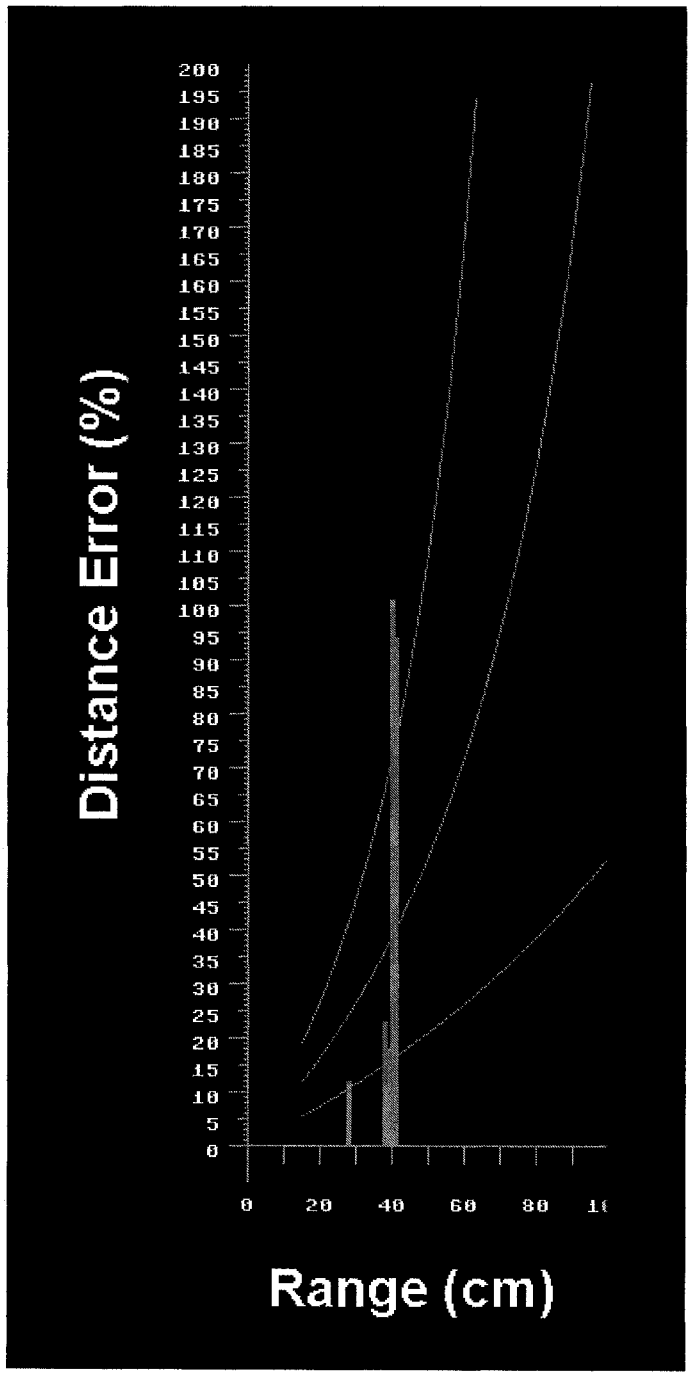


Figure 8.17: *Experiment 4: Plot of percentage error distribution. Vertical bars are error measurements, the curves from right to left are the estimated error according to a feature location error in the inner lobe of 1,2 and 3 pixels respectively.*

8.7 Experiment 5

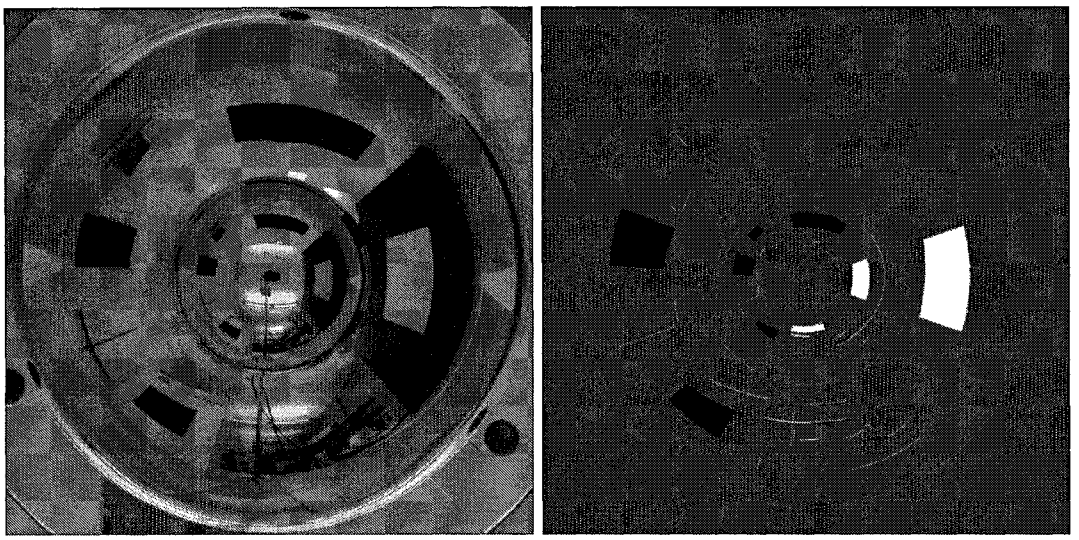


Figure 8.18: *Experiment 5: Original captured image (left) and extracted features (right). Correct polygons are shown in white, automatically reconstructed ones in grey.*

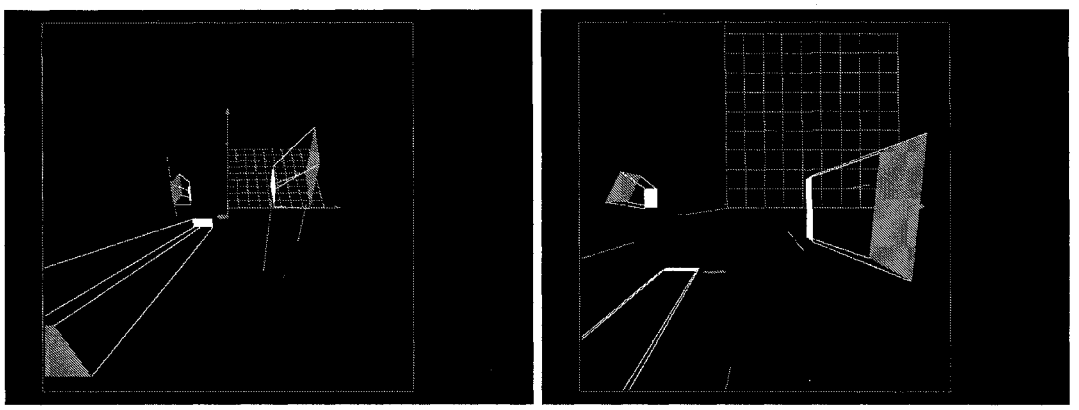


Figure 8.19: *Experiment 5: Two views of the stereo reconstruction.*

The sensitivity of the system to small errors can be seen in the dramatic error of the lower left rectangle's reconstruction. This was due to an erroneous feature detection of the polygon's projection in the inner lobe.

The top black rectangle is missing again in Experiment 5 for the same reason as in Experiment 4, because the edge falls on a mirror aberration.

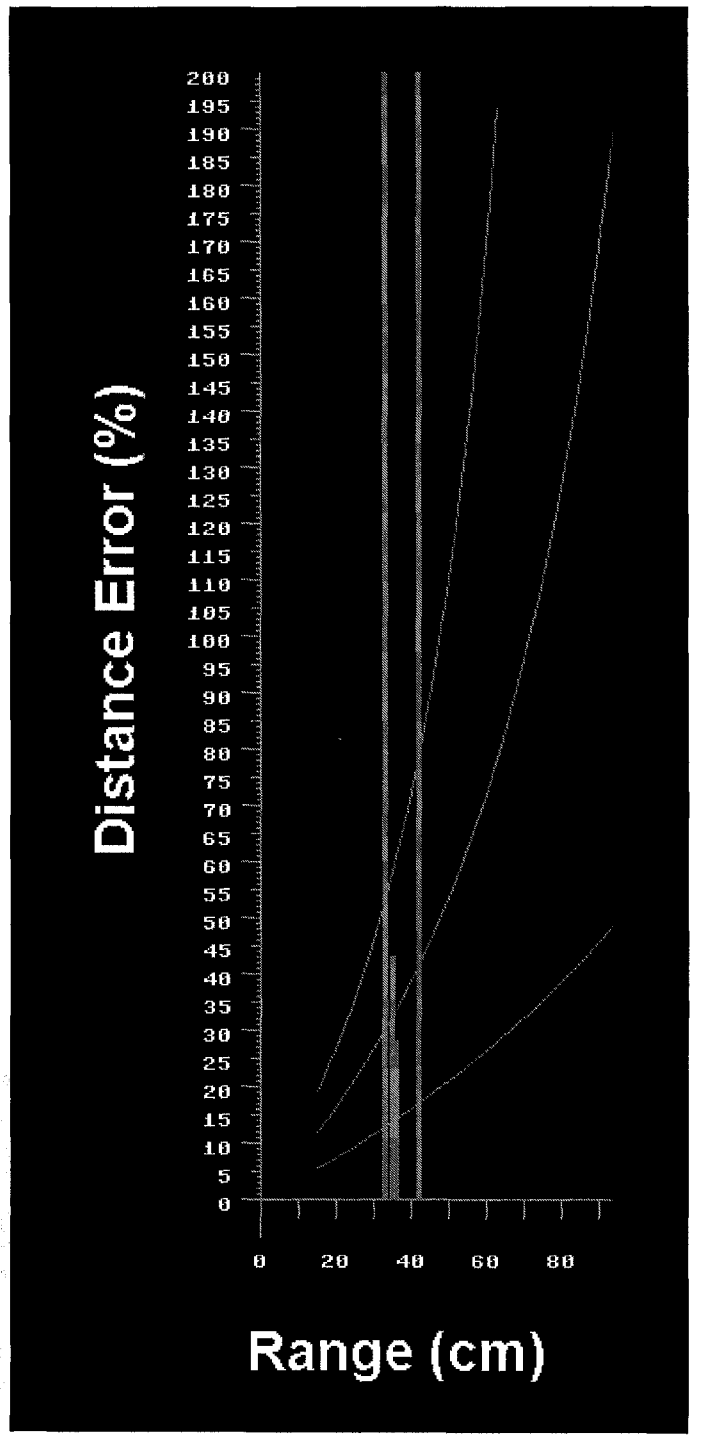


Figure 8.20: *Experiment 5: Plot of percentage error distribution. Vertical bars are error measurements, the curves from right to left are the estimated error according to a feature location error in the inner lobe of 1,2 and 3 pixels respectively.*

8.8 Experiment 6

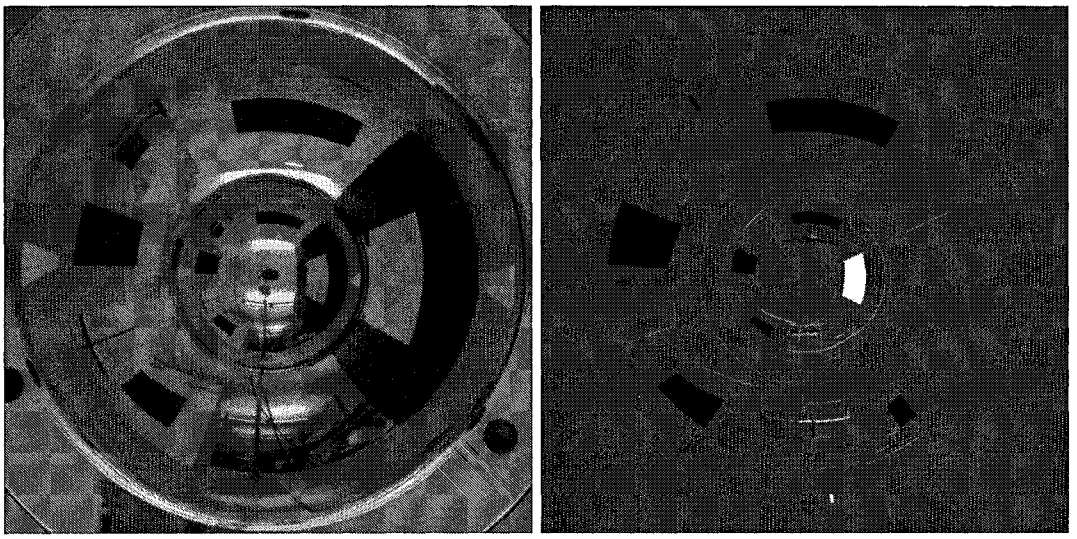


Figure 8.21: *Experiment 6: Original captured image (left) and extracted features (right).*

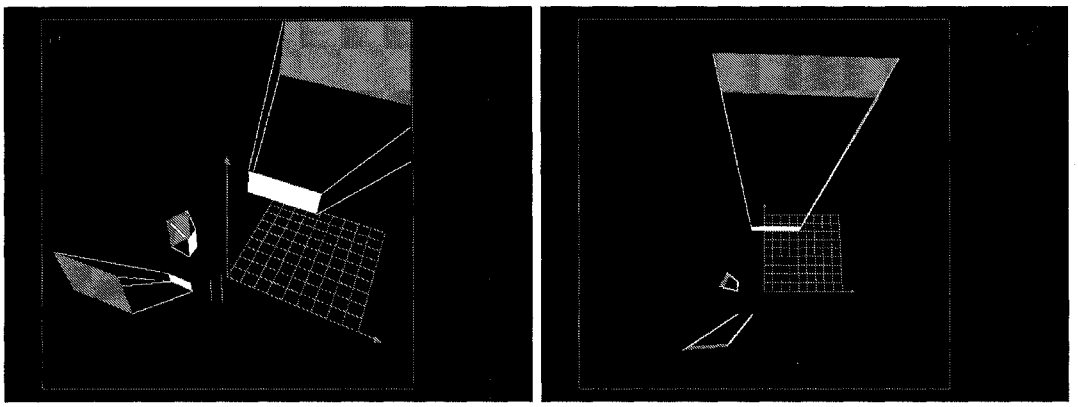


Figure 8.22: *Experiment 6: Two views of the stereo reconstruction. Correct polygons are shown in white, automatically reconstructed ones in grey.*

The white rectangle on the right is not represented in the outer lobe feature set, and hence missing in the 3D model because of the same reason as in Experiment 2, due to the top edge being slightly outside the stereo viewing volume.

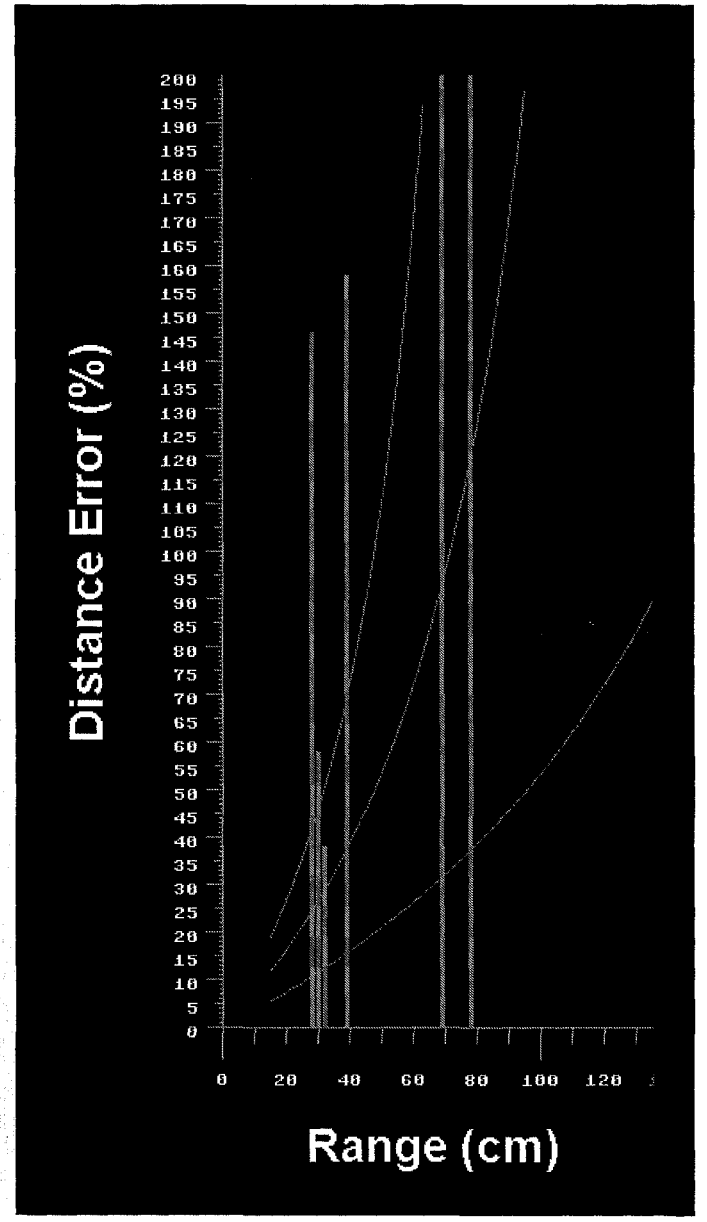


Figure 8.23: *Experiment 6: Plot of percentage error distribution. Vertical bars are error measurements, the curves from right to left are the estimated error according to a feature location error in the inner lobe of 1,2 and 3 pixels respectively.*

Chapter 9

Mobile Robot Localization Using Panoramic Landmark Tracking

The main two research thrusts using the Panoramic Hough Transform are stereo reconstruction and mobile robot localization. A feature based robot localization sytem was developed using the PHT based on recognizing and tracking landmarks with a single lobed catadioptric panoramic image sensor.

The application of the PHT provides an improvement over current robot *localization* panoramic schemes which typically use only vertical lines, or image-based approaches. Being able to find the projections of horizontal lines in panoramic non-SVP catadioptric images is an important improvement over the status quo, since the tasks of tracking and environment object recognition can operate at the polygon level. Indeed the author claims that the application detailed in this chapter is the the most advanced passive panoramic navigation system in the literature at the time of this writing.

This chapter summarizes some of the other work on mobile robot localization, focusing on catadioptric panoramic systems. Two methods developed by the author are then described, the latter one of which uses the PHT and was implemented in a real robot system.

9.1 Thesis Mobile Robot Vision System

A vision based navigation system is presented for determining a mobile robot's position and orientation using panoramic imagery. Omni-directional sensors are useful in obtaining a 360° field of view, permitting objects in the vicinity of a robot to be imaged simultaneously. The navigation system uses a catadioptric panoramic image sensor with a single lobe, and tracks the position based on an *a priori* map of landmark polygons and their vertices. The system visually tracks the projection of vertical rectangular polygons based on an estimated position and orientation. After the locations of the polygons' projections have been found, the new position of the robot is found through triangulation.

Two approaches were tested, differing in the manner in which they tracked scene polygons. The first used corner tracking on a quasi-cylindrical image, and either light or dark corners were located from a seed position with a corner detector. The second method was more robust, and uses the Panoramic Hough Transform tracking algorithms to track the projections of the horizontal and vertical line segment projections that formed a vertex. The second approach was expected to be much more robust, this was verified in experiments with synthetic and real image sequences, and was successfully implemented with a mobile robot.

9.2 Tracking Landmarks for Robot Navigation

One fundamental component for an autonomous mobile robotic platform is to determine its position and orientation with respect to its environment. Example positioning systems use sonar sensors, motor odometry and radio beacons [19]. Most robotics research with mobile platforms has focused on sonar systems, and there is a wealth of related research, hobby and commercial robot platforms and literature using sonar [109, 107, 89, 58]. The author built a mobile robot with a 16 sensor sonar ring in 1997. However vision systems offer much greater promise and emphasis has been shifting to vision guided navigation.

This chapter focuses on landmark identification and tracking, what to do with landmark information and how to represent the environment is a separate field of study. Methods used for map making, landmark detection using both metric, euclidean maps and topological graph based environment depictions are found in [88, 53].

A passive vision-based system would be very advantageous, and increase the practical utility and scalability of mobile robotics. Hager and Rasmussen define a framework for robot navigation using standard perspective cameras [45, 53, 44, 34]. If this vision system was panoramic, objects all around the robot could be used for finding and updating the position estimate.

As with 3D stereo reconstruction, a vision process for landmark recognition can be classified as seeking geometric features such as edges, or correlating matches with a textured surface. The first would be useful for man-made environments, typically consisting of smooth featureless planes meeting at abrupt, distinct edges, an example of the latter could be the rocky topography as seen by a Mars rover. The former scenario would provide good results for a feature based system, but sparse results for unique correlation matches. Similarly the vision system for indoor use might not be of much use in random, natural environments. This thesis has focused on the first system type, which extracts and tracks feature primitives corresponding to polygon boundaries.

The paradigm of an agent translating along a horizontal plane and rotating about a vertical axis, in a space defined and filled with rectilinear polyhedral objects is a reasonable assumption for a mobile robot operating in a man-made environment. Indeed most indoor scenes can be well defined by the primitives of horizontal and vertical lines, corners where such line edges meet, and rectangular surfaces with only horizontal and vertical edges.

Other mobile robot localization research using panoramic cameras also assume an indoor environment, and rely on vertical polygon edges only [27, 29, 108]. Utilizing horizontal elements can improve robustness by better uniquifying these vertical edges.

We define a framework for creating a passive vision-based navigation system as that of iteratively predicting the location of a prominent geometric feature composed of horizontal

and vertical lines, tracking the location of the feature in a panoramic image, and aggregating a number of such observations to arrive at a new position and orientation estimate. We are restricting the problem to that of a mobile agent with motion possible only along the horizontal plane, reducing the dimensions of navigation data to three, two for position and one for orientation. The presence of an *a priori* model of prominent landmarks for predicting trackable features is assumed.

9.2.1 Robot Navigation Using Panoramic Imagery: Other Research

Omni-directional viewing would require many standard narrow field of view cameras, a rotating narrow field of view camera, or one panoramic camera. There is an abundance of research where panoramic images are obtained by mosaicking many images from a revolving camera. However requiring a robot to stop and spend several seconds spinning a camera around, and relying on a static environment reduces the practicality of such a system. Clearly a system for real-time use on a moving mobile robot would benefit from gathering a panoramic "snapshot" in one image, preferably at some fast rate like the 30 Hz NTSC video rate.

Other researcher have built mobile robot systems using catadioptric panoramic cameras, indeed many with non-SVP mirrors. Betke and Gurvits use a spherical mirror on their "ratbot" [16]. Fig. 9.1 shows a mobile robot with a catadioptric camera of spherical profile in the work of Sivic and Pajdla [90, 103]. Imagery from this robot was used by Jogan and Leonardis [64, 76] to create an image based approach to localize a robot in an environment according to image matches to *eigenimages*.

Image-based methods which attempt to localize cameras from image matches to eigenvectors images have been applied to panoramic imagery, collected both with the catadioptric and rotating camera approach, with work such as [78, 87, 28, 8]. Ulrich and Nourbakhsh use an *Omnicam* (a commercial product using an SVP panoramic camera) [111] to determine location by matching colour histograms of environments. They are not using the SVP properties of the Omnicam. Svoboda [105] proposes a method to take advantage of the projection of lines as circles in panoramic SVP systems for navigation and stereo vision, but mentions no concrete plans and implementations.

Research in the literature that does not use image or *appearance* based methods typically rely on vertical edges only. Yata, Ohya and Yuta [108, 107, 109] use a panoramic catadioptric camera to complement sonar data, by assuming a polyhedral world of only vertical rectangles. Their work uses panoramic vision by using considering an annular region corresponding to the horizon elevation, and treating it as a one-dimensional signal as a function of azimuth angle and searching for step edges. Betke and Gurvits [16] also use a

spherical mirror and use only pixels from a single radius, they compare this one-dimensional signal to stored one-dimensional landmarks to estimate position.

An environment with vertical lines is also assumed in the robot localization approach of Pegard and Mouaddib [23] with the *SYCLOP* sensor [20, 22] which uses a conical mirror. The hough transform is used to find straight radial lines corresponding to vertical edges.

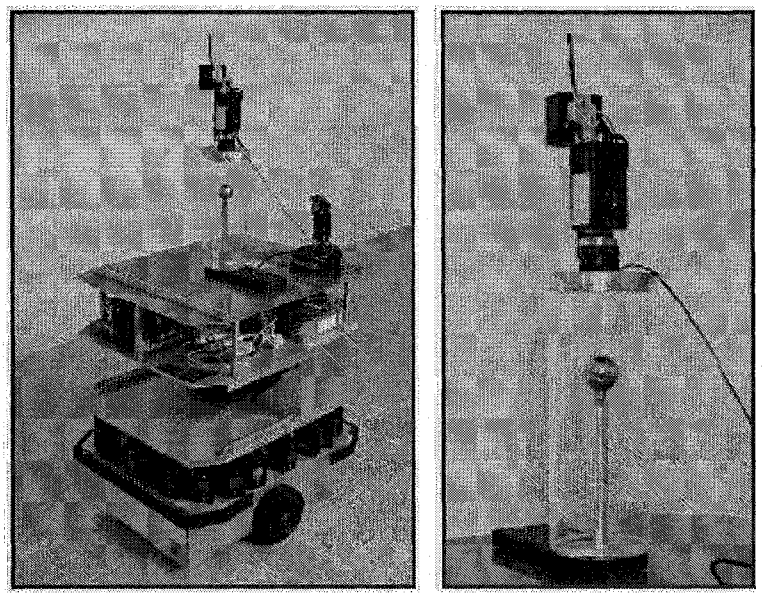


Figure 9.1: *Mobile robot with a non-SVP spherical mirror from the Center for Machine Perception at the Czech Technical University in Prague.*

Marques and Lima [73] and Andrea [6] use a panoramic camera for localization of their robots in the *Robocup* indoor soccer contest using Hicks' [55] mirror profiles. This is an ingenious method that exploits the fact that a position on the Robocup field can be determined by looking down only. Unfortunately this mehod is not usable for objects at or above the horizon, and hence is not classified under the heading of panoramic vision in this thesis.

In summary, approachs found in the literature surveyed using catadioptric panoramic cameras for robot localization rely on vertical line edges, or some image-based approach to estimate location based on some non-spatial image statistics. The novelty of this author's approach is the use of the Panoramic Hough Transform to find horizontal line segments, to allow entire vertical polygons to be used for landmarks.

9.2.2 Robot Navigation Using Panoramic Imagery: Thesis Research

Non-SVP catadioptric sensors permit the capture of a panoramic view, but with the added challenge of finding and tracking objects in a non-perspective projection. Panoramic catadioptric cameras are not necessarily even cylindrical projections, and locating and tracking

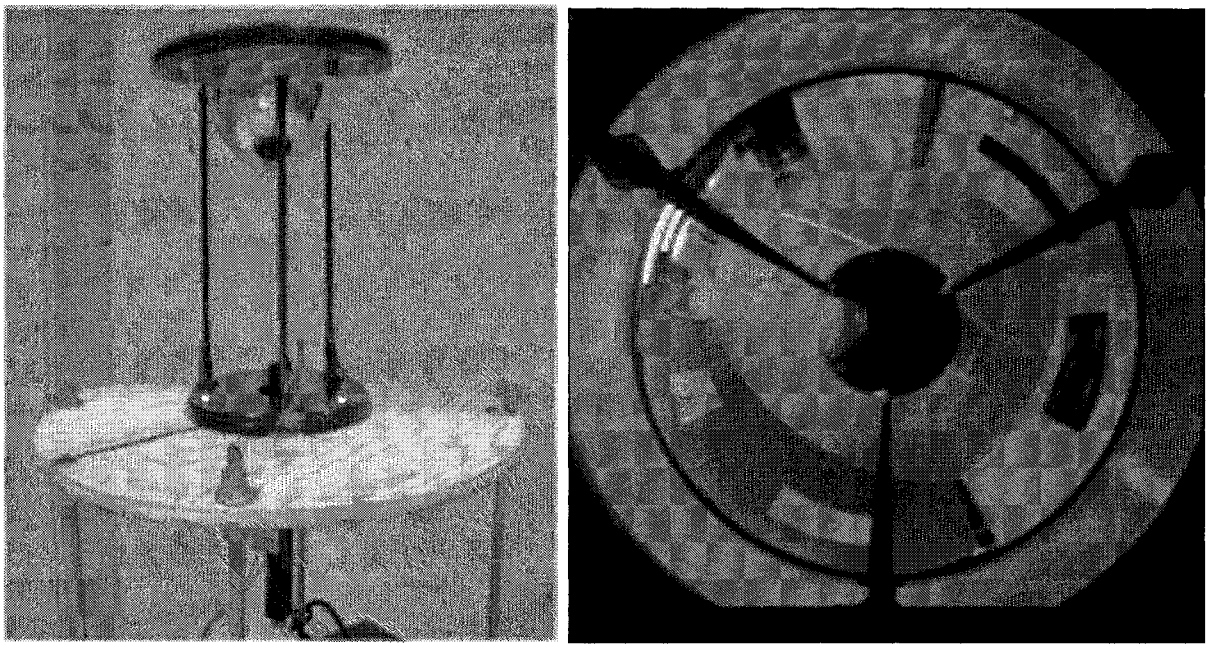


Figure 9.2: *Mobile robot used in the author's research (left). A Panoramic imaging system using a non-SVP catadioptric panoramic image sensor was used to implement a landmark tracking system based on the panoramic hough transform. A sample panoramic image is also shown (right).*

features (especially lines) introduces challenges for tracking not found in more traditional perspective view imagery.

The first positioning method demonstrated assumes a cylindrical projection, or at least a quasi- cylindrical view and uses vertices as the tracking primitive. The second method shown is more novel. It is a more robust method tracking horizontal and vertical lines, and uses the intersection thereof as the landmark primitive. Issues with using panoramic optics are addressed, specifically that of the absence of the preservation of the straightness of line features, a phenomenon that benefit traditional perspective image analysis. The *Panoramic Hough Transform* is a tool utilized to aid in line detection without restricting the mirror geometry to achieve a pseudo-perspective projection.

9.3 Triangulating Location From Landmarks

The landmark tracking function provides a set of detected landmarks and their azimuth angle. If the the mobile robot is constrained to movement on a horizontal plane, the elevation of landmarks is used in the tracking, but only the angle is required to find position [27]. Thus the result of landmark tracking need only be a set of landmark labels and their θ_l angles. The triangulation method below is basic, but included for completeness.

If the robot's camera orientation angle θ_c is known, then the position must lie along the line drawn from a detected landmark's world model position, along the azimuth angle $\theta_l + \theta_c$. This line can be described in the $aX + bY + c = 0$ form. The camera location can be determined from the convergence of all such lines. Assuming equal confidence for all θ_l angles, the camera position (X_c, Y_c) can be found by the method of least squares (Eqn. 9.1), where the quantity minimized is the perpendicular distance from the camera position to all the lines (Eqn. 9.2).

$$\begin{aligned} X_c &= \frac{-\sum b^2 \sum ac + \sum bc \sum ab}{\sum a^2 \sum b^2 - (\sum ab)^2} \\ Y_c &= -\frac{\sum a^2 \sum bc - \sum ac \sum ab}{\sum a^2 \sum b^2 - (\sum ab)^2} \end{aligned} \tag{9.1}$$

$$\begin{aligned} \sum dist^2 = X_c^2 \sum a^2 + 2X_c Y_c \sum ab + \sum c^2 \\ +2X_c \sum ac + Y_c^2 \sum b^2 + 2Y_c \sum bc \end{aligned} \tag{9.2}$$

The above assumes a known θ_c. It was found to be sufficient to calculate θ_c and (X_c, Y_c) independently since a θ_c value within $\pm 45^\circ$ of the correct value yields almost the same position. Numerical iteration with θ_c to find a minimum of Eqn. 9.2 produces θ_c, which is then used in Eqn. 9.1 to find (X_c, Y_c). This phenomenon is shown in Fig. 9.3 where an observed set of azimuth readings are matched to landmark locations, and lines drawn back from the landmarks for several assumed θ_c headings. The cross denotes the least squares error (X_c, Y_c) for the different θ_c values, and the error in position is negligible compared to the error in θ_c. Thus an estimate can be made from any roughly estimated θ_c.

The Newton-raphson method could be used to reduce the number of iterations, however it was not known for sure that there would be no local minimums so a more intensive search was done. The procedure actually used in the experiments is that a range of θ_c values were tried, and the two lowest that produced the lowest error according to Eqn. 9.2 set the bounds for a finer tuned set of θ_c. This iterative approach was done 3 times to get θ_c, and Eqn. 9.1 was only applied once to get (X_c, Y_c). It should be noted that the cost of these calculations is very low compared to the overall computational burden of the image processing.

This method showed a high degree of robustness, even without confidence measures for the landmark taken into account. In general, if a large enough set of landmarks are available, the required accuracy rate for the landmark detector can be relaxed. For a small set of landmarks a higher accuracy is required. The line-junction method is more accurate and less landmark polygons are required in the environment.

Further enhancements could attempt to reject outliers after (X_c, Y_c) is determined. Data that produces a line more than a threshold away from (X_c, Y_c) could be removed, and the

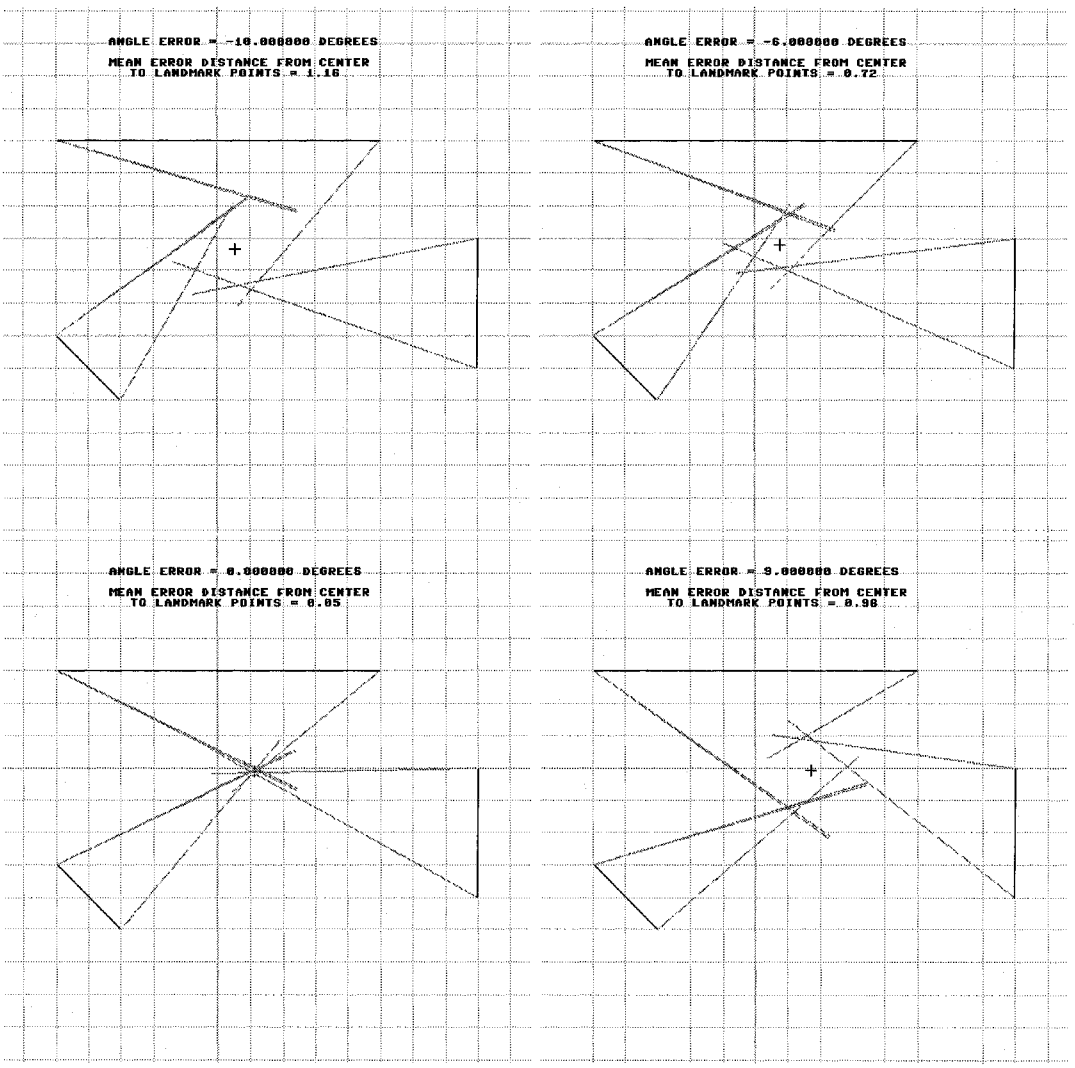


Figure 9.3: *Empirical demonstration of the separation between robot position and heading. Four different choices for the robot's assumed heading. The estimated location is given by the closest point to all lines drawn back from the observed landmarks. Only the bottom left image has the correct heading, but the estimated location (shown as a cross) does not vary despite errors in the assumed heading.*

triangulation process repeated.

9.4 Landmark Visibility Prediction and Feature Hypothesis

The localization problem is expressed as tracking image features based on predicted landmarks. According to an estimate of the sensor location, a list of predicted features (projections of scene objects) are tracked in the panoramic image. A sub-system must determine which landmarks are visible, and where in the captured image they are expected to lie. The design of a *visibility filter* to achieve the former is entwined with the method used to define the world map.

The landmark prediction should take into account what vertices are visible from each location in the map. This could be encoded in the map as visible landmark polygons, as is performed in the "Doom" video game 3D rendering system. A *graph*-based approach could assign landmarks to a node in the topological world model as in Hager's method [53]. Alternatively as in this these experiments, a metric map of the environment and a simple heuristic for a landmark visibility filter was used. The criteria for visibility was a minimum distance and viewing angle. The viewing angle was determined by finding the dot-product between the vector from sensor to vertex, and the polygon normal vector in a similar approach to *back-face culling* used in computer graphics.

The effect of incorrectly versus correctly predicting visible vertex landmarks is shown in Fig. 9.4. This example shows vertex landmarks but the same filtering process is performed for the line junction tracking method.

After it has been determined what landmark vertices or line projections are visible from the estimated position, a hypothesis can be formed of where the features are expected to be found in the image. The angle is simply calculated from the 2D geometry, but the radius of image points need the sensor dimension and profile information in the form of a function $R_{feature} = F(Z_{landmark}, RANGE_{landmark})$. In this way, projections of line segments and vertices are predicted according to the current estimate of the mobile robot position.

9.5 Tracking Vertex Landmarks

In this landmark extraction method, tracking is performed with a corner detector on the landmark vertices. Line edges and hence the PHT, are not used in this approach. Corners are located by a template matching corner detector function in a warped view of the captured

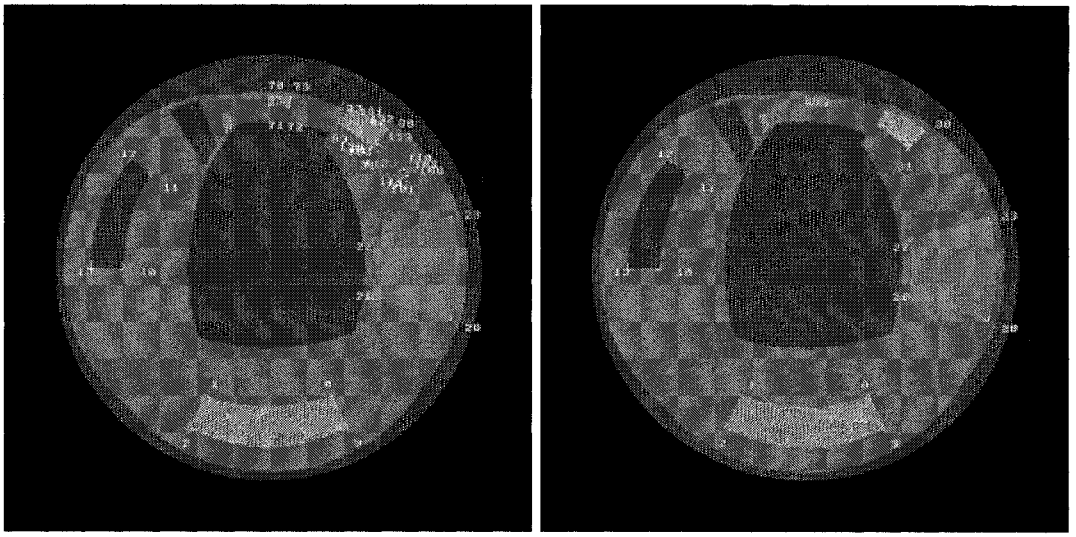


Figure 9.4: *(Left) Incorrect choice of visible vertex landmarks (all are shown). (Right) Filtering predicted landmarks according to visibility.*

image. A predicted location is given for each corner, plus a descriptor of the expected corner type. Since we are looking for corners of polygons composed of horizontal and vertical edges, there are eight types of corners to which this descriptor can refer. The corner can be to the upper left or right, lower left or right, and this corner region can either be lighter or darker compared to its neighborhood.

The useful annular region of source image is warped to a quasi-cylindrical view as in Fig. 9.5a and Fig. 9.12a using bi-linear interpolation. By convention the long rectangular image has pixels representing the azimuth direction in the X direction and radius in the Y direction. This quasi-cylindrical view is a true cylindrical projection only if the mirror profile was parabolic or hyperbolic, because these are the only possibilities for the existence of a virtual perspective point as shown by Baker and Nayar [10]. Note that even in a pure cylindrical projection, the horizontal elements of the corner will not always be horizontal in this image, but it is desired that the corner tracker be as accurate as possible, and that 100 percent performance is not required to still yield an accurate location estimation.

Square n by n sub-images within this warped pseudo-cylindrical view were convolved with an ideal corner of one of these eight types, an operation equal to finding the projection of the sub-image as a vector onto a $n \cdot n$ space as described by Li and Madhavan [115]. A mask size of 30 x 10 pixels was used in the experiments performed. This convolution is done over a r_{range} by θ_{range} in X and Y respectively, according to the maximum expected image flow of any landmark feature. These ranges are a function of the distance to, and speed of the mobile robot. The best variance normalized convolution response is chosen as the tracked corner location. its θ value is chosen and passed onto the triangulation stage.

Optic Flow methods such as finding the minimum of an SSD surface found by correlating

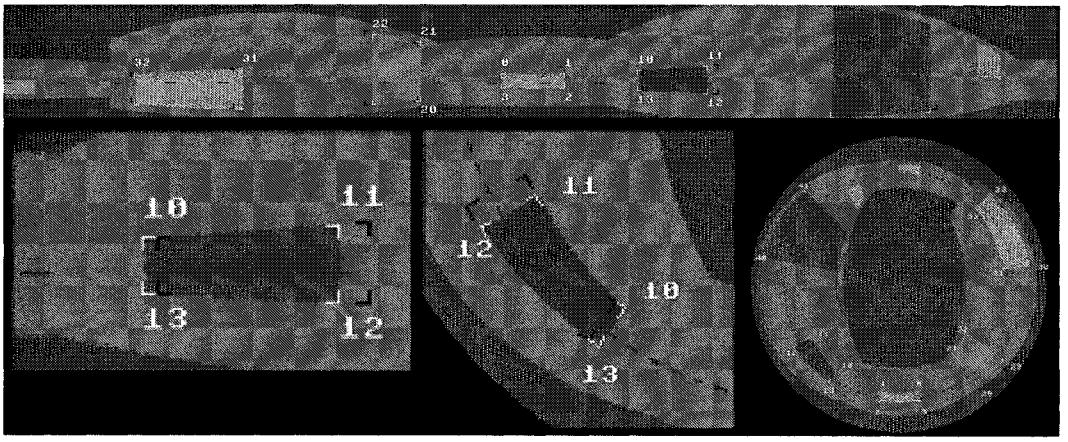


Figure 9.5: *(Bottom Right) Synthetic panoramic image. (Top) Quasi-cylindrical image warped from the panoramic image. (Bottom left) Close-up of quasi-cylindrical image. Black brackets represent the original predicted location of the corner. White brackets show the new landmark location after corner tracking. (Bottom middle) Close-up of annotated panoramic image of same vertices. The numbers annotate the corners according to their assumed match in the scene model.*

image fragments were also attempted, comparing image fragments around the landmark location between frames. However the location of the corner within this window would drift over the duration of the sequence because of the accumulation of matching and round-off error.

The quasi-cylindrical image was created with a width determined from the circumference of the horizon radius rad_h(in pixels). A rough expected estimate of the error angle can be made from the localization error, $Error_{det}$ pixels, of the corner detector in the warped image by $\theta_{error} = \frac{Error_{det}}{2\pi rad_h}$ (for small angles).

θ_{error} can be used to estimate the region of uncertainty in the 2D coordinates. The camera's tangential position at a distance of $RANGE$ is indeterminate to about $Error_{tan} = 2\pi RANGE\theta_{error} = RANGE\frac{Error_{det}}{rad_h}$. If we assume two corner landmarks at 90° angle then the best expected error would be a region of uncertainty with a diagonal width of:

$$Error_{dist} = \sqrt{2}Error_{tan} = \sqrt{2}RANGE\frac{Error_{det}}{rad_h} \tag{9.3}$$

Experiments were performed using synthetic and real image sequences, an example of a section of the quasi-cylindrical view with its tracked corners is shown in Fig. 9.5 (top). The original image, with the tracked corners annotated is shown for demonstration purposes in Fig. 9.5 (bottom right).

The two image sequences and views of the scenes that created them, along with sequences of the quasi-cylindrical warped images and plots of the camera position trajectory can be

viewed online at [1].

9.6 Tracking Line Junction Landmarks

The previous method served as an introduction to the main method described in this section, and shows what can be done without using the *Panoramic Hough Transform.*

Defining a corner landmark as the junction between two successfully tracked line segments decreases the possibility of false landmark detection. Rather than examining a small image segment (30 x 10 pixels in the above experiments) for corner-like properties, tracking the projection of line edges of landmark objects will use image information spanning a much larger area. The probability of a false positive of a corner detection is the much reduced probability of the simultaneous failure of three conditions: two component line edges of the same orientation being both falsely recognized and both meeting at the expected landmark corner position. The use of many more pixels in determining the image coordinates increases the potential accuracy, sub-pixel accuracy becomes viable.

Projections of expected line segments forming a corner are predicted according to the current position estimate, and these are then tracked. However, detecting straight line edges is more challenging in panoramic imagery, especially if the catadioptric sensor does not have a single virtual perspective point. The model of a perspective pinhole projection and the benefits of an affine transform cannot be used in panoramic catadioptric imagery, unless as Baker and Nayar [10] have shown, the mirror has a parabolic or hyperbolic profile. In these two specific cases, light rays are captured whose direction all converge at a virtual perspective point. With all other mirror profiles this is not the case and the direction of captured light rays have no such convergence point, and the optical system is said to be *Non-Single Viewpoint* (non-SVP).

Parabolic and hyperbolic profiles have a single viewpoint and allow the creation of virtual perspective projection views and hence feature extraction and tracking can reduce to methods used in the large body of work directed towards conventional image sensors. An example of what can be done with the geometry of a parabolic profile for line detection can be found in the work of Daniilidis [47].

Many other mirror profiles, however, are desirable for several reasons, an example being circular (spherical mirrors) for the ease of their manufacture. SVP parabolic systems utilizing panoramic mirrors require the additional cost and complexity of a telecentric lens to capture the parallel rays, motivating the consideration of other profiles. Other useful panoramic mirror profiles are designed to shape the density of image resolution as a func-

[1] http://www.cs.ualberta.ca/~fiala

tion of elevation, either to evenly distribute the image resolution throughout a desired range [30], or to improve resolution at certain elevations [55]. Derrien [32] demonstrated various advantages to relaxing the SVP constraint for panoramic catadioptric system design.

As an alternative to using a mirror profile that guarantees a virtual perspective point, or designing a profile that best approximates an SVP, one can depart entirely from the SVP restriction. Non-SVP profiles could still be used if straight line features can still be recognized, for the stated problem of mobile robot navigation in man-made environments, they are perhaps the most important feature type. The *Panoramic Hough Transform* models the projection of horizontal straight lines in non-SVP situations, and can be used as a replacement to the pinhole camera paradigm.

Assuming the orientation of the catadioptric sensor having the main axis vertical, horizontal lines can be found with the PHT. Vertical line detection is easier since a vertical line projects to a straight radial line. Together these two basic primitives, forming most indoor man-made environments, can be tracked to find landmark corners more robustly and accurately than corner detection alone.

9.6.1 Using the Panoramic Hough Transform

The PHT is applied to landmark tracking as follows. Given a set of polygon corner landmarks deemed visible from the current mobile robot's position, the horizontal and vertical line segments are converted to estimated projection parameters, and these parameters are updated if this line segment is indeed found. A corner landmark is then declared if both segments are found and meet within a threshold error from one another. The horizontal line projection is defined by four parameters: $(R_{main}, \theta_{main}, d\theta_{min}, d\theta_{max})$. The vertical line projection is defined by three parameters $(R_{min}, R_{max}, \theta)$. The radial parameters (R_{main}, R_{min} and R_{max}) are calculated from $R = E(D, Z)$, the radial mapping function given in the derivation of the PHT (Chapter 2). An original set of parameters for each segment is predicted, then recognized by the tracking operation, and the parameters are updated if the segment is recognized in the image.

A set of image plane points are found for each line segment projection. This was accomplished by taking a series of linear samples at 90° to the predicted line direction. The linear samples are independently analyzed to identify the the possible location of the largest step edge greater than a set threshold. With vertical line projections the samples are taken in the tangential direction, and for horizontal line projections they are taken radially. In both cases, extra samples are taken beyond the predicted end points to accommodate movement.

In the case of vertical lines, the new θ value is calculated by approximating the mode by a truncated median of the angular positions of the step points whose step positions form a

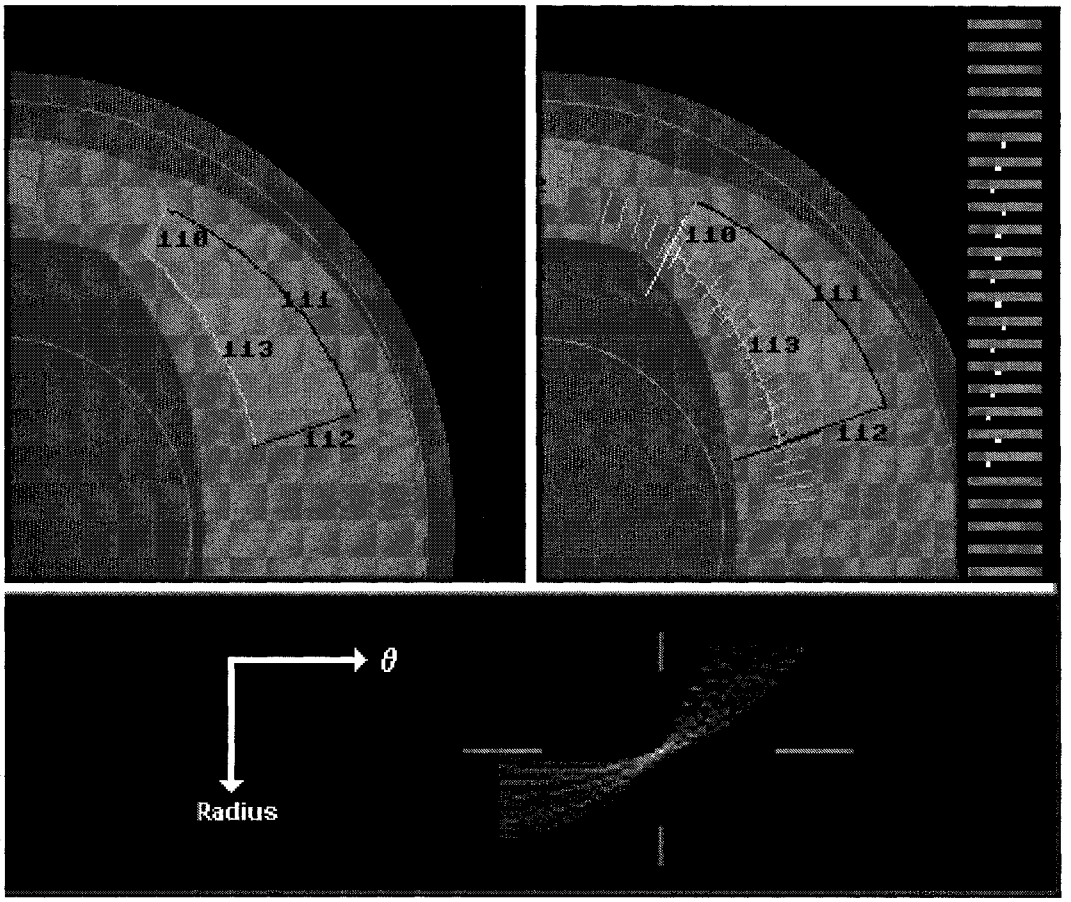


Figure 9.6: *A (Top Left): The predicted projections of landmark line segments. B (Top Right): The tracking process shown for horizontal line segment projection* 113. *Linear samples are taken (redrawn horizontally to the right). A linear step edge detector reports the presence and location of a step edge of the correct polarity (shown as a white dot beneath its corresponding linear sample). The PH-transform of all these points are plotted in C (bottom), and the new line parameters R_{main} and θ_{main} found. The new line segment is redrawn in B, and the new endpoints shown as black and white radial lines.*

contiguous line. For the horizontal line segment, the polar coordinates of the detected step edges are all plotted on the PH-Transform space. If a peak is detected, the line segment's R_{main}, θ_{main} is updated. For both horizontal and vertical line segments the endpoints are updated after the line definition is found, by iteratively taking a linear sample half-way between the last sample that found a step, and the one that did not. This process is shown below in Fig. 9.6 for a section of an image from the synthetic image sequence.

After the tracking processes confirm and fine-tune the location of the horizontal and vertical line segment projections, the position and orientation of the camera can be updated. The θ_{main} parameter is collected for each successful intersection of a horizontal and vertical line, and the robot's camera position extracted using Eqns 9.1 and 9.2.

9.7 Test Data Sets: Experimental Sequences

Two sets of image sequences were used to test the two localization methods. They can be found along with the experimental results at [2].

A synthetically generated sequence of 66 frames, each 500 x 500 pixels, was generated with ray-tracing methods and a simulated catadioptric image sensor. The scene consisted of uniformly shaded rectangular polygons along the path with walls behind to block other parts of the path. The rendering was performed with the *Povray* ray-tracing package. The Povray model file was created automatically from an FVE scene model for each position according to a preset trajectory file.

A preset trajectory file gave the (X, Y, Z, θ) position for each frame number, and was used to compare against the extracted trajectories from the two landmark navigation methods. The Z value was held constant to simulate the motion of a mobile robot on a horizontal plane.

The real image experiment consisted of 57 images captured by a panoramic sensor in steps 5 cm apart. The environment was approximately 200 cm x 200 cm, and was composed of distinct light and dark rectangular panels which were measured to create the *a priori* model (shown Fig. 9.8). The panoramic catadioptric sensor was composed of a SONY 999 NTSC video camera with a focal length of 590 pixels, and a spherical mirror of radius 4.9 cm, with its center located 15.1 cm from the camera's focal point.

Pre-processing was performed on the images (before warping) consisting of averaging with a 3 x 3 filter, and adjusting the aspect ratio (Fig. 9.9). The three supports were blended away to prevent the vertex and vertical line tracking routines from settling falsely on these occluding supports. This blending was done by linearly interpolating the intensity

[2] http://www.cs.ualberta.ca/~fiala

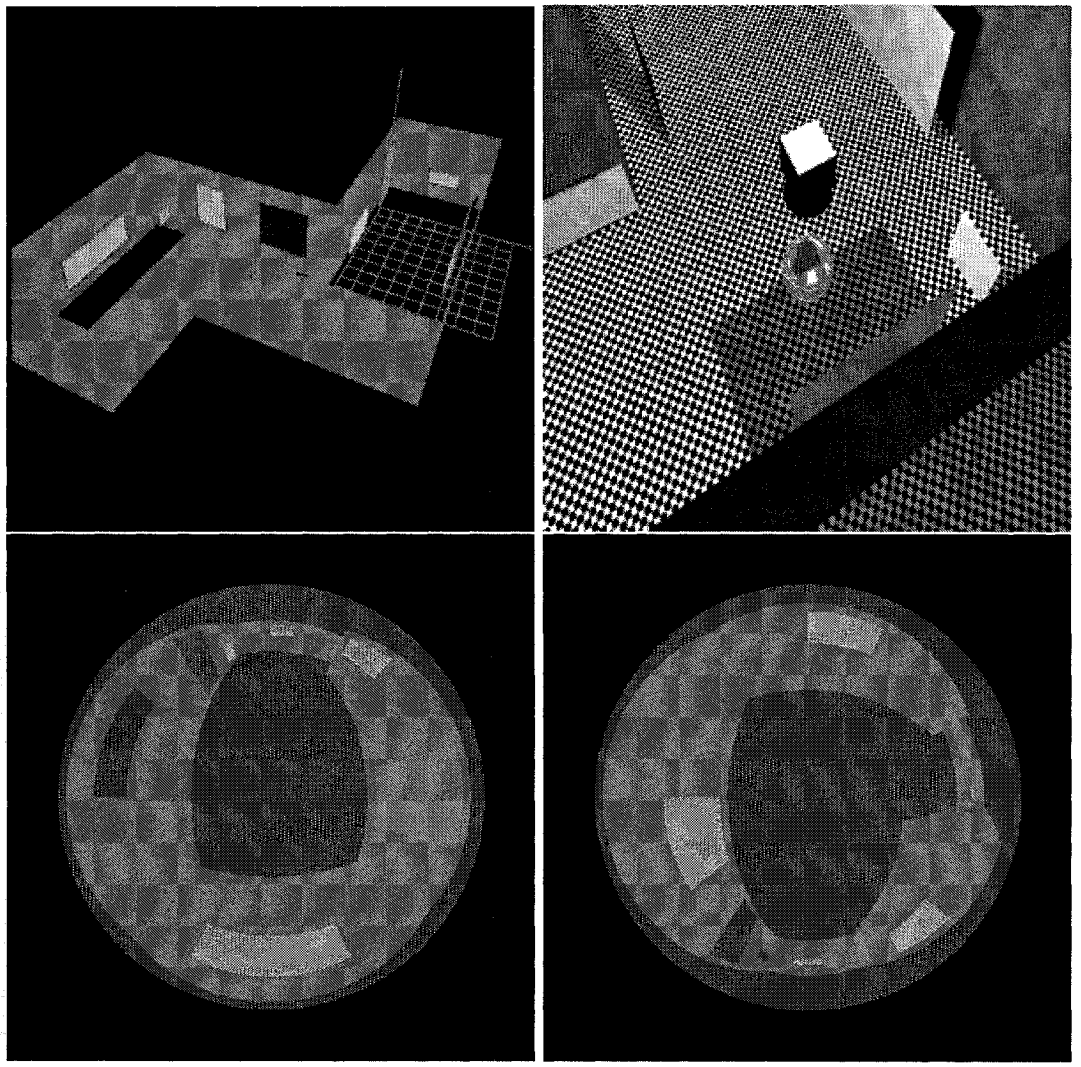


Figure 9.7: *Synthetic image data set: 66 images. The synthetic environment is shown in the FVE viewer (top left) and a top view from* Povray *(top right). This data set is used for to test both tracking methods: vertex and line junction. Ray-tracing is used to create a series of images from points along an S-shaped path through an environment of differently shaded uniform vertical rectangles. (Bottom left,right) Two frames from the sequence.*

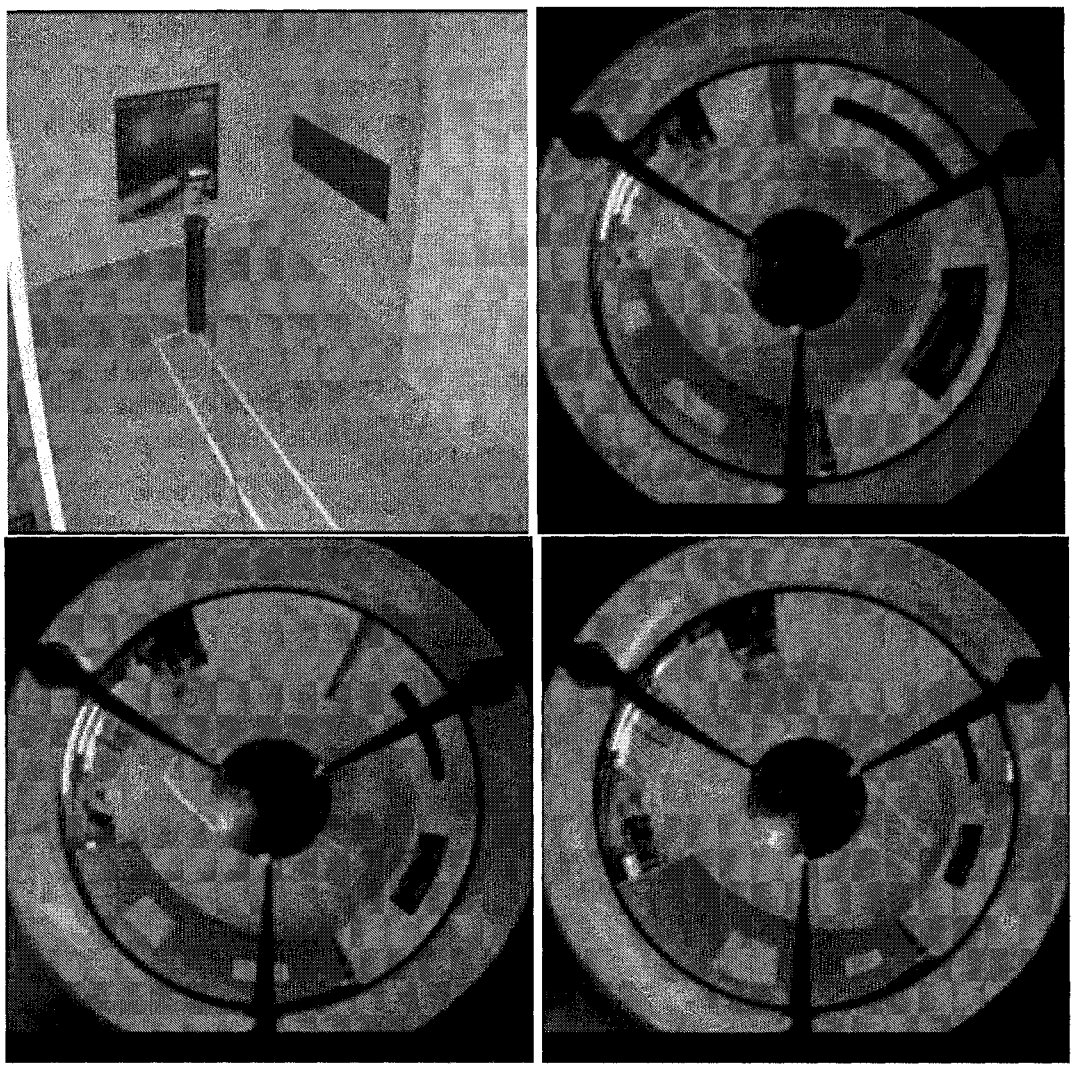


Figure 9.8: *Real image data set: 57 images. This data set is used for both the two tracking methods: vertex and line junction. The panoramic sensor is moved along a U-shaped path, with images taken at measured locations 5 cm apart. (Top left) Setup environment with 4 vertical rectangle polygon objects. (Top right) First image from sequence. (Bottom left) 15th image from sequence. (Bottom right) 35th image from sequence.*

between two azimuth angles along each radial line in the occluding region.

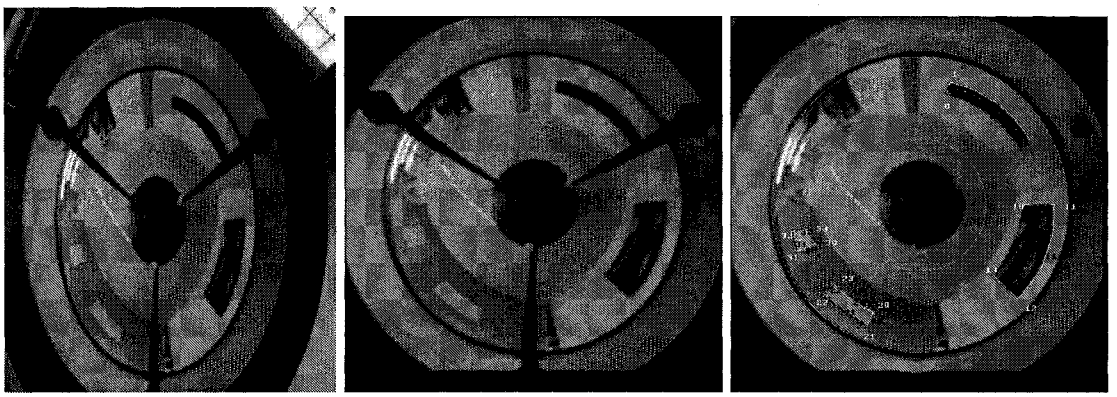


Figure 9.9: *Real image pre-processing. (Left) Original frame-grabbed NTSC image. (Middle) Image corrected for aspect ratio and smoothed by 3x3 averaging filter. (Right) Supports are blended away so as not to interfere with tracking routines.*

9.8 Vertex Tracking Results

9.8.1 Synthetic Images

The position tracking system progressed through all images without losing track, although the synthetic sequence had to be modified to add an extra polygon on the first turn to repair an area that didn't have enough landmarks. The estimated position would diverge and the system would be thereafter "lost" in the absence of at least two correct landmark measurements with a sizeable vergence angle. This change in the model was reflected in the sequence above and the synthetic sequence runs worked thereafter.

The position and orientation were successfully extracted for the 66 points, the error was divided into distance (from the correct location for that frame) in Table 9.1 and the orientation angle error Table 9.2. The distance to objects was typically about 70 units, and the error in position had a standard deviation of 0.5 units, a typical error of 0.7 percent. 3% of the location readings were considered outliers with a maximum error of 1.98 units (3.5 percent). The angular heading error had a standard deviation of 0.36° with some (excluding outliers) that stretched to 0.72°.

With $rad_h = 157$ pixels, a corner detection error of $Error_{det} = 1$ pixel, and $RANGE = 70$ units, the rough error estimate (Eqn. 9.3) is 0.63, fitting with the standard deviation observed of 0.50 units.

Excluding the outliers, the error is as expected. When the diagnostic data was examined,

Z-Score (SD=0.50)	Number Points (of 66)	percent
0 - 0.5	27 points	40.91 %
0.5 - 1.0	29 points	43.94 %
1.0 - 1.5	6 points	9.09 %
1.5 - 2.0	2 points	3.03 %
2.0 - 2.5	0 points	0.00 %
2.5 +	2 points	3.03 %

Table 9.1: Distance Error: Vertex tracking with synthetic image sequence. There were 2 outliers, the maximum with a distance error of 1.98 units.

Z-Score ($SD = 0.36°$)	Number Points (of 66)	percent
0 - 0.5	50 points	75.76 %
0.5 - 1.0	8 points	12.12 %
1.0 - 1.5	3 points	4.55 %
1.5 - 2.0	2 points	3.03 %
2.0 - 2.5	0 points	0.00 %
2.5 +	3 points	4.55 %

Table 9.2: Angle Error: Vertex tracking with synthetic image sequence. There were 3 angular measures considered outliers.

the outliers were found to be a result of failure of the corner detector.

9.8.2 Real Images

The position was successfully extracted for the 57 images, shown in Table 9.3 the angular error was not measured, since the observed heading angle was within the error of the sensor angle (it was difficult to ascertain the angle to better than 3° when moving and placing the sensor).

Z-Score (SD=5.06 cm)	Number Points (of 57)	percent
0 - 0.25	2 points	3.51 %
0.25 - 0.5	11 points	19.30 %
0.5 - 1.0	24 points	42.11 %
1.0 - 1.5	16 points	28.07 %
1.5 - 2.0	6 points	10.53 %
2.0 - 2.5	0 points	0.00 %
2.5 +	3 points	5.26 %

Table 9.3: Distance Error: Vertex tracking with real image sequence. There were 3 outlier points as the trajectory temporarily diverged.

The standard deviation of the error distribution was 5 cm, corresponding to an error of 3 percent over the average distance of 150 cm (from camera to landmark). This would be in accordance with a landmark azimuth error of about $Error_{det} = 5$ pixels, or 0.5 degrees of error. Because of the simple and well spaced corners, only one clearly incorrect outlier position occurred when several of the corner detectors failed at once. The corner detectors

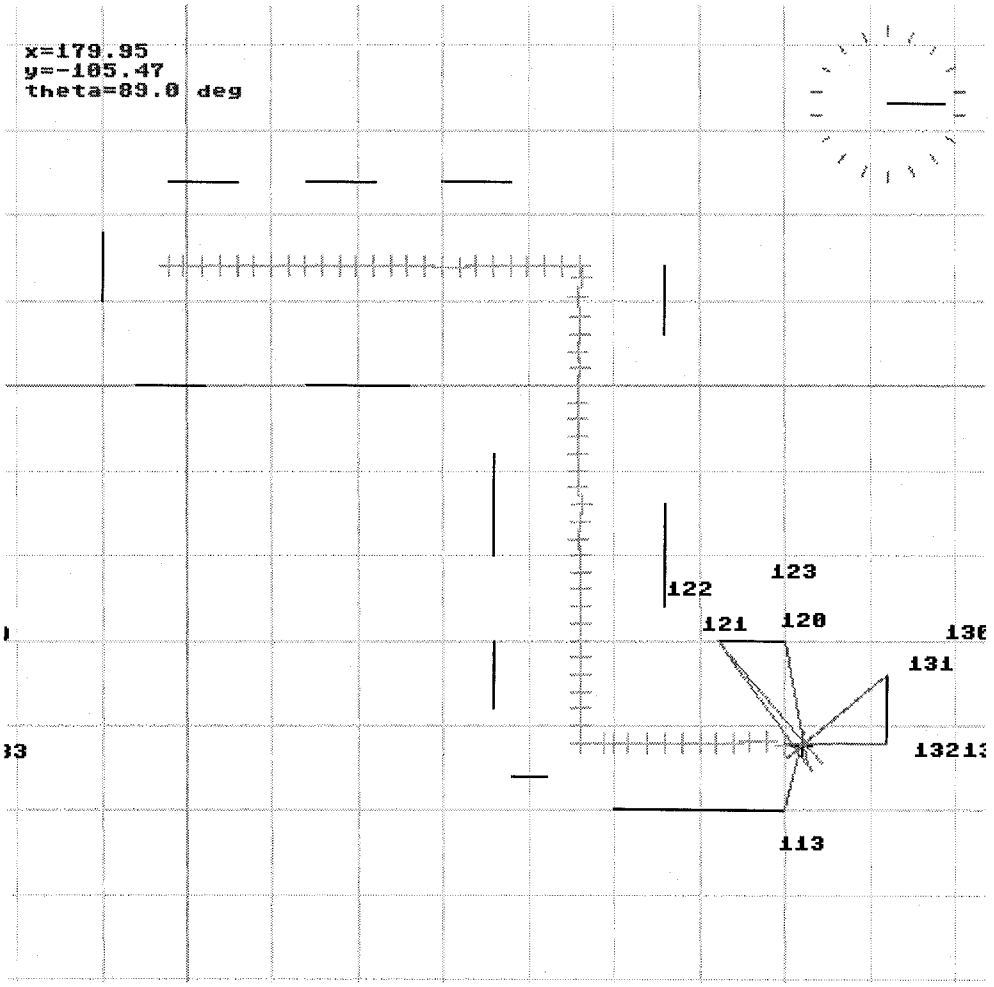


Figure 9.10: *Recovered Trajectory with vertex tracking: Synthetic image sequence.*

often drifted several pixels from the correct corner, and at times settled incorrectly on a corner-like feature. The effects of this were mitigated by the averaging of several landmarks.

A trade off exists between motion accommodated and the possibility of outliers. If the search window is too large, the corner detector may incorrectly locate the corner on another corner-like feature. Likewise if the search window is too small, then the corner will not be located at all. the possibility of finding the wrong corner is greater. This is a manifestation of the aperture problem encountered in optic flow studies, and it was clear that the method would not be as robust in more cluttered scenes. This motivated the need to examine more of the image in determining landmark location.

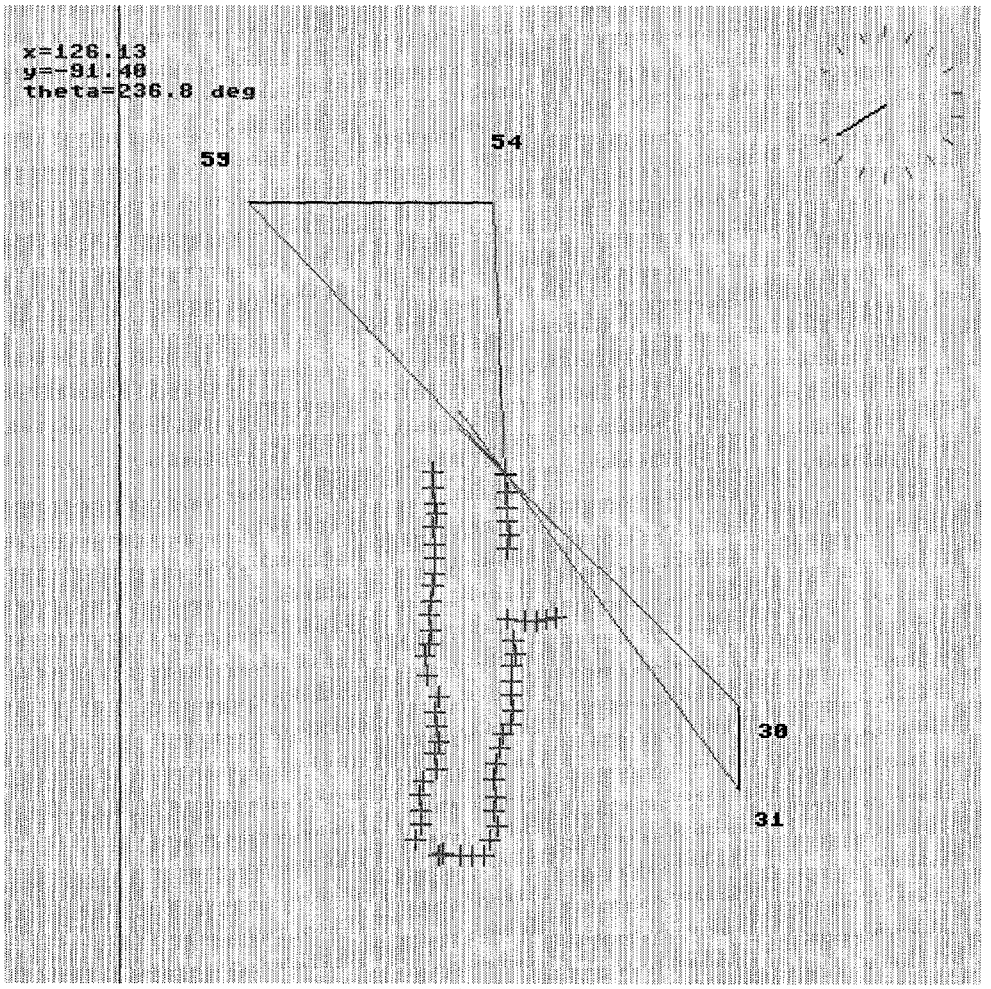


Figure 9.11: *Recovered Trajectory with vertex tracking: Real image sequence.*

A sample image, with the tracked corners annotated onto the image, and a section of the quasi-cylindrical view, is shown in Fig. 9.12.

Fig. 9.13 shows an error caused by occlusion of the supports. Feature vertex 28 lies inside the occluded area and so it is not properly tracked. This error was not propagated

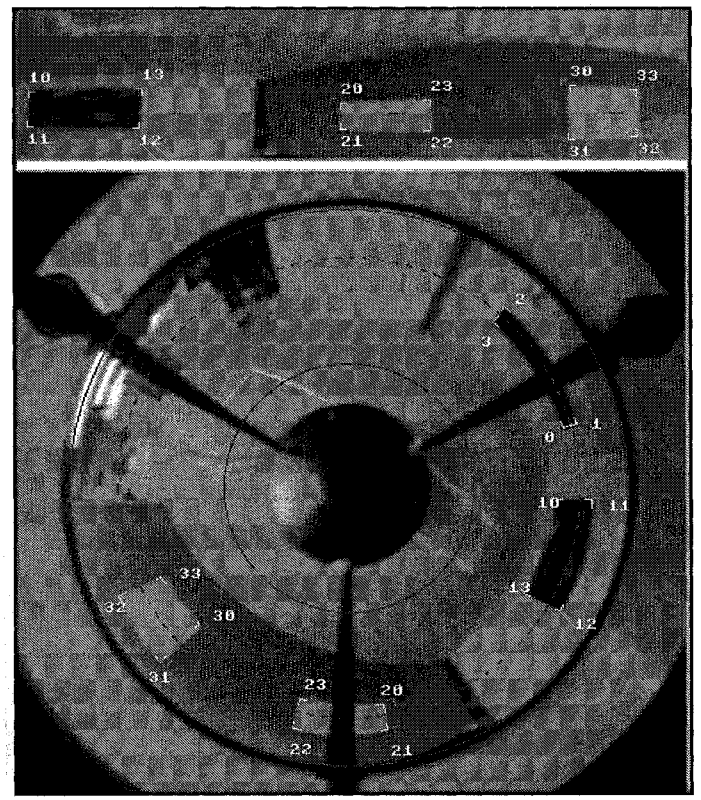


Figure 9.12: *Real image vertex tracking. A: (top) Section of the quasi-cylindrical image created from the panoramic image B: (bottom).*

by a heuristic of ignoring landmark headings inside the known occluding regions.

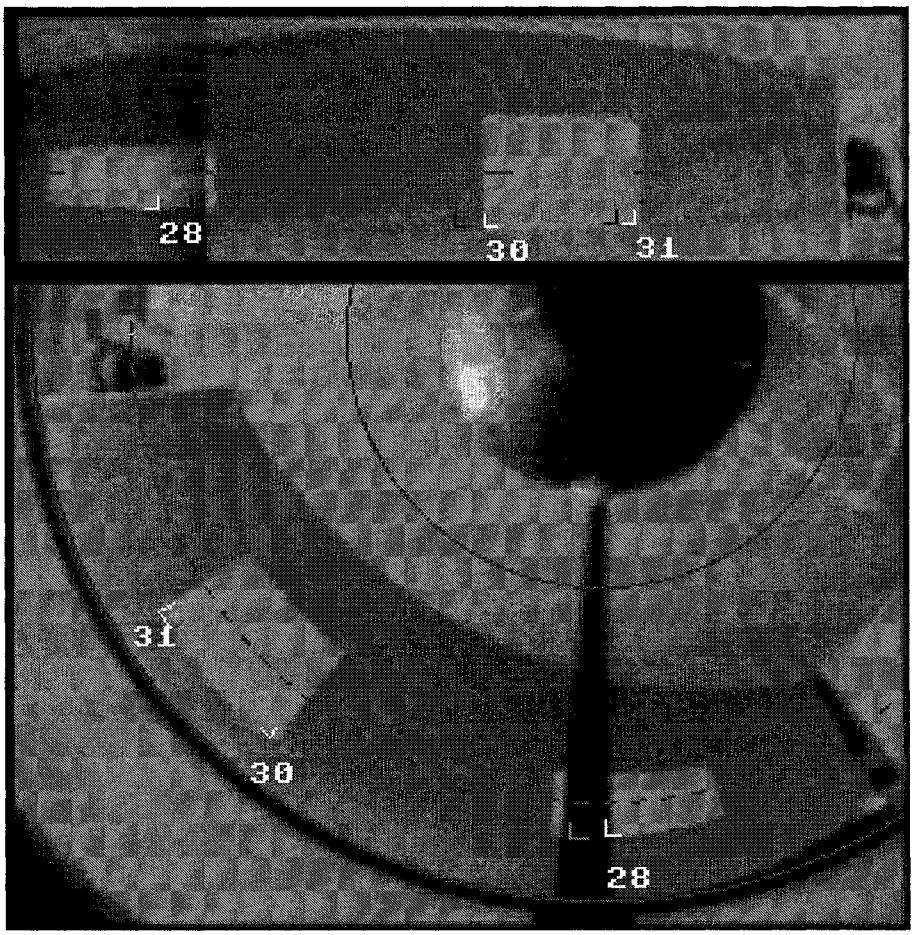


Figure 9.13: *(Top) Close-up of quasi-cylindrical image warp showing an error with the landmark behind the (blended) occluding support. (Bottom) Close-up panoramic image of same area. In both images, the black brackets show the predicted location and the white brackets show the new location found by the corner detector.*

9.9 Noise Test: High Landmark-Feature Match Outlier Rate

To evaluate the robustness of the system, the visibility filter was turned off, providing the system with many landmarks that weren't in the image. This was then run with the synthetic image sequence.

With the corner detector seeded by an erroneous vertex, it would either find no corner and safely report no heading, or settle on some feature close to the seeded point, providing an incorrect landmark-feature correspondence. The percentage of false features that could find matches would increase with the scene clutter and search window size.

The simulated outlier noise, and the performance was observed (Table 9.4). Note that the statistics were calculated on all the points, without first rejecting outliers, so the standard deviation was 52.09 units, and most of distribution falls within Z=0.15 which is not consistent with a normal distribution. However it serves to demonstrate the relationship. The system managed to keep roughly on the path until the end where it was thrown off as shown in Fig. 9.14. With the robot implementation (later section), a heuristic would analyze for the "lost" condition, and disregard completely wrong position estimates.

Z-Score (SD=52.09)	Number Points (of 66)	percent
0 - 0.05	12 points	18.18 %
0.05 - 0.10	16 points	24.24 %
0.10 - 0.15	18 points	27.27 %
0.15 - 0.20	2 points	3.03 %
0.20 - 0.25	4 points	6.06 %
0.20 - 0.50	2 points	3.03 %
0.5 - 1.0	0 points	0.00 %
1.0 - 1.5	3 points	4.55 %
1.5 - 2.0	4 points	6.06 %
2.0 - 2.5	0 points	0.00 %
2.5 +	5 points	7.58 %

Table 9.4: Distance Error: Vertex tracking with synthetic image sequence. There were 12 outliers as the system went entirely off track, the maximum with a max distance error=162.3 units.

Z-Score ($SD = 0.36°$)	Number Points (of 66)	percent
0 - 0.5	46 points	69.70 %
0.5 - 1.0	8 points	12.12 %
1.0 - 1.5	9 points	13.64 %
1.5 - 2.0	2 points	3.03 %
2.0 - 2.5	0 points	0.00 %
2.5 +	1 points	1.52 %

Table 9.5: Angle Error: Vertex tracking with synthetic image sequence. The angular performance didn't seem to suffer as much as the distance when the system lost track. There was 1 angular measure considered an outlier.

9.10 Tracking Line Junction Experimental Results

9.10.1 Synthetic Images

Experiments were performed with the same real and synthetic images sequences as the vertex tracking method. However, this time the projections of the horizontal and vertical edge segments of the scene polygons were used instead of vertices as landmarks. The edge

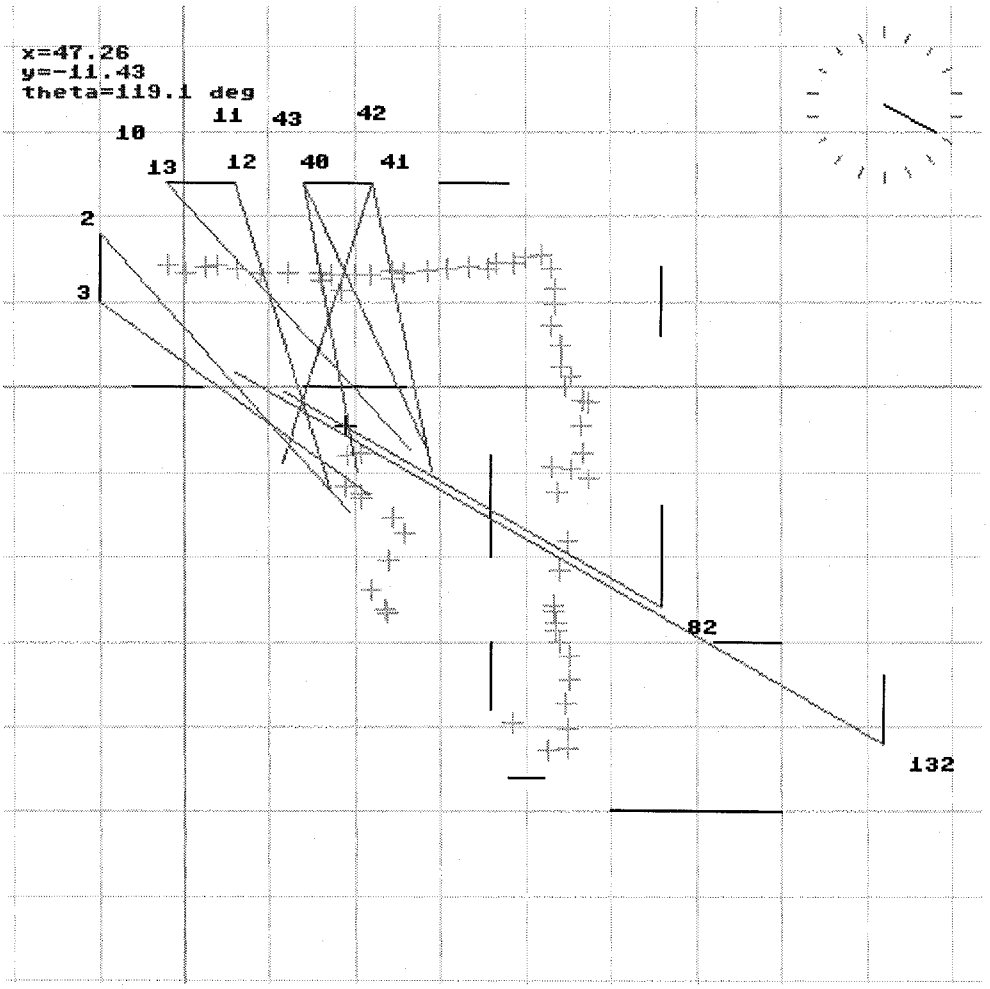


Figure 9.14: *Noise test, vertex tracking with many outliers: Synthetic image sequence.*

segments were tracked using tracking routines based on the PHT. The vertex was still used as a landmark, however it was determined from tracking the segment edges in the image rather than the projection of the corner itself. The tracking for a frame in the synthetic sequence is shown in Fig. 9.15, including a sample tracking of a horizontal edge projection feature.

Applying the junction tracking method to the synthetically generated sequence successfully recovered the camera trajectory, with a standard deviation of 0.4 units, a typical error of 0.5 percent over the average landmark range of 70 units. No outliers were found, and the results are shown in Tables. 9.6,9.7. Repeating the error estimate calculation from Eqn. 9.3, this indicates just less than sub-pixel accuracy on corner location detection. More important than the improved position accuracy is the absence of outlier points.

Z-Score (SD=0.41)	Number Points (of 66)	percent
0 - 0.25	1 points	1.52 %
0.25 - 0.5	12 points	18.18 %
0.5 - 1.0	29 points	43.94 %
1.0 - 1.5	22 points	33.33 %
1.5 - 2.0	3 points	4.55 %
2.0 - 2.5	0 points	0.00 %
2.5 +	0 points	0.00 %

Table 9.6: Distance Error: Line Junction tracking with synthetic image sequence. There were no points considered outliers, the maximum error was 0.97 units.

Z-Score ($SD = 0.36°$)	Number Points (of 66)	percent
0 - 0.5	29 points	43.94 %
0.5 - 1.0	18 points	27.27 %
1.0 - 1.5	14 points	21.21 %
1.5 - 2.0	3 points	4.55 %
2.0 - 2.5	0 points	0.00 %
2.5 +	2 points	3.03 %

Table 9.7: Angle Error: Line Junction tracking with synthetic image sequence. There were no points considered outliers,

9.10.2 Real Images

Tests with the real image sequence produced a slightly degraded distance error standard deviation of distance error of 5.25 cm corresponding to $Error_{det} = 3$ pixels (0.3 degrees of error), but with the improvement of no outliers.

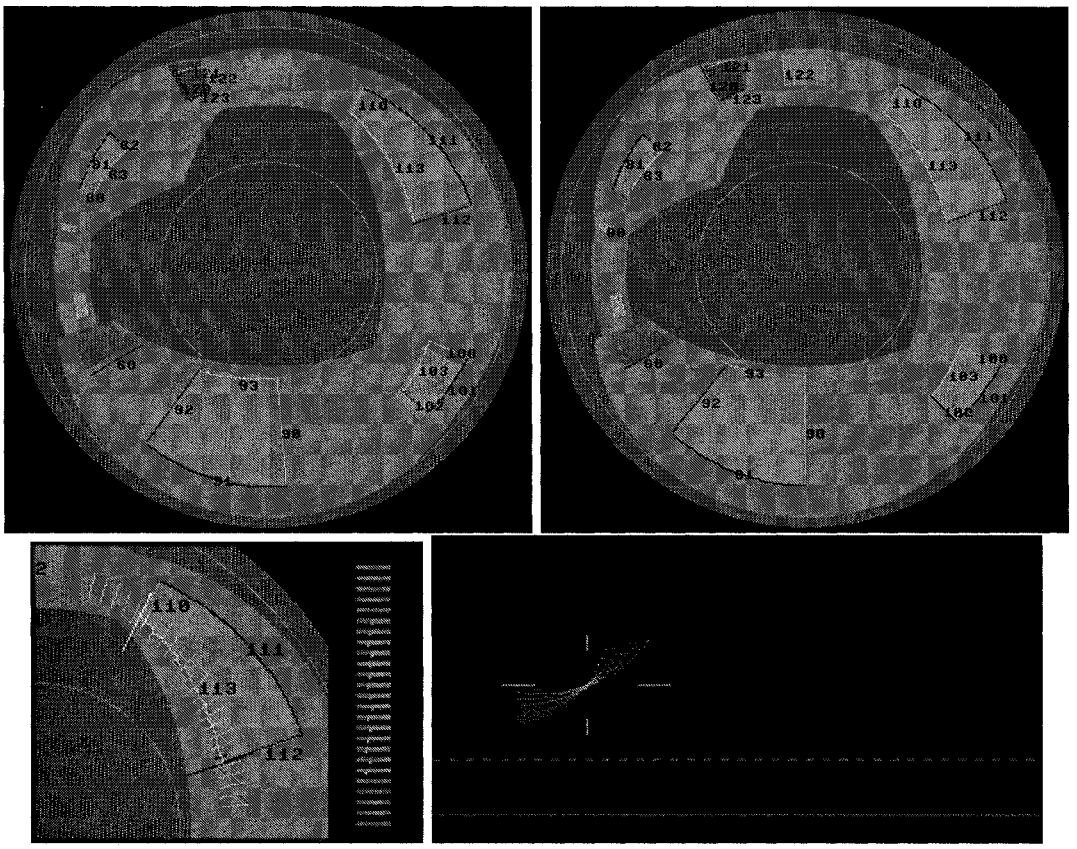


Figure 9.15: *(Top Left) Predicted feature projections overlaid over input image frame. (Top Right) Tracked feature projections overlaid over input image frame. (Bottom Left) Close-up of tracking horizontal feature No. 113 with locations of radial linear samples taken shown in white. (Bottom Middle) Linear samples shown, with results of linear edge processing performed underneath. The white dots report where an edge of the correct polarity and sufficient steepness was detected. (Bottom Right) PHT of the edge locations in the linear samples, finding a new definition for a horizontal line segment projection.*

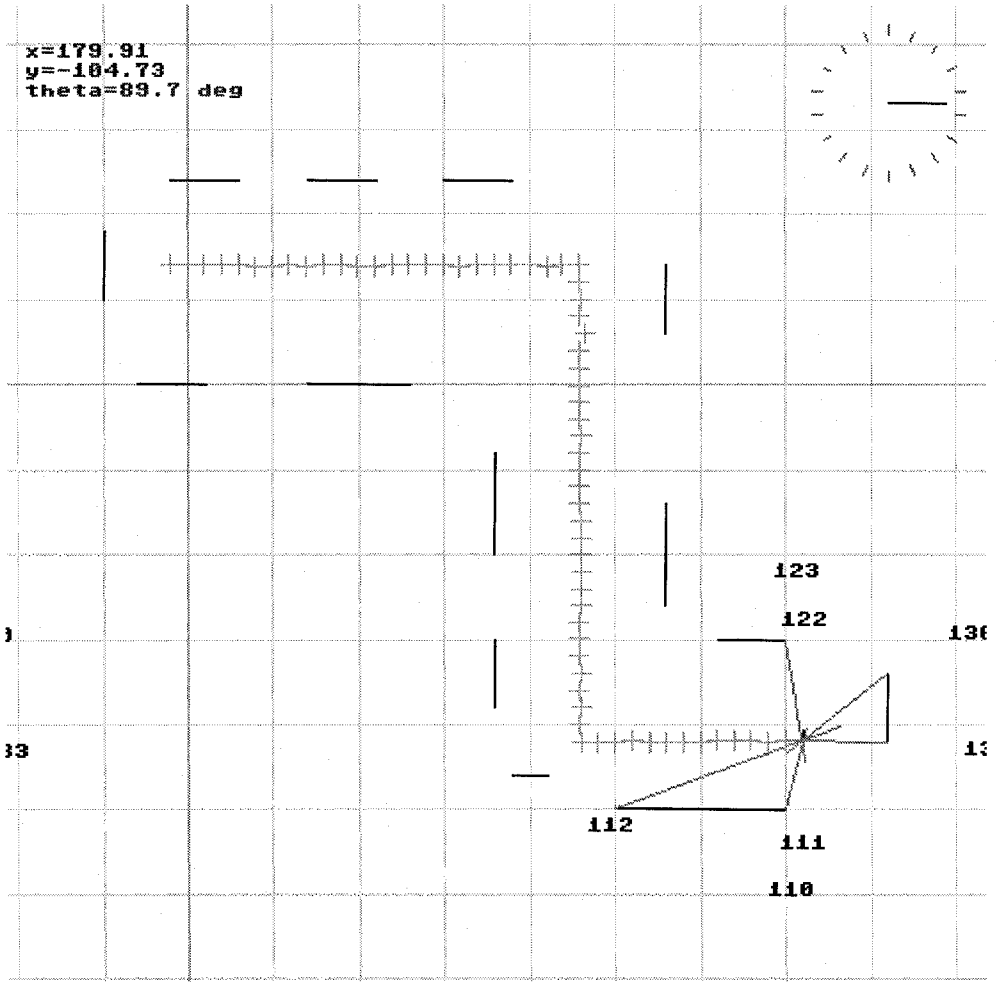


Figure 9.16: *Recovered Trajectory with line junction tracking: Synthetic image sequence.*

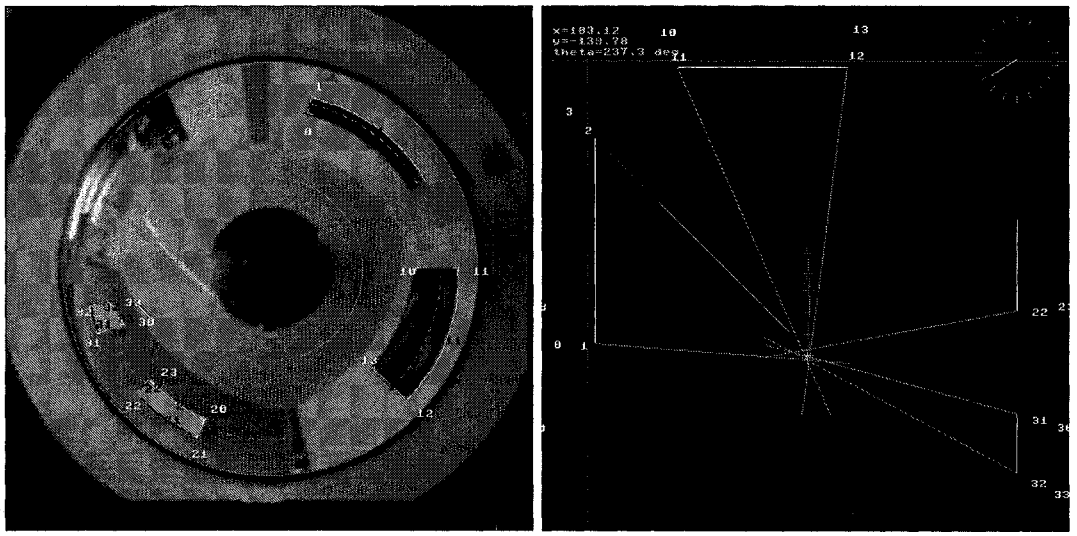


Figure 9.17: *(Left) Post-tracked image 10 (of 65) of the real image sequence. Manually positioned panoramic sensor. (Right) The recovered trajectory using line junction tracking at frame 44 of the synthetic sequence.*

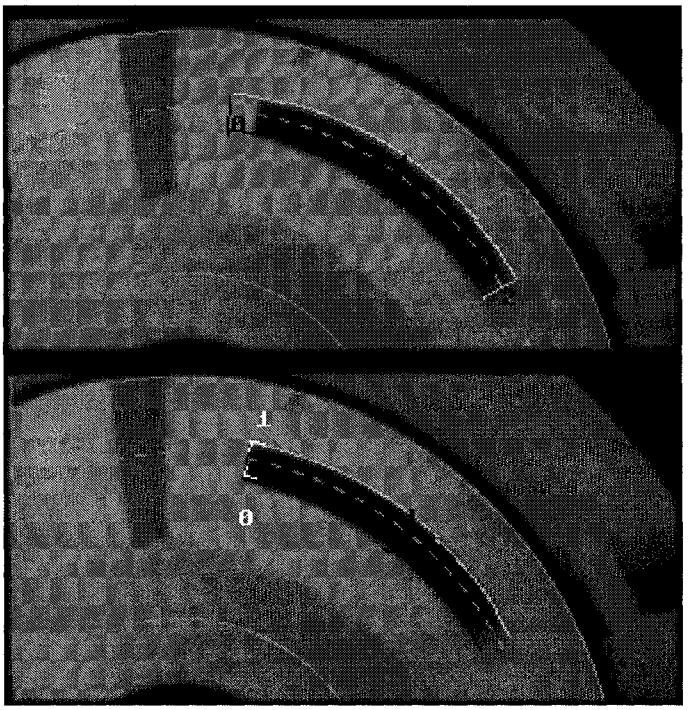


Figure 9.18: *Tracking of scene polygons in real imagery. Top shows pre-processed image overlaid with predicted segment projection features. Bottom image shows features after tracking. Notice bottom left edge of polygon is missing since it fell within the area blocked by the support (smoothed over during pre-processing). The segments are tracked in the image in this method (annotated with black numbers), not the vertices. The vertices are identified from the close endpoint locations of the two intersecting edges. Since the lower right edge is not found in the tracking step, only 2 of the possible 4 vertex landmarks are reported to the localization stage.*

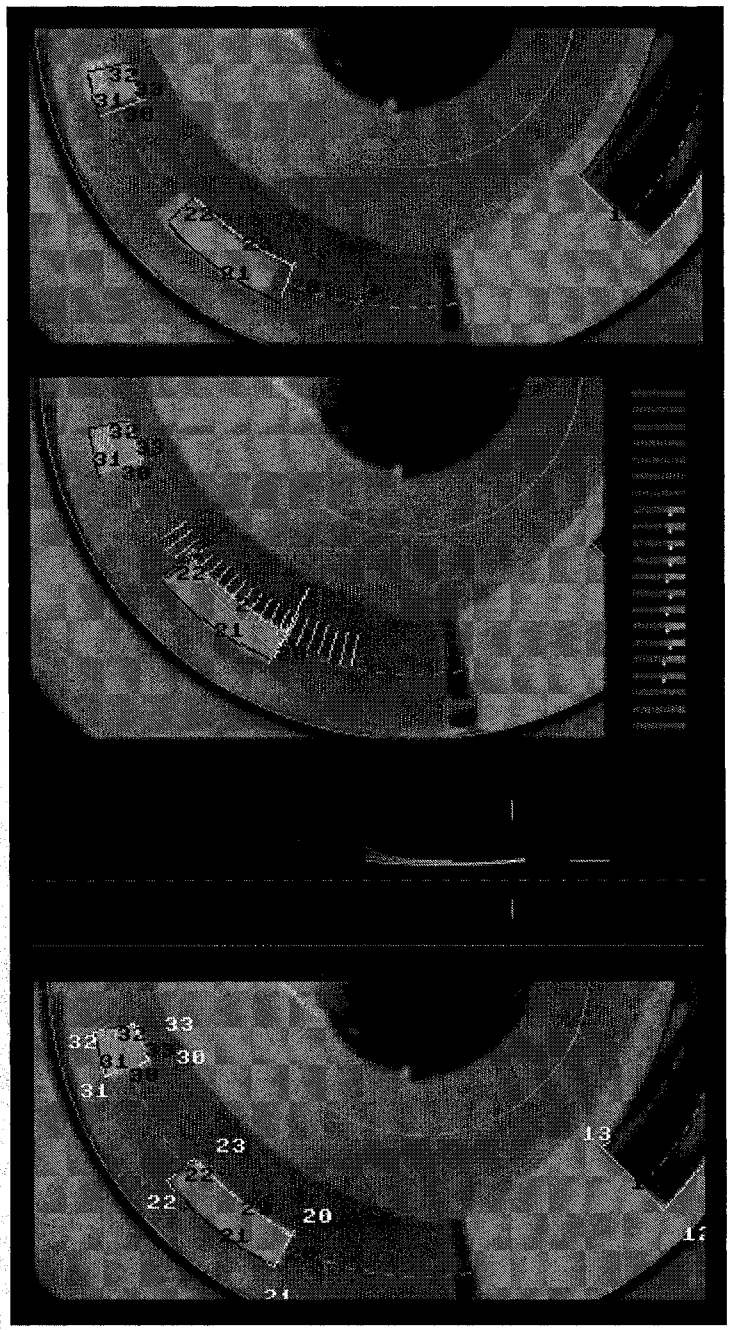


Figure 9.19: *Tracking of scene polygons in real imagery - further detail shown for the tracking of edge number No.23. (Top) Input pre-processed image with predicted feature segments overlaid. (2nd from top) Linear slices taken from image, shown at right. Slices (redrawn horizontally at right) are processed with one dimensional step localization routines, the located step position is drawn as a white dot under the slice. Only the angle and radius of this step position is sent to the PHT (3rd from top) Panoramic Hough Transform (PHT) image of projected step points. Automatically detected cluster peak, corresponding line is redrawn in 2nd image. Source image is examined to find endpoints of the located line. (Bottom image) Tracked feature segments redrawn on input image.*

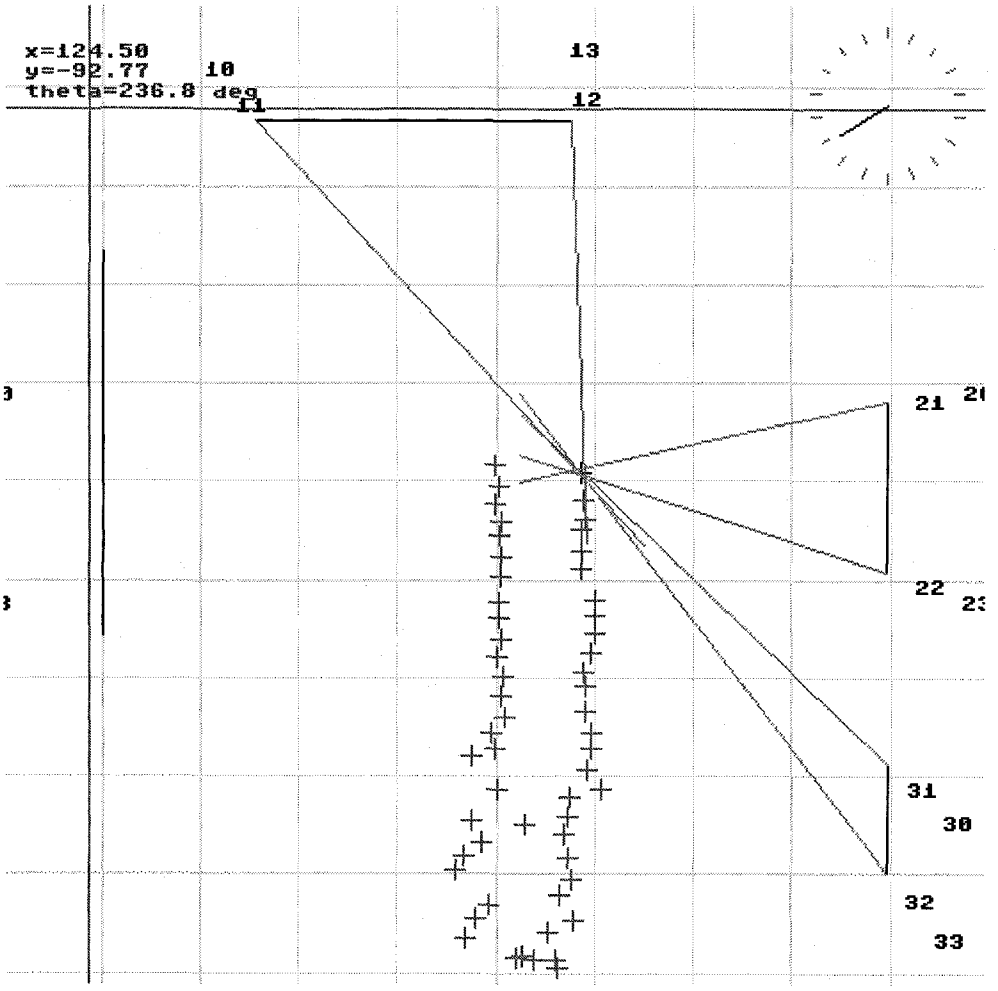


Figure 9.20: *Recovered Trajectory with line junction tracking: Real image sequence.*

Z-Score (SD=5.25 cm)	Number Points (of 57)	percent
0 - 0.25	1 points	1.75 %
0.25 - 0.5	12 points	21.05 %
0.5 - 1.0	29 points	50.88 %
1.0 - 1.5	13 points	22.81 %
1.5 - 2.0	2 points	3.51 %
2.0 - 2.5	0 points	0.00 %
2.5 +	0 points	0.00 %

Table 9.8: Distance Error: Line Junction tracking with real image sequence. There were no outliers, and the maximum distance error was 10.9 cm.

9.11 Discussion of Experiments

The experiments above show a reasonably reconstructed trajectory for both the simulated and real image sequences for both methods. The aggregated statistics for the synthetic and real sequences are shown in Tables. 9.9 and 9.10 respectively.

Tracking Method	Standard Deviation (units)	Outliers
Vertex	0.50	2
Line Junction	0.41	0

Table 9.9: Synthetic image sequence: comparison of distance error for both tracking types.

Tracking Method	Standard Deviation (cm)	Outliers
Vertex	5.06	3
Line Junction	5.25	0

Table 9.10: Real image sequence: comparison of distance error for both tracking types.

The outlier rate is more significant for assessing the performance of a localization system, for this represents when the system is "lost". Sometimes the system regains a position lock shortly thereafter, but just as often it stays lost as it keeps chaotically jumping around trying to match landmarks with the last incorrect position. The simulations were stopped when this occurred, otherwise the outlier rate would become very large. This position loss was observed much more often in the vertex tracking approach.

With the details of a given mobile robot, some thresholds for reasonably expected maximum motion between frames could be incorporated to filter the output of the localization system. This was done in the robot system shown below, the position estimate was not updated when this "lost" alarm was raised. The system has a better chance of relocating the position from the last known good reading than from the mostly random position that this failure would produce. Two additional recovery strategies could be to increase the search range for image features, and to get the robot to reverse its most recent movement to bring it back to a location where it could maintain position tracking.

In general, the more landmarks available, the greater the chance of a successful location. If one assumes a random model of outlier landmark-feature data points, then they should average out and the correct data points should mostly define the position. Section 9.9 above demonstrated how the trajectory could be maintained even with a large number of the landmark bearings in error. More than half of the bearings were false, but yet the trajectory was still correct enough to give a functional position to a robot navigation system (the randomness would need to be smoothed out with some type of filtering, such as Kalman filtering). The purpose of that example was to show that landmark tracking in general can be reasonably useful, even with lots of incorrect data. The system typically gets lost when there is not enough landmarks in sight, and is then much more susceptible to error.

Looking at the statistics for the real image sequence experiments, the distribution of distance error is not centered at Z=0 which leads to a suspicion that there may have been errors in the creation of the scene model, an erroneous measurement in even one landmark when creating the reference model would be sufficient to produce this. Also, note the deviation to the left in the extracted trajectories about two-thirds the way down the plots. This also leads to a potential error in scene model creation, and the performance would probably be better if the experiments were repeated.

The experiments above only used two sequences, and not a complete exhaustive test. The numerical results do not fully characterize the performance in the opinion of this author. The implementation with the robot described below is more meaningful, the qualitative observations of the system in action on the robot was more convincing of the whole approach.

Watching the robot's estimated position on the computer screen, and occasionally stopping it to measure the X,Y location (the heading was more difficult to measure in the field) gave an impression of general robustness. The lines projecting back from the landmarks almost always converged tightly at a point. As well as the main lab tests, the system was set up in several rooms for presentations, with hastily set up and measured landmark polygons, and the system did function correctly.

9.12 Prototype System

A proof-of-concept prototype system using the line junction tracking code was built with a panoramic camera mounted on a mobile robot. A frame rate of about 1.3 hz was achieved, which allowed for reliable tracking with slow movements of the robot. The reliability prognosis of the tracking was qualitative, determined by monitoring the performance. The robot and a sample screen shot of the robot's extracted position are shown in Fig. 9.21. The slow frame rate was mostly due to technical issues in the frame grabber and image display, the

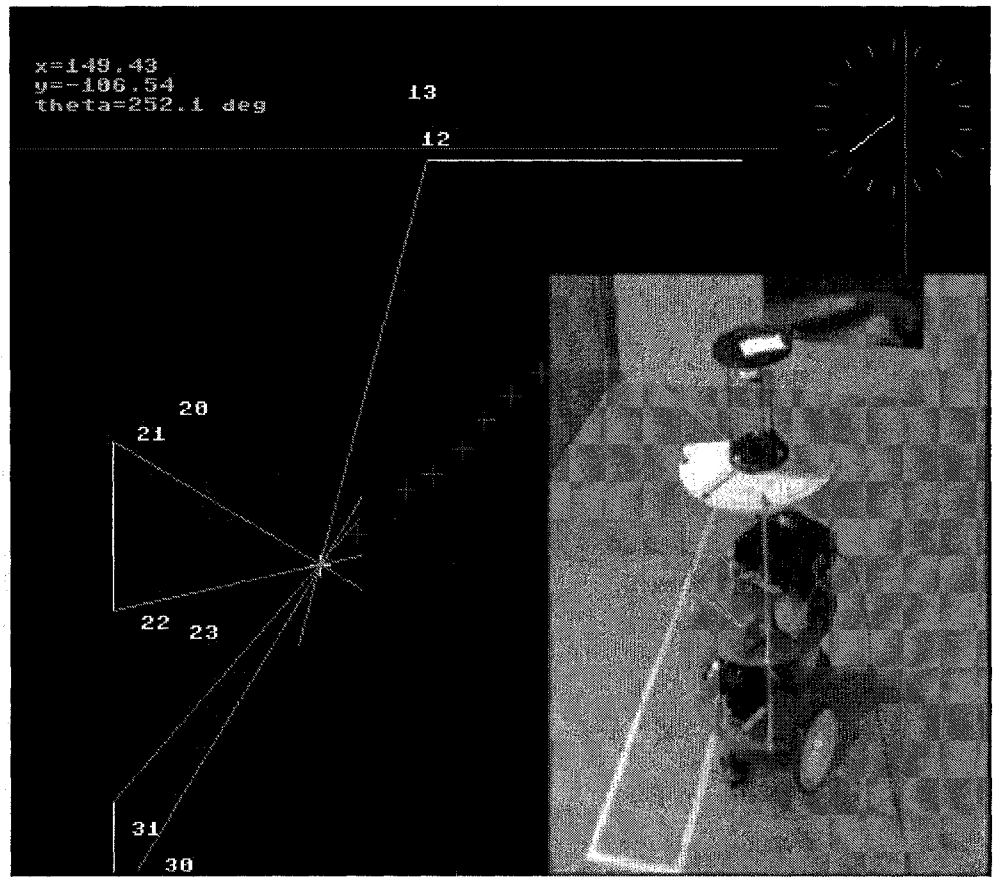


Figure 9.21: *Screen shot of prototype system showing triangulation of junctions in Fig. 9.17. The robot's trajectory is drawn by crosses indicating the position at previous frames. In this case the robot is translating towards the lower left.*

PHT calculation was done with lookup table and performed rapidly. Two consequences of the slow position update rate was that the radial and angular search ranges had to be large, and the robot motion kept slow, especially for rotations. A quick lurch would cause the landmark features to move further than the tracking search range and the position could not be found. With a dedicated real time 30 hz implementation, robust operation of the positioning system is estimated for speeds perhaps greater than 1 metre per second.

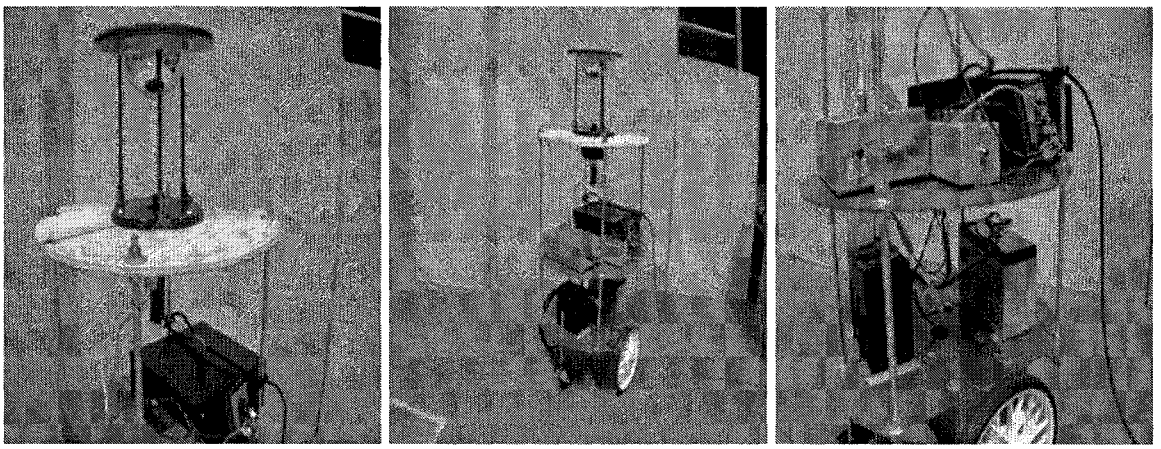


Figure 9.22: *Three views of the robot. The single-lobed spherical profile catadioptric panoramic sensor is mounted on the top of a two wheeled system. An RF modem, motor control systems and batteries make up the rest of the mobile platform. All processing is done back on a remote PC. The robot has no odometry and so all positioning is entirely visual.*

9.13 Conclusions

Two methods of providing position and orientation information for mobile robot navigation with a panoramic camera were presented, and results of synthetic and real experiments reported. The methods tracked the projections of environment landmarks predicted according to the robot camera's position. Both tracked vertices of polygons, but differed in how the vertex was found. The first method used a modified corner tracking procedure in a quasi-cylindrical view, and the second method tracked line segments and determined corners from the junction of two line segments. The latter was found to be more robust and immune to outliers. Both methods would perform best with as many landmarks as possible. This was found in the vertex tracking experiment where the correct location of some landmarks would reduce the error from false corner measurements.

A prototype mobile robot system was successfully demonstrated having a catadioptric image sensor with a spherical, non-SVP mirror profile.

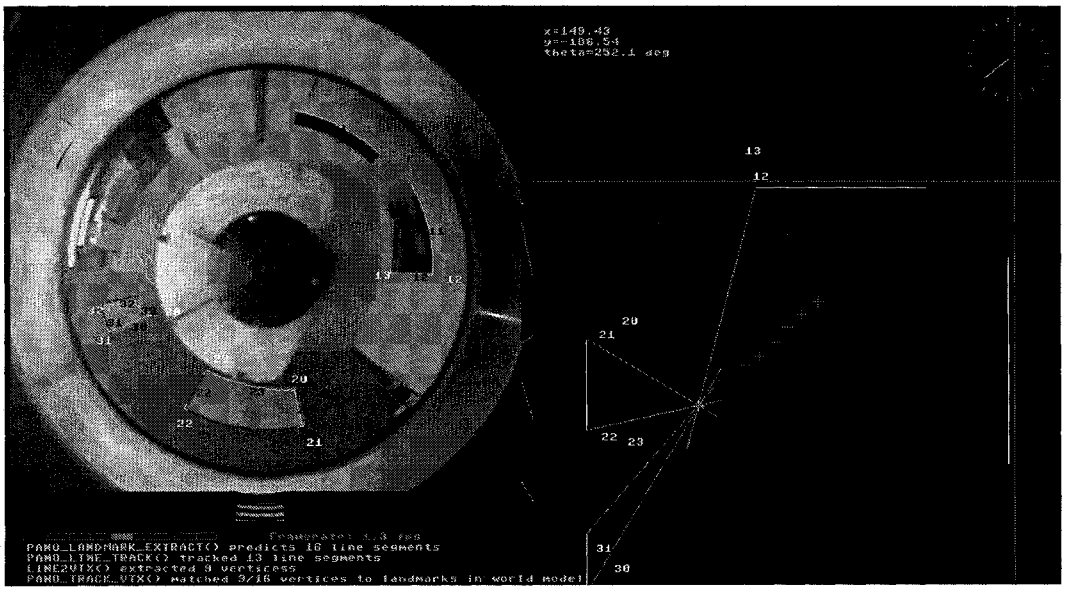


Figure 9.23: *Screen capture of X window on PC. The linux based system operates a frame grabber custom designed by the author), processes the images and relays control actions to the robot via the RF serial link.*

The more novel contribution to mobile robotic navigation lies in the second method where the application of the *Panoramic Hough Transform* allows tracking of straight line segments in catadioptric panoramic cameras free from the SVP mirror profile restriction.

9.14 Future Work

The system implemented in the robot used the PHT to track the projection of horizontal lines, and more conventional means to track vertical segments projections. An *a priori* map is required from which to predict the location of segments. The work of this thesis comprised both tracking already known segments with feature extraction techniques to find new ones.

The robot system could be extended to build its own map by using tracking on areas of the image associated with known areas, and feature extraction in the regions corresponding to unknown space. Segment projections from extracted polygons or corners could be tracked as the robot moves, and their existence verified and location established as the robot saw the feature from various angles. This would require map building and motion planning work, but the computer vision components for such a system have been created in this thesis.

Chapter 10

Structure from Motion using Panoramic Hough Transform Techniques

The *Panoramic Hough Transform* is the main theoretical tool used in previous chapters dealing with stereo reconstruction and robot navigation. These applications used the PHT within one image frame to find the projection of horizontal lines in imagery from panoramic catadioptric cameras with mirrors of a non-Single Viewpoint (non-SVP) profile.

The PHT is extended to spatio-temporal feature extraction in the three-dimensional space of an image sequence, since scene points trace such a path as they travel along a horizontal line trajectory relative to the camera. The method is applied to reconstruct a three-dimensional model from a sequence of panoramic images, where the panoramic camera was translating in a straight line horizontal path. Only the camera/mirror geometry is known *a priori*. The camera positions and the world model is determined, up to a scale factor.

The 3D model created is not a description of polyhedral objects faces bounded by horizontal and vertical lines only as in Chapter 8, but a 3D cloud of points/polygons of any shape. Three dimensional reconstruction can be performed by either extracting features which span many pixels within one image or by following points that span many images in a sequence. The latter would produce a 3D mesh based upon correlation between images and is commonly referred to as *shape through motion*.

This chapter introduces a shape through motion system using panoramic imagery and the PHT. Preliminary experimental results are shown with synthetic images.

10.1 Introduction

Shape from motion describes a computer vision approach to creating a model with three-dimensional information from a sequence of two-dimensional images. In the least restrictive paradigm, no knowledge of the scene, nor the camera motion is available. Researchers have demonstrated the possibilities, the classic example being the goal to reconstruct a model from a video clip captured by a moving video camera, as well as the camera's path [50, 5, 106].

Panoramic image sequences or real-time video can be produced for applications such as mobile robotics. Shape from motion model creation from such imagery would be useful for such robots for exploration and navigation, or for making scene models for such applications as immersive telepresence. The Panoramic Hough Transform for detecting lines and line segments in non-SVP panoramic images is extended to that of tracking the motion of points through a panoramic image sequence to allow a three-dimensional model to be created.

If an image sequence is captured while the camera is moving in a straight direction along a horizontal plane, scene points will move in a horizontal straight line trajectory relative to

the camera. Image points are tracked with *Optic flow* techniques, and this relative motion recognized with this line detection theory. This motion can be used to calculate distance to the point.

10.2 Shape from motion

Shape from motion relies on similarity between the different image frames, the camera is assumed to have a continuous trajectory and the viewpoint between subsequent images should not change drastically. This can be exploited by recognizing image features between frames that are from the same scene point. Optic flow techniques can allow some image points to be tracked between images. With enough such points and knowledge of the image formation (typically a perspective projection) one can solve for both the positions of the objects, and the relative camera motion between the frames. This requires robust recognition and tracking of features, and the ability to discern between stationary scene points and the possibility of moving objects.

Researchers such as Hartley [54] have used the *fundamental matrix* approach to attempt to recover scene geometry from image sequences. The model that is created is necessarily limited to only being computed accurately to a scale factor, if the scene objects and the motion were equally scaled, an identical image sequence could be generated. The model created can be in the form of a 3D point set, polygon mesh or list of lines or other primitives. This model could be colored with the intensity seen in the imagery for use in creating views from new angles for telepresence and virtual reality applications. Alternatively, as is the thrust of this thesis, it could be used by an autonomous robot to build up an obstacle, terrain model of the environment that it is passing through.

Panoramic catadioptric structure-from-motion theory based on the projection of lines in space to circles on the image plane with SVP sensors has been developed by Daniilidis [50]. However no work has been done on shape-through-motion with non-SVP catadioptric systems.

10.3 Panoramic Hough Transform Applied to Panoramic Shape From Motion

The *Panoramic Hough Transform* (Chapter 2) was developed for this purpose, and can be used to recognize horizontal lines when the panoramic camera described is posed so that its central axis is vertical. This variant of the Hough transform was created to provide a way

to automatically recognize the curved lines on the image plane that a horizontal line feature would map to. Similar to the classic Hough transform, clusters formed in a parameter space indicate the presence of a possible projection of a horizontal straight line. This method can be used for feature extraction within a single image, indeed the case can be made that the detection of horizontal and vertical lines can account for the majority of line features in man-made environments. The method is here extended to follow the straight horizontal trajectory a scene point makes relative to a panoramic camera. The camera is restrained to horizontal motion with some sections of its trajectory following a straight heading. This restriction is reasonable for mobile robots traveling on man-made horizontal surfaces, and furthermore this theory allows automatic detection of when the condition of straight travel is met.

Scene points which are successfully followed by optic flow tracking can be processed to find the presence of horizontal line motion and the 3D position and motion relative to the camera reported. This method considers each scene point separately, and no restriction exists on the shape of the scene objects.

10.4 Optic Flow

Optic flow [60, 102, 1, 5] deals with moving images, sequences of consecutive images and intends to determine which parts of an image are moving where, assigning a vector to each pixel, this result called a "velocity field". Singh [102] differentiates between "image flow" and "optic flow" where image flow is the current projections of object velocities onto the imaging plane, i.e. the ground truth, whereas optic flow is what can be obtained by just looking at the imaging plane itself and noting where each pixel appears to go. The goal of optic flow methods are to obtain a measure of optic flow that is as close as possible to the image flow.

Optic flow recovery techniques first attempt to extract info for each pixel by looking at this pixel's change over time and its relationship to neighboring pixels. If one uses the correlation method to find where a pixel has moved (mentioned below), this info is of varying precision and so usually a confidence or variance (sometimes in 2D) is provided. Optic flow is calculated to provide trajectories for processing by the PHT.

A short exploration of the field of optic flow is given in the next few sections, three different approaches are outlined, and ways to express this motion of image features are described. The reader is directed to Singh's comprehensive book [102] for more on the subject of optic flow. Much of the terminology in the next few sections is from this source. Aloimonos [4] gives a general overview of vision from moving images, and belongs to the

body of vision researchers that believe vision is best performed on moving sequences and not static images. Another good source is the work of Fermuller, which provides an in depth exploration of how 3D translation and rotation maps to image flow [39, 106].

The optic flow literature differentiates between *conservation* information which is found by looking at small pixel neighbourhoods and is calculated first, and *neighbourhood* information which assumes structure on a larger scale and is used to estimate the flow for image locations where unique conservation information is available.

10.5 Optic Flow:Calculating Conservation (Locally available) information

There are three main methods to calculate the locally available optic flow estimate (Singh calls this first step *conservation information* because it is the result of some equation which conserves some quantity). They are: 1-Gradient based approaches, 2-Correlation based approaches and 3-Spatio-temporal energy based approaches. All methods assume a constant image that has just translated over time on the imaging plane, and are followed by some iterative procedure to propagate velocity information of a high confidence to regions of lower confidence.

The three methods are briefly introduced below.

10.5.1 Approach 1: Gradient Approach

The gradient approach [56] is based on the equation $I_t = -(I_x u + I_y v)$ where (u, v) is the the optic flow, and (I_t, I_x, I_y) are the partial derivatives of image intensity with respect to time, x and y respectively. (I_x, I_y) could be the result of a Prewitt or Sobel mask, or just the subtraction of a pixel's image intensity from that of the pixel to the left or above. (I_t) is the change of intensity over time at a given pixel location, and can just be the difference between the intensity of the pixel between the current and previous frame. This can only recover the perpendicular flow if there is an edge, nothing is there is a corner or uniform region. The "normal flow" is of magnitude $\frac{I_t}{\sqrt{I_x^2+I_y^2}}$ in the direction of the fastest increasing intensity given by the unit vector $[\frac{Ix}{\sqrt{I_x^2+I_y^2}}, \frac{I_y}{\sqrt{I_x^2+I_y^2}}]^T$.

A 1D version of image flow would be the observation that if we had an image line translating, at a pixel, the rate of change of intensity would be the negative of the rate of change of the intensity spatially multiplied by the speed of translation. $I_t = -I_x u$. In 2D, we have a spatial gradient in both X and Y and so the observed change in intensity is a

linear combination of some motion in the X direction and some motion in the Y. So we know $I_t = -(I_x u + I_y v)$ and so we have 1 equation, 2 unknowns. If we make new basis vectors called n and m, one along the direction of maximum image intensity (spatially) and one perpendicular to that, then $I_t = -(I_t u_n + 0 v_m)$ where (u_n, v_m) are the projections of (u, v) on the new basis (n, m).

Note that since all the spatial intensity gradient is the direction of n and none in the m direction, and so the motion along the m direction is unknown. This is saying is that by looking at one point's spatial and temporal gradients, we can obtain the normal component only of optic flow if we are at an edge. This compares to knowing the full motion for distinct points such as corners, or observing no motion in regions of uniformity.

10.6 Approach 2: Correlation Approach

The correlation methods involve taking a section of an image (3x3 or 5x5 for example) and comparing it to a series of same sized windows in the previous or next image within a search window defined by the maximum assumed speed of objects in the scene and using some measure obtain a result for the quality of match for each spot in the search window. The result of the correlation search is an image (of search window size) with axis corresponding to the u,v displacements and intensity representing the "goodness" of match and therefore confidence that u,v offset is where that pixel moved. The correspondence measures mentioned in Singh are direct, mean-normalized, variance-normalized and Sum-of-Squared-Differences (SSD) correlation. These measures will be lowest for the best match and so Singh inverts them with $R = exp[-k \cdot SSD]$ to make them a larger number for a better match, another researcher Scott [99] uses $R = k1/(k2 + k \cdot SSD)$. Both Singh and Scott's "inversion" of the error measure avoid the divide-by-zero problem. The correlation approach is used in this application due to the expected large movement of feature points between frames, i.e. the motion is not expected to be slow sub-pixel motion.

10.6.1 Approach 3: Spatio-Temporal Energy Approach

This involves gathering information of spatial and temporal frequencies of points and placing them in a three-dimensional space with axis (w_x, w_y, w_t) which are the spectral content of spatial and temporal info. If enough points can be found, they will form a plane in this 3D space from which the optic flow (u, v) can be derived.

10.6.2 Representation of the Conservation Information

Singh mentions mostly the correlation method in his book, and the correlation method (after "inverting" the error result to get a positive measure of good fit) provides an image (of search window size) from which can be estimated by an estimate and distribution information describing the precision of the estimate. The estimate is the mean formed by dividing the moments in the X and Y direction by the sum, and the distribution shape is represented by a covariance matrix.

The precision varies from none at all in the case of a uniform bit of image (impossible to tell if it is moving at all) to an edge where one can tell motion perpendicular to the edge but not parallel to it, and to regions like corners which can provide a unique flow vector. The result can be a mean velocity with a description of the distribution such as providing 1-a covariance matrix or 2-the location and length of the principle axis (the major following the long axis of an ellipse and the minor axis traversing the narrow axis of the hopefully elliptical distribution). Near an edge the distribution of points is expected to be elliptical 90° to the edge direction, in a region of a distinct 2D feature such as a corner, the points should be bunched and in regions of uniformity spread out everywhere. The principle axis and lengths can be determined by finding the eigenvectors and eigenvalues of the covariance matrix, with some special case procedures for the places where the covariance matrix is singular and non-invertible (getting divide by zero problems).

10.6.3 Neighbourhood Constraints

Now an iterative procedure must go through the vector field supplied by this "conservation information" (mean displacement u,v and co-variance matrix) and adjust them according to their neighbors to propagate info from one area to another. Schunk [56] uses a smoothing function, assuming the velocity field does not change greatly from pixel to pixel. This does work to a degree but blurs motion edges of objects. Nagel [79] uses the second spatial derivative of the image to inversely weight this smoothing process to not smooth the vector field around edges and corners as much. Singh however finds the neighborhood "opinion" of optic flow for that pixels by finding the average (u,v) of the average velocity fields of neighboring pixels as well as the co-variance of this info, and weights them by a spatial Gaussian in an algorithm to update the velocity estimate (the Gaussian gradually reduces the effect of more distant pixels' velocities).

10.6.4 Iterating to find the Optic Flow Image

Both the conservation information and this Gaussian-weighted neighborhood information are combined to provide the next iteration's estimate of optic flow for that pixel. The covariances of the conservation and neighborhood information are also used to utilize the confidence of these measures in the two directions (X,Y).

The above methods typically refer to optic flow between just two consecutive image frames. In Appendix A of Singh's book [102], he details a Kalman filter based approach to a long sequence of images to provide an updating image of optic flow where each pixel in the velocity (optic flow) field has a Kalman filter with the state vector as the optic flow itself.

10.6.5 Applications of Optic Flow

Optic flow is applied to 3D surface estimation (producing a depth map), image compression (MPEG for example describes movement of blocks of pixels) and video enhancement. Video enhancement uses optic flow to move the pixels around to accommodate motion and then averages them together. Averaging several sequential images in time reduces image noise but blurs motion in non-static parts of the image, and so optic flow is used to move sections of the previous and maybe subsequent images around to provide images to average without introducing too much motion blur.

10.7 Using the Panoramic Hough Transform for Horizontal Trajectories

A point in the image captured by the camera (referred to herein as the source image) can belong to a family of curved line projections of planes containing horizontal lines. This family of potential candidate planes can be described by with two degrees of freedom R_{main} and θ_{main} as detailed in Chapter 2.

Points in the source image Fig. 10.1A are defined in polar coordinates (R_i, θ_i), and points in the PHT space are in cartesian coordinates $(\theta_{main}, R_{main})$ which are the X and Y coordinates respectively in the PHT image shown (Fig. 10.1B). A set of transform points are found and plotted in the transform image for a source point (R_i, θ_i). This set is found by finding θ_{main} and R_{main} according to Eqns. 10.1 and 10.2. for all $d\theta$ values in the range $(-\pi/2 < d\theta < \pi/2)$ (in steps of a resolution for a desired accuracy in determining θ_{main}).

$$R_{main} = PH(R_i, d\theta) \tag{10.1}$$

$$\theta_{main} = \theta_i + d\theta \tag{10.2}$$

This is repeated for the next (R_i, θ_i) for all points in the original trajectory set, and a cluster peak will appear corresponding to the best fit θ_{main} and R_{main}. If these points don't well fit the projection of a horizontal line, the transform response will be spread out and it can be detected that this set of source image points is not from a trajectory of a horizontally translating scene point.

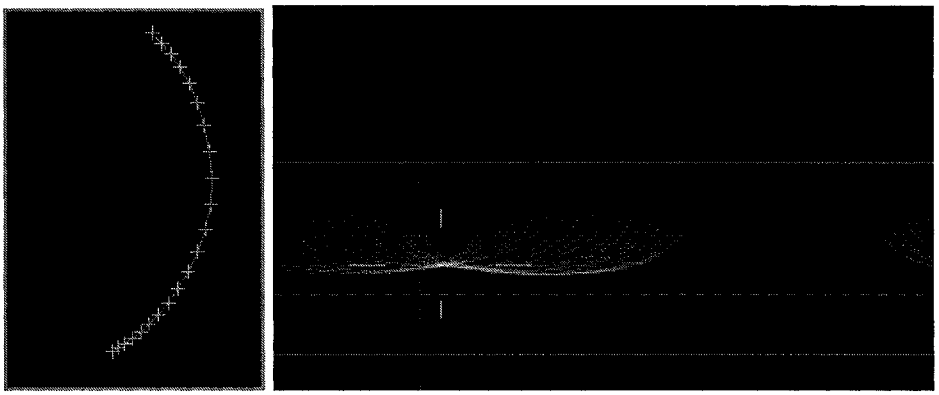


Figure 10.1: *A (left). A trajectory of points in the source image. Each point maps to a set of points in the PH hough image B (right). If a cluster is found in the PH image (shown with cross-hairs), then the trajectory of source image points can lie upon a horizontal line trajectory in 3D whose projection is defined by* R_{main} *and* θ_{main}

10.8 Calculating Optic Flow

The set of trajectory points are extracted from the panoramic image sequence of J frames using optic flow extraction methods between successive frames. Optic flow vectors and confidence values are extracted for each image pair of of $frame_j$ and $frame_{j+1}$. The experiments shown use a correlation based approach of a template N pixels on a side, in a search window $\pm\frac{M}{2}$ in both directions. A square image section of NxN pixels in $frame_j$ is compared to MxM similarly sized image sections $frame_{j+1}$. A variance normalized *Sum of Squared Differences* (SSD) measure is obtained for each search position. This value is lowest at a point of best fit, and a modified response is calculated from the SSD value to provide a number that is largest at the position of best fit, similar to Singh [102]. The modified response used is

$$SSD_{mod} = \frac{1}{SSD + diff_{av}}$$

(where $diff_{av}$ is the average of absolute differences between pixels at the match position where SSD was calculated).

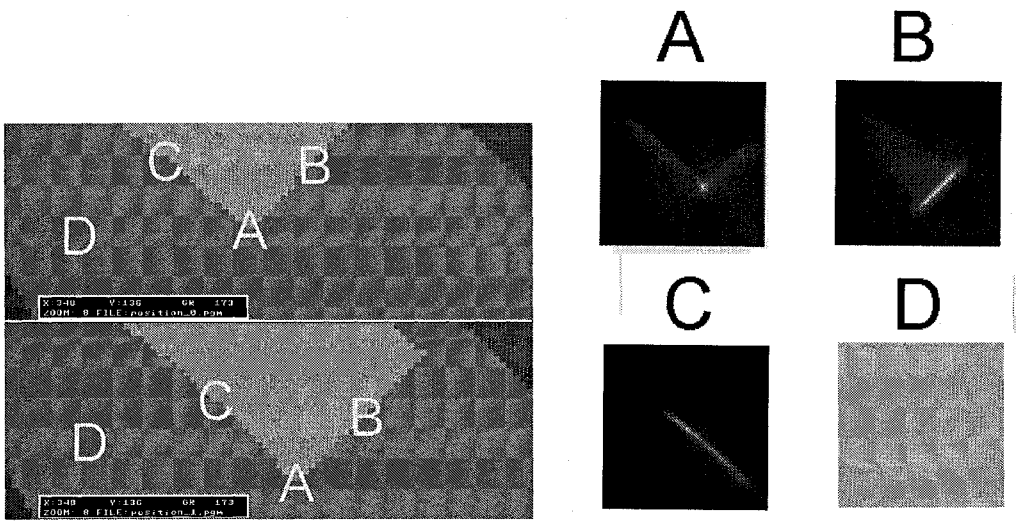


Figure 10.2: *Correlation-based comparisons: examples from Sequence 1 (Fig. 10.4) The* aperture problem *allows only some distinct points to have unambiguous flow vectors. 'A' is one distinct point, a clearly defined corner, its comparison function has a single peak. 'B' and 'C' are along edges, and the distribution of their comparison function is defined in the direction normal to the edge, but spread out along due to an uncertain match. 'D' has a uniform comparison response due to no features. The comparison was done between 10 x 10 pixel image apertures, and the comparison function is performed for a search window of 30 x 30 pixels.*

This modified response is calculated for every position in the MxM search window and a response image, such as the four in the right of Fig. 10.2, is generated for each pixel in $frame_j$. This response typically resembles one of the response images shown, a tight peak for a distinct image neighbourhood, ridges for image neighbourhoods containing line edges (where the aperture problem does not allow a distinct response), and regions of a flat response for uniform image sections. The response can be described as a flow vector defined by the center of a response peak, with major and minor axis values of the response shape representing the confidence in that flow vector. In the experiments shown in this chapter, only points with a small major and minor axis, corresponding to an un-ambiguous optic flow vector were used in further processing.

No iterative methods [102] were used to propagate flow from pixels with a higher confidence to the flow of pixels with less distinct neighbourhoods. A binary decision was made for each pixel position representing whether the spread of the correlation match function was below a threshold. Each vector set then consists of a binary confidence image, and a two-dimensional array of flow vectors.

10.9 Extracting Scene Point Position From Image Feature Trajectories

A set of trajectories was calculated from all $J-1$ vector sets, tracing the motion of distinct points in the panoramic sequence. A trajectory would start with a point of high confidence, and advance to a new position in the following image according to the image flow vector. If this new location in the next vector set had a high confidence, the location would be added to the trajectory. If this new location did not have a high confidence, its eight neighbour points would be examined for their confidence and the trajectory continued from this neighbour point if one was found. If none of these nine points were binary 'true' for their confidence, the trajectory ended. This process continued until either the end of a trajectory was found or the end of the sequence was reached. Thus a trajectory consisted of a list of up to $J-1$ points, each point representing a link in a chain of points whose flow vectors, or their immediate neighbours, have a high confidence.

The trajectories were then pruned to those that spanned a minimum angular travel through the course of the sequence, and the separate Panoramic Hough transform image prepared for each trajectory, providing a R_{main} and θ_{main}. Trajectories were filtered by θ_{main} value to find those that corresponded to stationary scene points. This filtering is based on the observation that if the panoramic camera is translating through 3D space at a heading corresponding to θ_{CAMERA}, then stationary objects will be translating past along trajectories with $\theta_{main} = \theta_{CAMERA} \pm 90°$.

θ_{CAMERA} may not be known *a priori* and must be estimated. This is done by creating a histogram of all θ_{main} values, and choosing a heading 90° to the left or right of a maximum. This defines stationary as being the direction of greatest cohesive movement. Which of two possible θ_{CAMERA} hypothesis values is determined to be correct is chosen simply by examining the relative angular movement of trajectories, since points will move away from θ_{CAMERA} towards $\theta_{CAMERA} + 180°$.

At this stage, θ_{CAMERA} is determined, and a set of trajectories that represent motion of scene points passing on the left and right of the panoramic camera remain. The angular positions θ_i for each frame index i in a given trajectory are then converted to linear positions, normalized to radial distance, according to Eqn 10.3.

An assumption of constant camera travel speed is adopted to simplify the following discussion, but not necessary in practice. If such as assumption is true, a normalized tangential velocity TV_{ave} for the trajectory can be found by averaging the differences between subsequent y_i values as shown in Fig. 10.3.

$$y_i = tan(\theta_i - \theta_{main}) \tag{10.3}$$

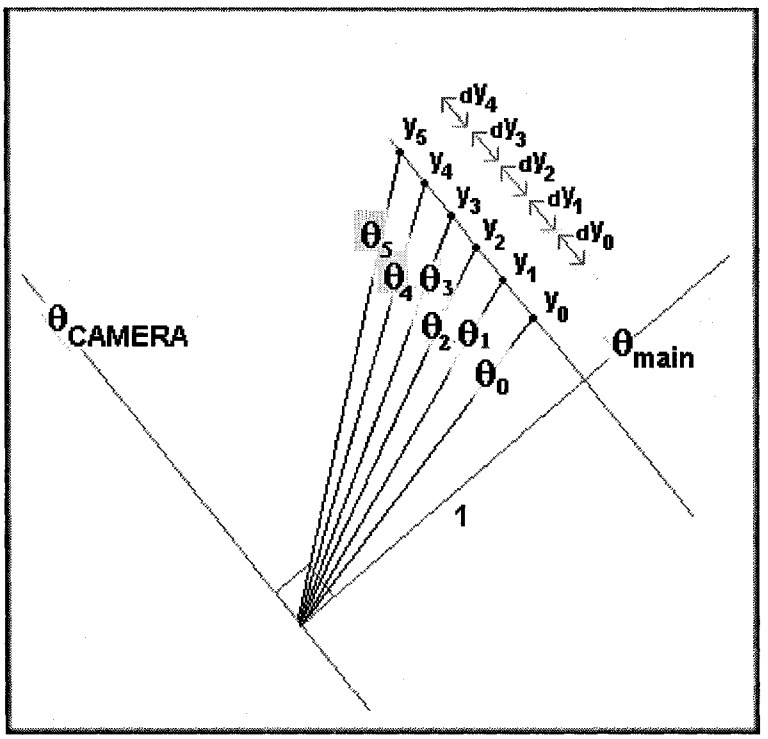


Figure 10.3: *Conversion of all θ_i readings in a trajectory to normalized tangential positions y_i and velocities dy_i distances along a direction parallel with the camera heading.*

The distances D_{main} for each trajectory can be found, to a scale factor, by equating all the trajectories' tangential velocities to one common scene speed S. D_{main} can be determined for all trajectories using the relation $\frac{S}{D_{main}} = TV_{ave}$.

The relative location of the scene point to the camera can be obtained by $D_{main} \cdot y_i$ at any frame time i. Finally, the height of this point can be found using R_{main} and D_{main}.

Only stationary objects were used in the experiments below, but the position and motion of moving objects can be also found. Moving scene points can be identified as those whose θ_{main} differ significantly from the two directions $\theta_{CAMERA} \pm 90°$, or those who's radially normalized y_i positions cannot be multiplied by a consistent scalar to equal the camera positions.

10.10 Experiments

Two synthetic image sequences of $J = 30$ frames each were generated, both produced with ray tracing methods simulating the catadioptric camera. The 500 x 500 pixel images were

captured from a simulated standard perspective view camera with a focal length of 590 pixels, with the focal point 15.1 units from a the center of spherical mirror of radius 4.9 units. (These numbers were chosen to match the parameters of a real system).

The first sequence (Fig. 10.4) uses a scene of untextured polygons where only the corners provide points for unambiguous optic flow determination, whereas the second uses highly textured polygons.

Optic flow vectors were generated between each of the successive images, $J - 1 = 29$ vector sets in all. The optic flow was calculated with a aperture (window size) of 10 pixels and a search window of $\pm$ 15 pixels ($M = 30$) in both sequences.

One set of trajectories was calculated from all 29 vector sets, tracing the motion of distinct points in the panoramic sequence. These trajectories were then pruned to those that spanned at least 20° of motion in Sequence 1 and 45° in Sequence 2. In the first image sequence, there were 2347 trajectories found, 937 of which were found to cover at least 20°. The second more textured sequence produced, not unexpectedly, more trajectories, 8301 in all, 3165 of which traveled a minimum of 45°.

The Panoramic Hough transform technique was then applied to each of these trajectories, locating a θ_{main}, R_{main} horizontal line definition. These θ_{main} values were aggregated across all the trajectories in a histogram, which was then filtered and the camera's heading θ_{CAMERA} determined. Trajectories representing stationary scene points were chosen according to having a θ_{main} close to θ_{CAMERA}. Other trajectories were assumed to be that of moving objects and neglected in these two experiments (since no moving objects were part of the image sequences). The location and distance of each trajectory in 3-dimensional space were calculated as described above, and the estimated 3-D point written to a database. This point was the relative position to the camera in the middle of the sequence. A view of the 3-D point sets generated from each image sequence is shown.

10.10.1 Sequence 1: Untextured Polygons

In this sequence, the simulated panoramic sensor passes through a scene of uniformly shaded polygons. The virtual camera passes by four polygons are passed by completely, and most of their corners produce trajectories that are tracked by optic flow correlation. A perspective view of this scene and two sample images in the sequence generated from this scene are shown below in Fig. 10.4.

The trajectories are all drawn together below in Fig. 10.5. Each trajectory is shown as a contour of connecting line segments, together they demonstrate the flow of distinct points. Each trajectory is processed with the Panoramic Hough transform, and a histogram is constructed of the resultant θ_{main} angles. This histogram is smoothed with a linear low

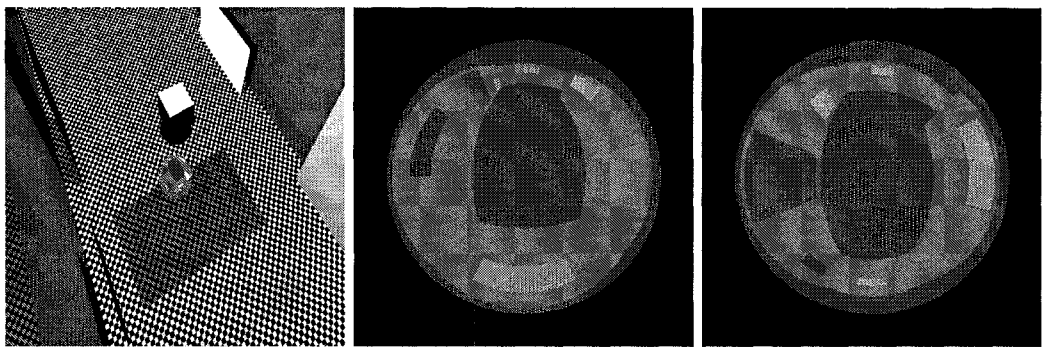


Figure 10.4: *A view of the scene used to create image Sequence 1 and two sample images.*

pass filter of width equal to 30° of the histogram width. Two opposite camera heading hypotheses of 178° or 358° were extracted. The hypothesis of 358° was verified by the angular direction of the trajectory points, as that they move away from θ_{CAMERA}.

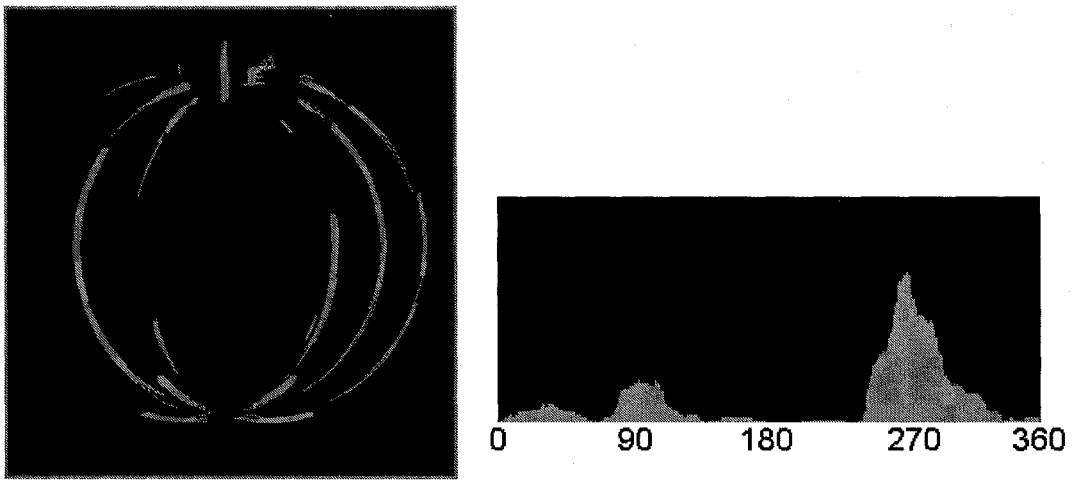


Figure 10.5: *Left: Trajectories of distinct points in Fig. 10.4 found by linking optic flow vectors between successive frames. Right: Smoothed histogram of θ_{main} of trajectories as determined with the Panoramic Hough transform. This low-pass filtered version of the histogram is used for the determination of camera heading (θ_{CAMERA}).*

The 3-D location of the tracked trajectories of points passed on the right side of the camera are shown along with another view of the original scene to demonstrate the reconstruction of the distinct corner points in Fig. 10.6. Groupings of points are labelled to match the corners whose trajectories were tracked.

10.10.2 Sequence 2: Textured Polygons

In the second sequence, shown in Fig. 10.7, the scene consists of texture mapped polygons where a distinct optic flow can be calculated for most 10 x 10 pixel window positions.

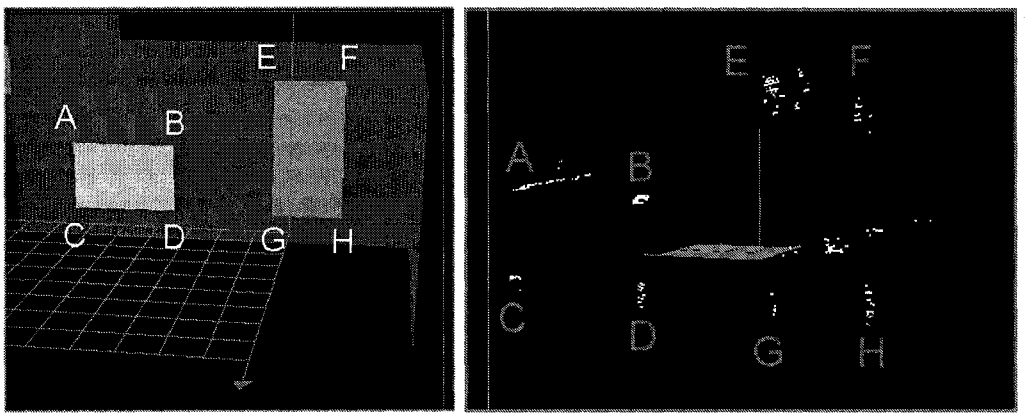


Figure 10.6: *Left: A view of the original scene. Right:* shape from motion *3-D reconstruction with points labelled corresponding to the 8 corners in the original scene (the object in the center is the axis and X-Y axis grid from the 3-D viewing program).*

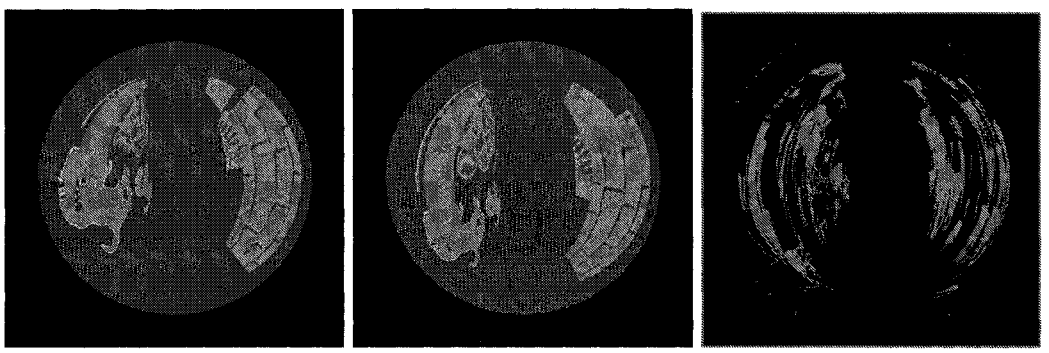


Figure 10.7: *Two images from Sequence 2, and a composite of the flow trajectories.*

The 3-D reconstruction of Sequence 2 is shown below, along with a perspective view of the original scene for comparison. More accurate results are expected if more sophisticated optic flow methods are used. Because of the very basic optic flow method used (correlation with no smoothing/interpolation of flow field) many trajectories were lost. However, since the aperture size for the correlation window was reasonable large (10 x 10 pixels), many pixels in the neighbourhood of a distinct point were able to be tracked, and so the loss of many still left enough remaining to discern the general shape. Only qualitative comparison of the reconstructed scene to the original was performed.

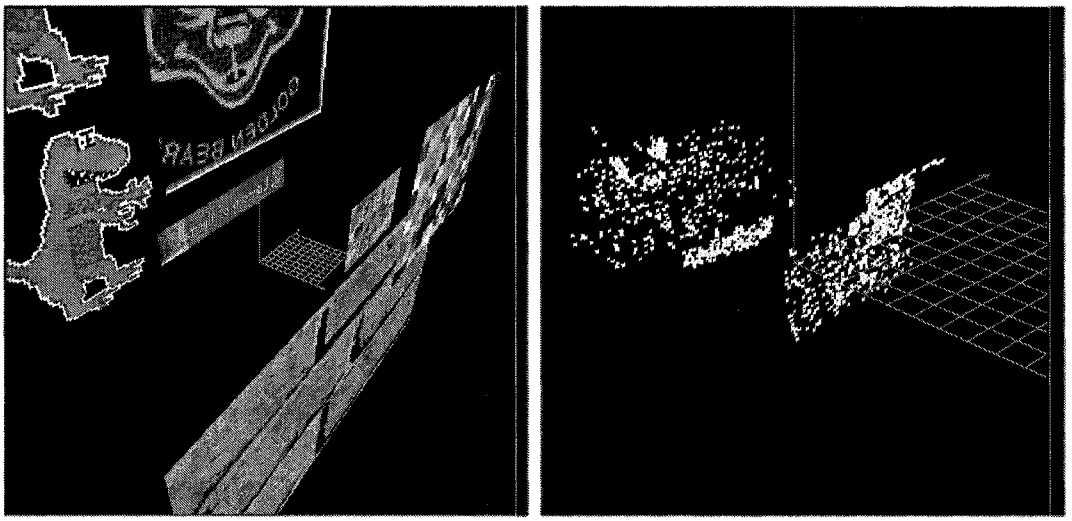


Figure 10.8: *Left: A view of the original scene. Right:* shape from motion *3-D reconstruction.*

10.11 Discussion

The 3D model points recreated were a reasonably accurate reconstruction of the initial scene when viewed qualitatively. It is difficult to display the results in a single or a few static images, the model is best viewed in a 3D viewing program where one can rotate around the model, as was done by the author. The experiments shown are preliminary, but the principles introduced were validated.

The reconstruction had some outlier noise points, but this was due to an inadequate number of correct trajectories provided by the optic flow and trajectory linking stage. Many were rejected due to not forming a distinct enough cluster in the transform space, and various thresholds (minimum angular travel, maximum deviation of θ_{main}, etc) had to be relaxed to provide enough points in the output model. A denser set of trajectories can be extracted from the imagery with better optic flow.

Methods that iteratively propagate the flow velocities to provide correct image motion for image neighbourhoods of low confidence, such as line edges which suffer from the aperture problem, would increase the number of valid trajectories. Thresholds can be then be tightened to produce more accurate reconstructions.

10.12 Conclusions

A structure from motion method for imagery from panoramic catadioptric image sensors was introduced and demonstrated with preliminary experiments. This method has not been carried to successful implementations with real experiments as with the stereo reconstruction and navigation applications of the PHT.

This structure from motion method utilizes the Panoramic Hough Transform method for extracting trajectories in images captured by sensors free from the single viewpoint restriction, and thus applicable to catadioptric systems with any radially symmetric mirror profile, such as conical or spherical mirrors.

The theory presented is applicable to a paradigm where the camera motion is restricted to a horizontal plane, and reconstruction can only be done for the frames where the camera was moving in a straight line. These are believed to be acceptable limitations suitable for mobile robotics applications.

Chapter 11

Conclusions

To perceive is to recreate. A vision system creates a model, an abstraction of the surrounding world in a language based on simplified primitives.

This thesis introduced a feature based computer vision system for a general class of panoramic image sensors different from the classical perspective projection cameras which previously dominated computer vision research. This enlarges the useful family of catadioptric panoramic sensors by providing feature extraction methods to process imagery from sensors that do not meet the *Single Viewpoint* criteria. This expands the useable family of catadioptric panoramic sensors from those that are only of a parabolic or hyperbolic radial profile to all catadioptric systems which have a convex radially symmetric profile .

Two working vision systems based on much of the same theory were designed and implemented in practice. Stereo vision and robot localization systems were built and tested in the lab, and some groundwork laid for using non-SVP panoramic sensors for shape-through-motion research.

11.1 Panoramic Stereo Vision

A method for panoramic stereo reconstruction using a bi-lobed catadioptric image sensor was shown. Stereo reconstruction with the general class of non-SVP optics was achieved, with the only restriction being that the mirror must be radially symmetric. A process for modeling a polyhedral world with this sensor was described with stages of feature detection and verification. Four error failure modes were identified, and shown to be corrected or reduced with the detection and verification paradigm.

The *Panoramic Hough Transform* was introduced and used in both detection and verification of horizontal line segment projections. Vertical edge segments were located with standard methods due to their projection to straight radial lines. Horizontal edge segments, which previously could not be extracted from non-SVP panoramic imagery with existing methods were detected using a combination of the new Panoramic Hough Transform and the *Identify and Remove* algorithm. The features were further processed to join, purge and predict other features in the form of abstract feature lists, with verification stages confirming and correcting the features. Verification consisted of a tracking stage utilizing the Panoramic Hough Transform. Finally some features were aggregated into closed regions representing rectangular polygons which were matched between the two mirror lobes, and a 3D model was constructed.

The synthetic experiments reliably reconstructed scene polygons, with errors within ranges expected by the resolution and equivalent vergence angle. The real image experiments were successful in obtaining a qualitatively correct model, and quantitative information was

provided as to the error accuracy. The feature extraction method proved quite robust, even with very roughly approximated geometric parameters. The sensitivity to calibration accuracy was shown with the bi-lobed panoramic sensor, which is an example of a narrow baseline stereo system. Further calibration to account for non-ideal mirrors was necessary. Also it should be noted that the errors were significant in one direction only, and that if this system was used in an application where the camera moved (as on a mobile robot), a more accurate 3D scene model could be created. Many stereo systems, especially those that produce depth maps by correlation, are in general not very accurate in practice for depth measurement. The depth error, due to geometrical limits, can be considered acceptable for one image frame captured at a single viewpoint, especially since the model is created for objects in all azimuth directions simultaneously.

The bi-lobed catadioptric image sensor design, along with one implementation of the necessary accompanying vision processing, was shown to validate this sensor's use for omnidirectional stereo reconstruction.

11.2 Panoramic Robot Navigation

A successfully working, near real-time system was implemented on a mobile robot, and robust localization demonstrated in the lab. The robot was able to track its location using vision only, with no odometry, sonar, laser range-finders or other sensors.

Two methods of providing position and orientation information for mobile robot navigation with a panoramic camera were presented, and results of synthetic and real experiments were reported. The localization methods used a tracking procedure to update the projection of scene landmarks from their location predicted by the estimated robot position. Both methods tracked vertices of polygons, but differed in how a vertex was found. A prototype mobile robot system was successfully demonstrated with the more advanced of the two methods, using a catadioptric image sensor with a spherical, non-SVP mirror profile.

The novel contribution to mobile robotic navigation lies in both methods as how the location of landmark projections in the image are predicted. The second method, which tracks junctions of line segment projections, was proved to be more robust than the first. The second method uses an application of the *Panoramic Hough Transform* which tracks straight line segments in catadioptric panoramic cameras free from the SVP mirror profile restriction. This expands the capabilities of a non-SVP localization system over status-quo feature-based systems in the literature that use image information from the azimuth dimension only of panoramic imagery.

11.3 Fulfillment of Thesis Objectives and Future Work

The development of the Panoramic Hough Transform enables feature detection without the Single Viewpoint criteria thus extending the family of usable catadioptric panoramic sensors. Three applications were given for the theory presented namely stereo reconstruction, mobile robot localization and shape-through-motion.

The stereo reconstruction system provides image understanding within one image frame creating an instantaneous 3D snapshot. These 3D snapshots could be aggregated over time and space to create a more comprehensive model. Likewise the feature extraction methods of model making could be applied to the single lobed mobile robot system to provide map creation and not just map following. Indeed stereo modeling and localization could converge to create a robot navigation system that does not require an *a priori* world model and therefore allow a mobile robot to work reliably in a polyhedral environment. Also a shape-through-motion technique was introduced which allows a 3D model to be generated from a non- polyhedral world.

Overall, new theory and methods were presented to further develop computer vision with panoramic image sensors.

Bibliography

[1] A. Optic flow computation: A unified perspective, ieee computer society press, los alamitos. *CA*, 92:1992, 1992.

[2] M. Jain A. Basu and X. Li. Improving boundary detection using variable resolution masks. *PRL*, 16(11):1205–1211, November 1995.

[3] G. Matas J. Illingworth A. Etemadi, J. Schmidt and J. Kittler. Low-level grouping of straight line segments. In *BMVC91*, pages 119–126, 1991.

[4] Y. Aloimonos. Visual navigation: Flies, bees and ugv's. In *Visual Navigation: From Biological Systems to Unmanned Ground Vehicles. Advances in Cimputer Vision*, volume 2, 1997.

[5] P. Anandan. Measuring visual motion from image sequences. Technical Report UM-CS-1987-021, 31, 1987.

[6] P. Andrea. Omni-directional catadioptric vision for soccer robots. In *citeseer.nj.nec.com/440571.html.*

[7] P. Askey. Canon eos-d30 review. In *http://www.dpreview.com/reviews/canond30/*, 2000.

[8] R. Bunschoten B. Krose, N. Vlassis and Y. Motomura. A probabilistic model for appearance-based robot localization. In *Image and Vision Computing*, volume 16(6), pages 381–391, May 2001.

[9] S. Baker and S. Nayar. A tutorial of catadioptric image formation. In *www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/BAKER/main.html.*

[10] S. Baker and S. Nayar. A theory of catadioptric image formation. In *IEEE ICCC Conference*, pages 392–197, 1998.

[11] J. Baldwin and A. Basu. 3d estimation using panoramic stereo. In *International Conference on Pattern Recognition*, volume 1, pages 91–100, Barcelona, Spain, September 2000.

[12] D. Ballard and C. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[13] A. Basu and J. Baldwin. *A Real-Time Panoramic Stereo Imaging System and its Applications.* Springer, 2001.

[14] A. Basu and D. Southwell. Omni-directional sensors for pipe inspection. In *IEEE SMC Conference*, pages 3107–3112, Vancouver, Canada, October 1995.

[15] R. Benosman and J. Devars. Panoramic stereovision sensor. In *Proc. 14th Intl. Conf. on Pattern Recognition*, pages 767–769, 1998.

[16] M. Betke and K. Gurvits. Mobile robot localization using landmarks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 135–142, May 1994.

[17] S. Bogner. Application of panospheric imaging to an armored vehicle viewing system. In *IEEE SMC Conference*, pages 3113–3116, 1995.

[18] S. Bogner. Introduction to panoramic imaging. In *IEEE SMC Conference*, pages 3100–3106, Vancouver, Canada, October 1995.

[19] J. Borenstein, H. Everett, L. Feng, and D. Wehe. Mobile robot positioning: sensors and techniques. 14:231–249, April 1997.

[20] E. Brassart and C. Cauchois. Syclop sensor web page. In *http://www.crea.u-picardie.fr/omnidirectional/index.html.*

[21] A. Bruckstein and T. Richardson. Omniview cameras with curved surface mirrors. In *IEEE Proc. Omnidirectional Vision 2000*, pages 79–84, 2000.

[22] C. Pegard E.Mouaddib C. Cauchois, E.Brassart and A.Clerentin. Technique for calibrating an omnidirectional sensor. In *IROS*, Kyongju, Korea, 1999.

[23] B.Marhic L.Delahoche A.Clerentin C. Pegard, E.Mouaddib and E.Brassart. Localisation methods for mobile robots using panoramic sensor. In *2nd Workshop on European Scientific and Industrial Collaboration*, 1999.

[24] J. Chahl and M. Srinivasan. Range estimation with a panoramic visual sensor. In *Journal of the Optical Society of America*, volume 14(9), pages 2144–2151, Sept 1997.

[25] J. Chahl and M. Srinivasan. Reflective surfaces for panoramic imaging. In *Applied Optics*, volume 36(31), pages 8275–8285, 1997.

[26] J. Chahl and M. Srinivasan. A complete panoramic vision system, incorporating imaging, ranging, and three dimensional navigation. In *IEEE Proc. Omnidirectional Vision 2000*, pages 104–111, 2000.

[27] D. Cobzas and H. Zhang. 2d robot localization with image-based panoramic models using vertical line features. In *Proc. of Vision Interface*, 2000.

[28] D. Cobzas and H. Zhang. Cylindrical panoramic image-based model for robot localization. In *IROS*, pages 94–99, 2001.

[29] D. Cobzas and H. Zhang. Mobile robot localization using planar patches and a stereo panoramic model. In *Proc. of Vision Interface*, pages 94–99, 2001.

[30] T. Conroy and J. Moore. Resolution invariant surfaces for panoramic vision systems. In *IEEE ICCV Conference*, pages 392–397, 1999.

[31] M. Fiala D. Southwell, A. Basu and J. Reyda. Panoramic stereo. In *ICPR*, pages 378–382, Vienna, Austria, August 1996.

[32] S. Derrien and K. Konolige. Approximating a single viewpoint in panoramic imaging devices. In *IEEE Proc. Omnidirectional Vision 2000*, pages 85–90, August 2000.

[33] U. Dhond and J. Aggarwal. Structure from stereo - a review. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1489–1510, November 1989.

[34] Z. Dodds and G. Hager. A color interest operator for landmark-based navigation. In *AAAI/IAAI*, pages 655–660, 1997.

[35] R. Duda and P. Hart. Use of the hough transform to detect lines and circles in pictures. *Communications of the ACM*, 15, 1972.

[36] S. Peleg et al. Cameras for stereo panoramic imaging. In *CVPR*, volume 1, pages 208–214, 2000.

[37] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Cambridge, Massachusetts, 1993.

[38] O. Faugeras and G. Toscani. The calibration problem for stereo. In *CVPR*, pages 5–20, 1986.

[39] C. Fermuller. Passive navigation as a pattern-recognition problem. *IJCV*, 14(2):147–158, March 1995.

[40] M. Fiala and A. Basu. Line segment extraction in panoramic images. In *Journal of WSCG*, volume 10, pages 179–186, 2002.

[41] W. Frei and C. Chen. Fast boundary detection: A generalization and a new algorithm. In *IEEE Trans. Computers*, volume 26, pages 988–998, Oct 1977.

[42] E. Magli G. and Olmo. Determination of a segment endpoints by means of the radon transform. In *ICECS (IEEE International Conf. on Electronics, Circuits and Systems)*, 1999.

[43] E. Magli G. and Olmo. On high resolution positioning of straight patterns via multiscale matched filtering of the hough transform. *PRL*, 22(6-7):705–713, May 2001.

[44] A. Georghiades G. Hager, D. Kriegman and O. Ben-Shahar. Toward domain-independent navigation: Dynamic vision and control. In *CDC*, Dec 1998.

[45] E. Yeh G. Hager, D. Kriegman and C. Rasmussen. Image-based prediction of landmark features for mobile robot navigation. In *IEEE Robotics and Automation*, volume 2, pages 1040–1046, 1997.

[46] C. Geyer and K. Daniilidis. Catadioptric camera calibration. In *ICCV (1)*, pages 398–404, 1999.

[47] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical applications. In *ECCV (2)*, pages 445–461, 2000.

[48] C. Geyer and K. Daniilidis. Catadioptric projective geometry. In *International Journal of Computer Vision*, volume 43, pages 223–243, 2001.

[49] C. Geyer and K. Daniilidis. Omnidirectional vision and catadioptric geometry. In *Vision Interface*, Ottawa, Canada, 2001.

[50] C. Geyer and K. Daniilidis. Structure and motion from uncalibrated catadioptric views. In *IEEE CVPR*, December 2001.

[51] M. Yamamoto H. Ishiguro and S. Tsuji. Omni-directional stereo for making global map. In *3rd Intl. Conf. Computer Vision*, pages 540–547, 1990.

[52] M. Yamamoto H. Ishiguro and S. Tsuji. Panoramic representations of scenes around a point. In *IROS*, 1991.

[53] G. Hager and C. Rasmussen. Robot navigation using image sequences. In *AAAI Conference on Artificial Intelligence*, pages 938–943, 1996.

[54] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. Cambridge, UK, 2000. Cambridge University Press.

[55] A. Hicks. Reflective surfaces as computational sensors. In *Second IEEE Workshop on Perception for Mobile Agents*, 1999.

[56] B. Horn and B. Shunck. Determining optic flow. In *Artificial Intelligence Vol. 17*, pages 185–203, 1981.

[57] P. Hough. Method and means for recognizing complex patterns. In *US Patent 3069654*, 1962.

[58] J. Howell and B. Donald. Practical mobile robot self-localization. In *ICRA*, San Francisco, USA, Apr 2000.

[59] M. Hueckel. An operator which locates edges in digitized pictures. *Journal of the ACM*, 18(1):113–125, 1971.

[60] S. Beauchemin J. Barron, D. Fleet and T. Burkitt. Performance of optical flow techniques. *CVPR*, 92:236–242.

[61] S. Nayar J. Gluckman and K. Thorek. Real-time omnidirectional and panoramic stereo. In *Proceedings of the 1998 DARPA Image Understanding Workshop*, Nov 1998.

[62] H. Takemura J. Shimamura, N. Yokoya and K. Yamazawa. Construction of an immersive mixed environment using an omnidirectional stereo image sensor. In *IEEE Proc. Omnidirectional Vision 2000*, pages 62–69, 2000.

[63] P. Cohen J. Weng and M. Herniou. Camera calibration with distortion models and accuracy evaluation. In *PAMI*, volume 14, Oct 1992.

[64] M. Jogan and A. Leonardis. Panoramic eigenimages for spatial localisation. In *Computer Analysis of Images and Patterns*, pages 558–567, 1999.

[65] Y. Yagi K. Yamazawa and M. Yachida. 3d line segment reconstruction by using hyperomni vision and omnidirectional hough transforming. In *ICPR00*, pages Vol III: 487–490, 2000.

[66] Y. Kim and J. Aggarwal. Positioning 3-d objects using stereo images. *IEEE Journal of Robotics and Automation*, RA-3(4):361–373, August 1987.

[67] J. Lee and I. Jurkevich. Coastline detection and tracing in sar images. In *IEEE Trans. Geoscience and Remote Sensing*, volume 28 (4), 1990.

[68] G. Lui and R. Haralick. Two practical issues in canny's edge detector implementation. pages 680–682.

[69] R. M. Boldt, R. Weiss and E. Riseman. Geometric grouping applied to straight lines. In *CVPR86*, pages 489–495, 1986.

[70] G. Medioni M. Nicolescu and M. Lee. Segmentation, tracking and interpretation using panoramic video. In *IEEE Proc. Omnidirectional Vision 2000*, pages 21–28, 2000.

[71] H. Herman M. Ollis and S. Singh. Analysis and design of panoramic stereo vision using equi-angular pixel cameras. *Technical Report: CMU-RI-TR-99-04*, January 1999.

[72] T. Majumdar and B. Bhattacharya. Extraction of shoreline and drainage patterns from aerial mss thermal ir data over cambay basin, india – an attempt to automize the threshold selection. In *Pattern Recognition*, volume 24, pages (2) 157–164, 1991.

[73] C. Marques and P. Lima. A localization method for a soccer robot using a vision-based omni-directional sensor. In *RoboCup*, pages 96–107, 2000.

[74] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings Royal Society of London Bulletin*, 204:301–328, 1979.

[75] M. Mason and I. Davenport. Accurate and efficient determination of the shoreline in ers-1 sar images. *IEEE Trans. on Geoscience and Remote Sensing*, pages 1243–1253, 1996.

[76] J. Matjaž and A. Leonardis. Robust localization using eigenspace of spinning-images. In *IEEE Workshop on Omnidirectional Vision - OMNIVIS 2000, Hilton Head Island, South Carolina*, pages 37–44. IEEE Computer Society, June 2000.

[77] F. Moffitt and E. Mikhail. *Photogrammetry, 3rd Edition.* Harper and Row, New York, 1980.

[78] G. Lacey N. Winters, J. Gaspar and J. Santos-Victor. Omni-directional vision for robot navigation. In *IEEE Proc. Omnidirectional Vision 2000*, pages 21–28, 2000.

[79] H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from images sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):565–593, 1986.

[80] V. Nalwa. *A Guided Tour of Computer Vision.* Addison-Wesley, Reading, Massachusetts, 1993.

[81] V. Nalwa and T. Binford. On detecting edges. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8(6):699–714, 1986.

[82] S. Nayar. In *US Patent No.5760826.*

[83] S. Nayar and T. Boult. Omnidirectional vision systems: Pi report. In *Proceedings of the 1997 DARPA Image Understanding Workshop*, May 1997.

[84] R. Nevatia and K. Babu. Linear feature extraction and description. In *CVGIP*, volume 13, pages 257–269, 1980.

[85] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(2):139–154, March 1985.

[86] G. Olmo and E. Magli. All-integer hough transform: Performance evaluation. In *ICIP01*, page Architecture and Software, 2001.

[87] Z. On. Using image-based panoramic models for 2d robot localization. In *citeseer.nj.nec.com/425204.html.*

[88] C. Owen and U. Nehmzow. Middle scale robot navigation - A case study. In *AISB–97 Workshop on Spatial Reasoning in Mobile Robots and Animals.* University of Manchester, 1997.

[89] C. Owen and U. Nehmzow. Landmark-based navigation for a mobile robot. In *Proc. Simulation of Adaptive Behaviour.* MIT Press, 1998.

[90] T. Pajdla. Robot localization using shift invariant representation of panoramic images. In *Research Report K335-CMP-1998-170*, Czech Technical University, Prague, November 1998.

[91] M. Pietikainen and D. Harwood. Depth from three camera stereo. In *CVPR*, volume 1, pages 2–8, 1986.

[92] M. Gupta P.Musilek and G.Schmidt. Adaptive fuzzy approach to edge detection.

[93] W. Pratt. *Digital Image Processing.* John Wiley and Sons, Second Edition, New York, 1991.

[94] D. Qing Q. Gao and S. Lu. Fuzzy classification of generic edge features. In *ICPR (III)*, pages 672–675, 2000.

[95] Y. Qin-Zhong. A preprocessing method for hough transform to detect circles. In *CVPR96*, 1986.

[96] T. Maiere R. Benosman and J. Devars. Multidirectional stereovision sensor. In *International Conference on Pattern Recognition*, pages 161–165, September 1996.

[97] I. Cox R. Boie and P. Rehak. On optimum edge recognition using matched filters. In *CVPR86*, pages 100–108, 1986.

[98] T. Lam S. Yuen and N. Leung. Connective hough transform. *IVC*, 11:295–301.

[99] G. Scott. Local and global interpretation of moving images. Academic Press/Morgan Kaufmann, Apr 1988.

[100] J. Shen and S. Castan. An optimal linear operator for edge detection. In *CVPR86*, pages 109–114, 1986.

[101] H. Shum and R. Szeliski. Stereo reconstruction from multiperspective panoramas. In *ICCV (1)*, pages 14–21, 1999.

[102] A. Singh. Optic flow computation: A unified perspective. Los Alamitos, California, USA, 1991. IEEE Computer Society Press.

[103] J. Sivic. Interactive control of a mobile robot with panoramic camera. In *Technical Report K335-CMP-1998-165*, Czech Technical University, Prague, August 1998.

[104] T. Sogo. Real-time target localization and tracking by n-ocular stereo. In *IEEE Workshop on Omnidirectional Vision (OMNIVIS'00)*, pages 153–160, 2000.

[105] T. Svoboda. Estimation of mobile robot position using panoramic cameras. In *(Thesis Proposal) Czech Technical University, Center for Machine Perception*, Prague, Czech Republic, 1998.

[106] C. Fermuller T. Brodsky and Y. Aloimonos. Structure from motion: Beyond the epipolar constraint. *International Journal of Computer Vision*, 37(3):231–258, 2000.

[107] A. Ohya T. Yata and S. Yuta. A fast and accurate sonar-ring sensor for a mobile robot. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pages 630–636, May 1999.

[108] A. Ohya T. Yata and S. Yuta. Fusion of omni-directional sonar and omni-directional vision for environment recognition of mobile robots. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 3926–3931, April 2000.

[109] A. Ohya T. Yata and S. Yuta. Using one bit wave memory for mobile robots' new sonar-ring sensors. In *Proceedings of the 2000 IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 3562 – 3567, October 2000.

[110] R. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *CVPR*, pages 364–374, 1986.

[111] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *ICRA*, pages 1023–1029, 2000.

[112] V. Venkateswar. Extraction of straight lines in aerial images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-14(11):1111–1114, November 1992.

[113] G. Wei and S. Ma. Two plane camera calibration: A unified model. In *CVPR91*, pages 133–138, 1991.

[114] K. Wetzel and D. Frosini. When digital cameras need large pixel areas. In *http://www.kodak.com/US/plugins/acrobat/en/digital/ccd/largePixels.pdf*, March 1999.

[115] T. Wu X. Li, C. Shanmugamani and R. Madhavan. Correlation measures for corner detection. *CVPR*, 86:643–646.

[116] S. Kawato Y. Yagi and S. Tsuji. Real-time omni-directional image sensor (COPIS) for vision-guided navigation. *IEEE Transactions on Robotics and Automation*, 10:11–22, 1994.

[117] R. Chellappa Y. Zhou and V. Venkateswar. Edge detection using the directional derivatives of a space varying correlated random field model. *CVPR*, 86:115–121.

[118] O. Faugeras Z. Zhang, R. Deriche and Q. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1-2):87–119, 1995.

[119] X. Zhang and H. Burkhardt. Grouping edge points into line segments by sequential hough transformation. In *ICPR*, volume 3, pages 676–679, 2000.

Appendix A

System Details

This chapter shows implementation details for the research conducted for this thesis. The intention is to further describe the system to interested readers, and in the hope that others may use the results of these years of labour in their work.

The functionality of different modules for processing the panoramic images are described. Some of the programs are applicable to all three applications (stereo reconstruction, robot localization and shape-through-motion), and others are specific to one application.

The system was, with the exception of the prototype robot, designed to be modular and functionality was separated into separate programs for each stage. The system consisted of C programs that communicated mostly via text files. With the exception of the original images, and pre-processed forms thereof, all files were standard ASCII text to facilitate visibility during the system's design and debugging. There were different text formats for each information type, and all were parsed with one of two test parsing libraries built by the author.

The programs were all run from a UNIX or LINUX command line platform. Originally the system was started in DOS, but migrated to LINUX to reduce memory management issues. Each program takes one or more input files, performs some processing, and then exits producing an output file used by the next program. During design and debugging, these programs were invoked manually from the command line, but during the final working system operation, this was performed by batch files. For example for processing the input image captured by the digital camera in the stereo reconstruction system, the image was copied to *in.pgm* and a single batch file invoked.

These run-time batch files, and necessary configuration files were created by other batch files, which could all be set up by running a single first program. This program *MAKE_SCRIPTS.C* (for reconstruction) or *MAKE_SCRIPTS_TRACK.C* (for localization) could be ran after providing the the basic geometry parameters in a single file.

The practical work in this thesis implemented vision systems for spherical mirrors only, i.e. mirrors of circular profile. The PHT information is encoded in look-up table files, which could be generated for any profile. Only a few programs implicitly assume spherical geometry and it would not be difficult to modify the system to handle other profiles. The system was designed with the desire of being "profile agnostic" in mind.

A.0.1 File Types

Files can be divided into those that hold the information of the image and abstract feature themselves, and those containing configuration and instruction information. These types can be identified by their file extension and are listed below in Tables A.1, A.2.

Table A.1 displays file types that contain information that would be flowing through

File Extension	File Type	Information Conveyed	Application
.PGM	Binary	2-D Pixel Array	3D,ROB,SHAPE
.PGM_LIST	Text	image sequence file list	ROB,SHAPE
.FLOW	Binary	2-D Optic Flow Array	SHAPE
.FLOW_LIST	Text	2-D .FLOW file list	SHAPE
.TRAJ	Text	2-D pixel trajectories frame num, U, V	SHAPE
.FEA	Text	horizontal segment projections vertical segment projections vertical polygon face projections	3D,ROB
.VTX	Text	vertex location and type	ROB
.PANO_VTX_TRACK	Text	3D landmark location and type	ROB
.SCENE	Text	3D Polygon definitions	3D,ROB,SHAPE

Table A.1: *Run-time File types: Files used during actual vision processing.* 3D *refers to files used in 3D reconstruction.* ROB *refers to files used in mobile robot localization.* SHAPE *refers to files used in shape-through-motion*

the processing in any implementation of this system. Table A.2 below lists file types that hold configuration, calibration, batch files and support information used in this software implementation.

File Extension	File Type	Information Conveyed	Application
.GEOMAIN	Text	geometry parameters for entire sensor, input image file name, sensor handle	3D,ROB,SHAPE
.GEO	Text	geometry parameters for single lobe	3D,ROB,SHAPE
.LOOKUP	Text	Panoramic Hough Lookup Table	3D,ROB
.CFG	Text	lookup table creation parameters	3D,ROB,SHAPE
.BAT	Text	batch file to run programs	3D,ROB,SHAPE
.POV	Text	Povray commands, objects	3D,ROB,SHAPE

Table A.2: *Support File types: Files containing configuration, geometry, etc information not in the main vision processing data flow.* 3D *refers to files used in 3D reconstruction.* ROB *refers to files used in mobile robot localization.* SHAPE *refers to files used in shape-through-motion*

All image data is contained in the binary .PGM file format. The original image is provided to the system in PGM format, and reconstructed image and scene views are produced in this format.

The image features consist of projections of horizontal and vertical line segments and projections of vertical posed rectangular polygon faces. These are contained in .FEA files. The highest level abstraction in these systems is the 3D polygon representation (the vision system *language* from the introductory chapter) which is in the .SCENE format. The synthetic image creation was performed by *Povray* which is a public domain ray-tracing package and used files in the .POV format. A conversion utility converted the custom .SCENE files into .POV in these cases.

A.0.2 PGM Image File Format

The vision processing systems start with an input 2-D intensity array transferred from the image plane of the optical sensing device. The .PGM file format was chosen at the start of the thesis for it's simple, straightforward design. A .PGM file (Table A.3 below) consists simply of a brief text header, and then an uncompressed binary byte-stream, one byte for every pixel in the image.

#Comment line (any number) (Carriage Return 0x0D)
500 400 (Carriage Return 0x0D) *image width and height, 500 x 400 in this case*
255 (Carriage Return 0x0D) *maximum greyscale value*
... greyscale data (integer binary) 0x00=black 0xFF=white ...
.....
.....

Table A.3: *PGM file format: for greyscale images.*

A.0.3 PGM_LIST Image File Format

This is an ASCII list of .PGM files, one per line for use with *MOVIE.C.*

A.0.4 FEA file format

The .FEA format represented the two levels of information abstraction between the input images and the output 3D model. Projections of horizontal and vertical line segments were produced by *PANO_SEG.C* and expressed as four types *ha_seg, hb_seg, va_seg, vb_seg* primitives as seen in Fig. A.1.

Feature combining, tracking and purging programs work on line segment features in files of this format. *PANO_VRECT.C* finds vertical polygon projections from these line segment projection features, adds the 5th primitive type in .FEA files, the *vrect* primitive.

As with all the text files used by programs in this thesis, the // and /**/ commenting notation was used to for comments.

A.0.5 LOOKUP file format

The .LOOKUP file contains the Panoramic Hough Transform information for the forward transform $R_{main} = PH(R_i, d\theta)$. It contains R_{main} values for a range of $PH(R_i$ and $d\theta)$

```
//PANO_FEA_PURGE.C from input file <thu_in_tr_v1.fea>//Input image = <pano_track_li
//-------------------------------------------
width 2160 height 1440
image_centerx 1061 image_centery 761
min_radius 83 max_radius 209 horizon 173
image pano_track_line.pgm
lookup first_inner_lobe.lookup
//-------------------------------------------
//-----------HA edges - horizontal edges (dI/dR pos)
//ha_seg center_theta spread radius | begin_theta end_theta num_pixels avg_edge
ha_seg 1 0 99 | 333 36 186 0
ha_seg 358 0 114 | 30 42 146 0
//-----------HB edges - horizontal edges (dI/dR neg)
//hb_seg center_theta spread radius | begin_theta end_theta num_pixels avg_edge
hb_seg 7 0 104 | 8 45 409 0
hb_seg 358 0 198 | 342 13 357 0
hb_seg 355 0 101 | 8 34 54 0
//----------VA edges - vertical edges (dI/dT pos) cw
//va_seg theta begin_radius end_radius num_pixels avg_edge
va_seg 9 97 104 | 0 0
va_seg 50 137 177 | 0 0
//----------VB edges - vertical edges (dI/dT neg) cw
//vb_seg theta begin_radius end_radius num_pixels avg_edge
vb_seg 36 110 116 | 0 0
vb_seg 43 133 193 | 0 0
//----------Vertical Rectangles -----------
//vrect theta spread begin end | radius1 radius2 | avg_grey num_pixels avg_edge
vrect 179 10 199 232 | 96 146 | 0 0 0
vrect 184 4 160 183 | 114 132 | 255 0 0
```

Figure A.1: *.FEA file describing segment and polygon projections in a 2D image.*

values. This is loaded into an internal memory array by programs, and remapped to provide the inverse $R_i = PH^{-1}(R_{main}, d\theta)$ function.

A.0.6 SCENE file format

The .SCENE file format was designed for the *FVE.C* program as a command language, and has the primitives of polygons, viewports, objects, virtual cameras, image grabbing and more. A subset of these capabilities were used in this thesis, basically just polygon primitives and that to create a single viewport. An example is shown in Fig. A.3 below.

A.0.7 Mobile Robot Localization Files

The panoramic localization system has some overlap with the Stereo Reconstruction system as far as using .SCENE 3D models, .LOOKUP, .GEO, .GEOMAIN and .FEA segment feature files. The .FEA files are used in the second of the two localization methods, which use line junctions to define vertices.

The other files specific to the mobile robot localization schemes are .VTX, .PANO_VTX_TRACK and .PGM_LIST files.

```
//Lookup created by GEN_LOOKUP.C
//geo_camera_focal length (geo_camera_focal)=6050.000000
//Height of mirror on Y axis (geo_camera_loc)=48.700001
//geo_mirror_radius=1.950000
//mirror center image_centerx=1061
//              image_centery=761
//Min,Max radius (geo_min_radius,geo_max_radius)=83,209
//Number of discrete line heights (geo_num_lines)=126
//Angular step for each x-pos in Hough image (geo_angle_step)=0
image_centerx 1061 image_centery 761
//Lookup table lines format:
//  image_radius,image_edge_angle,world_edge_height,
//  angle_to_world_edge, closest radius, ray slope
//Range height_min=-3.240480 to height_max=13.279836 step=0.131114
//---Image radius=83.000000
83.000000 -1.228267 0 -0.209440 83
   find
83.000000 -1.348237 0 -0.174533 83
   find
83.000000 -1.478132 0 -0.139626 83
   find
83.000000 -1.478132 0 -0.104720 83
   find
83.000000 -1.348237 0 -0.104720 84
```

Figure A.2: *.LOOKUP file holds a lookup table performing the forward Panoramic Hough Transform.*

```
viewport
   name=main
   xwidth=450
   ywidth=450
   xoffset=45
   yoffset=15
   f=2.0
   ccd_xwidth=1.0
   ccd_ywidth=1.0
   pos=-30.318,-200.952,210.213
   ix=-0.987688,0.156435,0.000000
   iy=0.110616,0.698401,0.707107
   iz=0.110616,0.698401,-0.707107
/viewport

poly
   colour=255
   vtx=-45.000000,13.000000,16.600000
   vtx=-45.000000,13.000000,38.400002
   vtx=-45.000000,-15.000000,38.400002
   vtx=-45.000000,-15.000000,16.600000
/poly

poly
   colour=255
   vtx=99.000000,29.500000,28.500000
   vtx=99.000000,29.500000,56.000000
```

Figure A.3: *.SCENE file represents 3D polygons. Top section is to create a viewport for viewing in* FVE.C.

A.0.8 VTX, PANO_VTX_TRACK file formats

The .VTX file format communicates vertex position and type between mobile robot localization programs. A vertex ID, a source polygon ID (not used in the example), three type descriptors and the angle, and radius are included on each line. The type descriptors describe if it's a corner of a light or dark polygon, and which of the four corners it corresponds to.

```
//PANO_BLINDCHECK_VTX.C
//0 Vertices replaced by estimated angle in feature file <a_fix_0.vtx>
//-------------------------------------------
width 484 height 484
image_centerx 242 image_centery 242
min_radius 85 max_radius 190 horizon 157
image a_fix_0.pgm
lookup single_only_lobe.lookup
//-------------------------------------------
//-2 types of rectangles: light,dark
//-4 types of vertices for a rectangle, ccw=left
//light_inside/dark_inside ccw/cw angle radius
4 -1 light_inside ccw bottom    229.290054 151.000000
5 -1 light_inside cw bottom    250.466537 148.000000
6 -1 light_inside ccw bottom    259.229218 148.000000
22 -1 light_inside cw top    205.922928 169.000000
23 -1 light_inside ccw top    229.290054 168.000000
24 -1 light_inside cw top    250.466537 169.000000
25 -1 light_inside ccw top    259.229218 169.000000
```

Figure A.4: *.VTX file holding label, type and location information for vertex landmarks.*

The PANO_VTX_TRACK file extension is for a file which contains a list of vertex landmarks, their type and their 2D location. The vertex ID numbers are the same as in the .VTX file, and together are used to triangulate position. The .PANO_VTX_TRACK file is created from the original 3D model .SCENE file. For an image sequence there is a .VTX file for every image but only one .PANO_VTX_TRACK file for the environment.

A.0.9 POS file format

The .POS file format holds the result of the robot localization programs, the camera's (C_x, C_y, C_θ) is provided for each frame (Fig. A.5).

A.0.10 Shape-Through-Motion Files

The shape through motion part of the thesis involves tracking image patches by correlation methods, and finding trajectories which are processed with the Panoramic Hough Transform

```
//id x y z theta
0 102.786987 -90.691299 -60.299999 236.378403
1 103.706131 -96.184898 -60.299999 236.304672
2 102.985825 -100.826057 -60.299999 237.564697
3 104.008217 -105.129364 -60.299999 236.788528
4 103.670349 -108.898079 -60.299999 236.739151
5 104.261948 -114.253136 -60.299999 236.843323
6 104.150169 -119.086388 -60.299999 237.184586
7 103.785194 -125.454506 -60.299999 236.552673
8 103.760239 -129.731247 -60.299999 236.709518
9 104.210312 -134.817642 -60.299999 236.933655
10 103.120705 -139.775742 -60.299999 237.284576
11 104.653999 -144.377548 -60.299999 235.690567
```

Figure A.5: *Except from a .POS file.*

to identify horizontal trajectories at different distances and locations relative to a translating single-lobed Panoramic camera.

A .FLOW file is a binary file containing image flow differences between two .PGM images. Direction and distance information for each pixel is transferred in a binary format. For convenience, this is itself a .PGM format file, and so can be viewed with *SEE.C* if one knows how to interpret the values.

A list of N PGM files has N-1 .FLOW files, which are listed in a .FLOW_LIST file, which is used to find individual trajectories, stored in a .TRAJ file. A section of a .TRAJ file is shown below in Fig. A.6. A trajectory consists of a series of lines ending in *END*, each line contains the frame number and the U, V coordinates. This represents where a pixel is believed to travel in each frame through the sequence.

A.0.11 General Setup Files

The geometry for a single or bi-lobed catadioptric system with spherical mirrors can is defined in a .GEOMAIN file by the user. This also contains sensor and lobe names for later reference handles. The .GEOMAIN file used for the bi-lobed catadioptric prototype sensor built for the stereo reconstruction experiments is shown below in Table A.4.

An enviroment for 3D reconstruction or robot localization can be created by running the *MAKE_SCRIPTS.C* and *MAKE_SCRIPTS_TRACK.C* programs with the .GEOMAIN file as an input.

The setup files also produce .CFG and .GEO files. The .CFG files simple list the .LOOKUP and input .PGM file, whereas the .GEO file communicates the geometry for a single lobe, and is used to create the .LOOKUP table, and reconstruct 3D points with the ideal spherical mirror reconstruction program *PANO_MATCH.C*. A sample .GEO file is

```
6 183 115
7 177 118
8 171 122
9 164 128
10 156 135
11 147 145
end
0 206 106
1 203 107
2 200 108
3 197 109
4 193 110
5 189 112
6 184 114
7 179 118
8 173 122
9 166 127
10 159 135
11 150 145
12 140 157
end
0 207 106
1 204 107
2 201 108
3 197 109
```

Figure A.6: *A .TRAJ file contains many image flow trajectories. Each trajectory is a list of frame numbers and (U,V) positions. Trajectories are separated by* END.

shown below in Table A.5.

A.1 User Programs

The following programs were not part of the vision processing, but needed to view the input image files and final output 3D polygon files. The .SCENE 3d models, and intermediate files (.FEA,.VTX files) were ASCII text and so were viewable in a standard text editor. However these files can be examined in a visual format with *FVE.C*, *PANO_DRAW.C* and *PANO_ANNOTATE.C*.

PANO_DRAW.C creates a .PGM file from a .FEA file, allowing one to view the results of feature extraction graphically. *SEE.C* can be used on several stages of feature processing to verify operation and debug problems. For example quite often the author would display the original image, the output of the initial feature detection, the result of feature tracking and the result of vertical rectangle location all at once, permitting zooming in and examination of edges and corners with the locked cursor feature. *MOVIE.C* is used for examining image sequences composed of .PGM files.

PANO_ANNOTATE.C overlays brackets and numbers on an input .PGM file from a .VTX file, useful in debugging the landmark tracking in the mobile robot localization experiments.

```
//Thur, Jan 18 calibrated Canon D-30
sensor_name= "first"
image_size = 2160
camera_y 53.9
fpixel = 6050

lobe
name="inner"
pos=0,0,5.2
radius=1.95
min_radius = 83
max_radius = 209
mirror_centerx = 1061
mirror_centery = 761
/lobe

lobe
name="outer"
pos=0,0,0
radius=5.71
min_radius = 262
max_radius = 564
mirror_centerx = 1061
mirror_centery = 751
/lobe
```

Table A.4: *First three sections of* TRACK_VTX_RUN.BAT *processing an image sequence called A_FIX.PGM_LIST.*

FVE.C allows visualization of the 3D model .SCENE files in an interactive 3D viewer. *FVE.C* and *SEE.C* both use a custom library of functions for line drawing, text overlaying, etc and thus only basic I/O and image blitting functions need to be rewritten to port entire programs between operating systems, as was done with these two programs which were originally written in DOS.

A.1.1 .PGM File Viewer

The author was unable to find a proper image viewing program that satisfied all expected needs, and so an image viewing program *SEE.C* was written. It was originally written for DOS, but ported to X windows so it can be run on LINUX and UNIX stations.

SEE.C has a cursor that can be moved with mouse or keyboard, displays the X,Y coordinates and greyscale at the location of the cursor, allows zoom and screen capture. A neighbourhood window allows all pixel intensity values in a 9 x 9 neighbourhood to be displayed.

The program can view from one to four images simultaneously, with the option to lock the cursors in all four images. The combination of locking, zooming and neighbourhood

```
//mirror assembly .geo file created by make_scripts.c from /em feb20_02.geomain
//outer
//
mirror_radius 5.710000
camera_loc 53.900002
focal 6050
//
filename first_outer_lobe
min_radius 262
max_radius 564
num_lines 302
mirror_centerx 1061
mirror_centery 744
angle_step 2
image_width 2160
image_height 2160
height_diff_thresh 0.3
```

Table A.5: *First three sections of* TRACK_VTX_RUN.BAT *processing an image sequence called A_FIX.PGM_LIST.*

viewing allows operations performed by image processing programs to be analyzed on a pixel level. Fig. A.7 (top) shows four images viewed simultaneously, in this case showing stages of feature extraction for a perspective view camera (the initial research direction). Fig. A.7 (bottom) shows two images testing median filtering for edges, zoomed in with locked cursors and neighbourhood viewing turned on.

A.1.2 *MOVIE.C* Image Sequence Viewer

A custom made image sequence viewing program was written, which allows analysis of one or more image sequences on a frame by frame basis. The filename of each image in a window would be shown, and the user could advance forward or backward through the sequence one frame at a time, or play them all in a fast or slow "movie" mode.

The images were listed in a .PGM_LIST file, an ASCII file with one filename per line. Only .PGM files could be viewed. One quick way to view a directory of PGM files could be to redirect the *ls* command to a file. The file could then be sorted with *ALPHASORT.C*, another program written by the author.

Several sequences could be viewed at once, with several windows all showing a .PGM file from their sequence. All windows would be at the same frame number in their respective sequence, allowing the viewing of input and output image sequences.

For example, typing *movie in.pgm_list in_line.pgm_list in_pos.pgm_list* at the UNIX or LINUX prompt would open three windows, which in this example represent the image sequences from the mobile robot localization system. *in.pgm_list* was the input sequence from

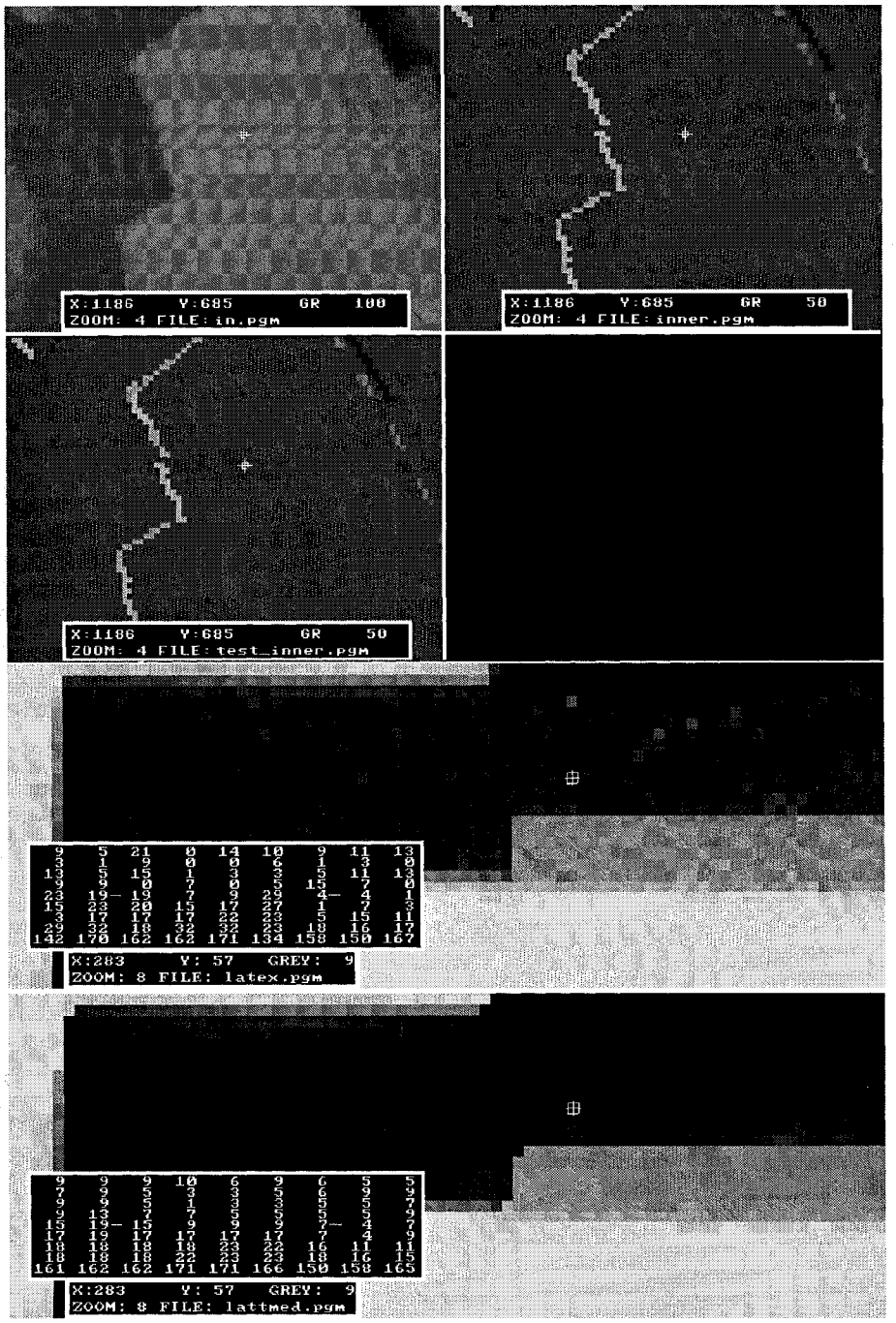


Figure A.7: SEE.C *PGM image viewing program. (Top) Three images loaded with locked cursors. (Bottom) Two images zoomed in with locked cursors and neighbourhood pixel viewing mode on.*

the camera, *in_line.pgm_list* was the annotated image sequence with tracked line segment projections and their junctions marked, and *in_pos.pgm_list* was a top map view showing the triangulation of landmarks.

A.1.3 *FVE.C*: SCENE File Viewer

As that the original thesis direction was to develop a computer vision system with standard narrow field-of-view perspective cameras, a comprehensive environment was built to both create synthetic views, and to edit and view three dimensional polygons. *FVE.C* is a viewer for .SCENE files[1]. Polygons could be defined in the .FVE language defined in the above section, and assigned a colour or texture map, and an optional normal vector. The images below (Fig. A.8 demonstrate the multiple viewports, 3D cursor and object manipulation). Virtual cameras can be created, along with radial distortion and a scale factor to simulate real world cameras.

Perspective transformations and texture mapping were the methods used, so when the choice was made to switch directions to panoramic viewing with curved mirrors, it was clear this program could not be used for more than editing and viewing 3D polygon files, and so the public domain program ray-tracing *Povray* was used. A conversion utility *FVE2POV.C* was written to convert 3D polygons from FVE format to that expected by *Povray.*

The 3D models shown in Chapter 8 were created by using the screen capture facility of *FVE.C.* The world models for the robot were partially edited in FVE with the 3D cursor capability.

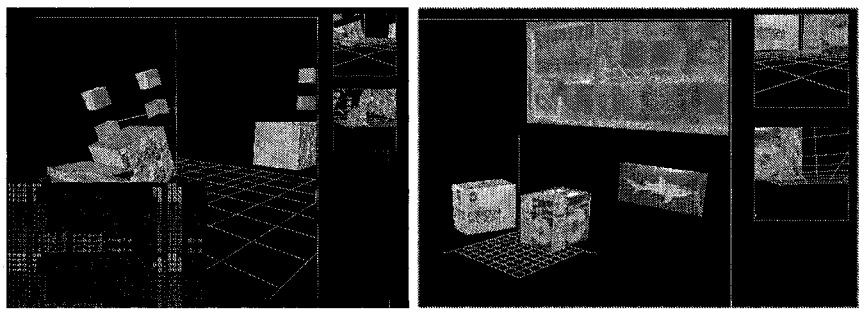


Figure A.8: FVE.C *.FVE 3D model viewing/editing program. (Left) Texture mapped graphics, with 3D cursor and object position pop-up menu. (Right) System used to view reconstructed model of objects from a stereo experiment.*

[1]FVE stands for *Fiala Vision Environment*

A.1.4 *PANO_DRAW.C* .FEA File Converter

.FEA feature files can be visualized with *PANO_DRAW.C* as shown in Fig. A.9. The program produces a .PGM file which can be viewed concurrently with other image files to inspect the progress at the various stages.

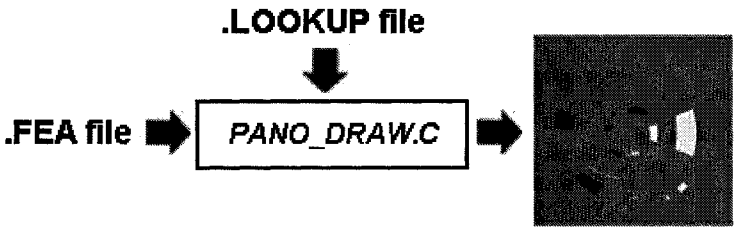


Figure A.9: PANO_DRAW.C *Creates PGM file (for viewing with* SEE.C *from a feature file. Line segments are drawn as white or black, depending on the edge polarity they represent. Vertical rectangle projections are also drawn.*

A.1.5 *ANNOTATE_PANO.C* .VTX File Converter

.VTX vertex feature files can be visualized by overlaying them over a PGM file, typically one would use the image from which the vertices were tracked, as is shown below in Fig. A.10 for frame 20 of the real image sequence from the mobile robot localization experiments.

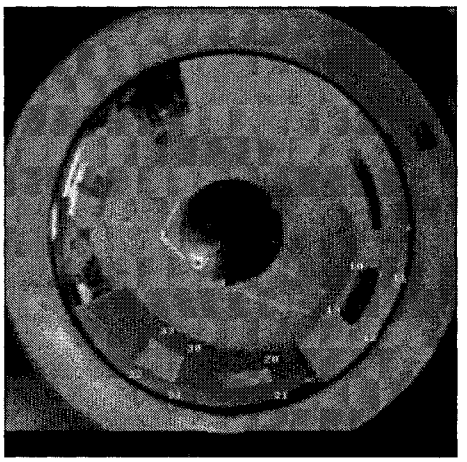


Figure A.10: ANNOTATE_PANO.C *Overlays information from a .VTX file onto a .PGM file.*

A.2 Vision Processing Programs

The fundamental programs used in this thesis are the pre-processing program *PANO_PREPROCESS.C*, the feature detection program *PANO_SEG.C* and the feature verification-tracking program *PANO_TRACK_LINE.C* The following few sections demonstrate the operation of the vision processing stages, viewed as black boxes inputing and outputting image and text files.

A.2.1 *PANO_PREPROCESS.C* Pre-Processing

PANO_PREPROCESS.C was used for the noisy images captured from the NTSC camera, performing a smoothing operation with a 3x3 averaging filter, followed by aspect ratio adjustment. this was not necessary for the imagery from the Canon D-30 digital camera as that the image was very low noise, and the pixels were square. Fig. A.11 demonstrates the operation on an image from the localization image sequence.

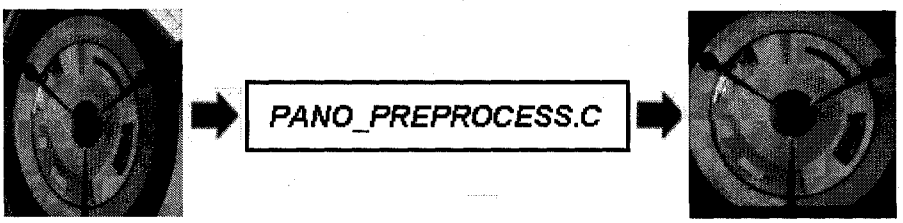


Figure A.11: PANO_PREPROCESS.C *smoothes noise and adjusts aspect ratio.*

PANO_SEG.C Feature Detection

PANO_SEG.C is the main feature extraction program used in the 3D reconstruction. It takes a .CFG files as an argument, which indicates which .PGM file to load (usually IN.PGM), the name of the lookup table to use (there are at least two in the directory for a stereo bi-lobed system) and some optional settings such as minimum length, etc. The output is a .FEA file, which contains the output of feature detection. Fig. A.12 demonstrates how a .CFG file, a .LOOKUP file and an input image are input to *PANO_SEG.C*, and a feature file is output.

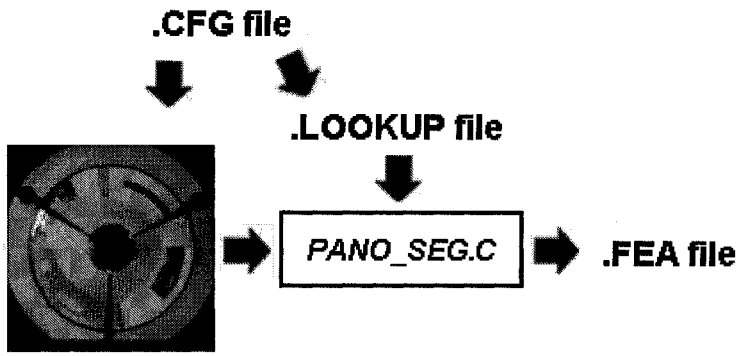


Figure A.12: PANO_SEG.C *searches for line segment projection features in an image according to the Panoramic Hough Transform data in the .LOOKUP file.*

***PANO_TRACK_LINE.C* Feature Verification**

PANO_TRACK_LINE.C performs the verification stage in 3D reconstruction, and the line segment tracking function in the robot localization (Fig. A.13). The image is searched for the presence of the features listed in the .FEA file, and the .FEA file is updated with only located features listed, and their new updated position parameters. This program is run with an angular and radial search range.

PANO_TRACK_LINE.C is the only run-time program used by both the stereo reconstruction and mobile robot localization applications.

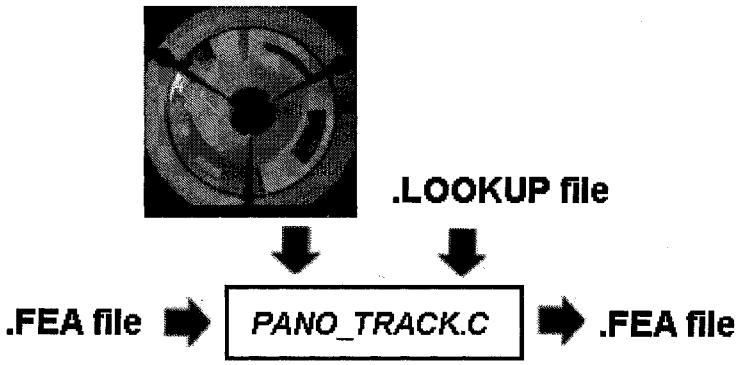


Figure A.13: PANO_TRACK_LINE.C *verifies and fine-tunes line segment projection features according to an image.*

A.2.2 Segment Projection Feature Processing

A family of programs work within the abstraction level of the segment projection features, each inputing a .FEA file, and outputting a .FEA file. These do not use any image pixel

information as does *PANO_TRACK_LINE.C*, they only make inferences and hypotheses from the input .FEA file(s). All these programs' output .FEA files are later processed by a verification pass with *PANO_TRACK_LINE.C*.

COMBINE_FEA.C takes in a list of .FEA files, and combines horizontal projection primitives that appear to meet. This was created to join the feature files created by dividing the input image into angular slices (as described in Chapter 5).

PANO_VRECT_HYPOTHESIZE.C attempts to repair the effects of *false negatives* (where segment projections were not recognized that did exist in the image). Heuristics look for segment features that suggest a vertical rectangle projection but do not have a full closed set of edges. If three sides are found, the fourth is added to the output .FEA list. Likewise if two parallel horizontal line projections have similar endpoints, but no vertical projection segments joining them, two hypothetical vertical segment features are added. This program's operations, especially, should be followed by *PANO_TRACK_LINE.C* to verify these

PANO_PURGE.C prunes out short segment projections that do not meet in a corner with another segment, as that they are assumed to have valuable meaning to the higher level model. Either they are the result of the feature extraction system trying to model some noise or non-edge feature, or the image feature is too small to be useful.

A.3 Segment Projection to Polygon Projection Feature Processing

PANO_VRECT.C is the final stage in the feature extraction process, it takes the input .FEA file, and attempts to find horizontal and vertical line segment projections which meet and form a closed shape. A lighter polygon projection would be surrounded by two horizontal and two vertical segment projections whose edges had the intensity derivative positive facing into the region. Likewise heuristics look for dark polygons by finding four segments whose darker sides are inside this region.

Since the feature endpoints will likely not meet perfectly, a decision has to be made whether four segments do define a region consistent with the projection of a vertical polygon face. This is accomplished with a combination of examining the confidence levels of the features (all features have a corresponding confidence level) and ad hoc thresholds.

This program is the only one in 3D stereo reconstruction whose output is not verified by comparison with the input image This is a weakness of the system, but not fatal since the error probabilities of generating a false 3D polygon is dependent on two such matching errors in both lobes.

A.3.1 *PANO_MATCH.C* 3D Scene Generation

The final stage in creating a 3D model of what the panoramic sensor sees is to match the features and triangulate their 3D location. This thesis does not focus much attention matching, and for each primitive type simply orders them according to radius, and then matches the list from each lobe. This will not account for occlusion or features visible in only one view, but matching strategies are a problem to all stereo vision and other works focus on this.

PANO_MATCH.C takes two .FEA files, one from each lobe, and a .GEO file from each lobe to triangulate on a 3D location (Fig. A.14). *PANO_MATCH2.C* is a variation that loads two lookup calibration lookup tables, each a height vs radius lookup. Two calibration lookup tables are loaded for two depth values, for each lobe for a total of four files (Fig. A.15). This empirically calibration is described in Chapter 7 and provides much better 3D accuracy due to the mirror imperfections.

The output polygons are written to an output .SCENE file.

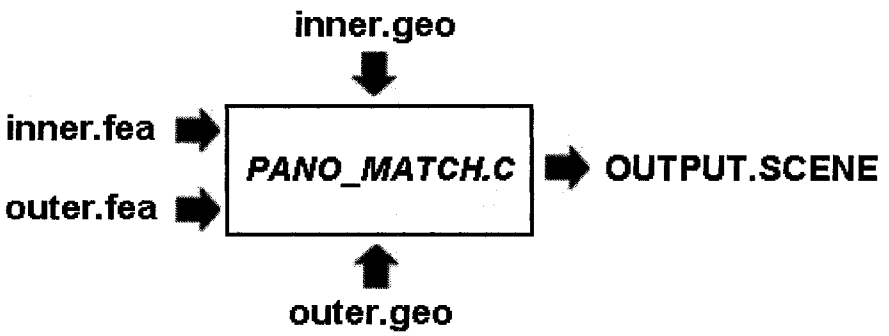


Figure A.14: PANO_MATCH.C *creates a 3D model from two .FEA feature files and two ideal sphere model .GEO files.*

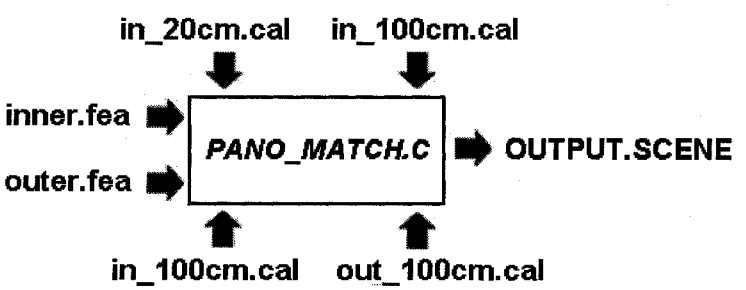


Figure A.15: PANO_MATCH2.C *creates a 3D model from two .FEA feature files and four empirical calibration files.*

A.3.2 *SCENE_ERROR.C* 3D Scene Analysis

In order to numerically assess the accuracy of the reconstruction, a comparison needs to be made against a ground truth model. The vertical rectangles in the environment were measured and a CORRECT.SCENE file created manually.

An analysis program, *SCENE_ERROR.C* loads in two .SCENE files, one from the reconstruction, and the CORRECT.SCENE file and matches the points (since 3D points are typically erroneous only in distance). Statistics on error are aggregated and output in .PGM graph files.

An output SCENE_ERROR.SCENE file is created that shows the correct and reconstructed polygons in white and grey respectively which can be viewed with *FVE.C*. Screen captures from *FVE*, and error graphs produced by *SCENE_ERROR.C* are what was shown in Chapter 8 on stereo reconstruction.

A.3.3 Landmark Tracking Programs

The programs *PANO_LANDMARK_EXTRACT.C*, *VTX_TRACK_MASK.C*, *PANO_TRACK_LINE.C*, *PANO_BLINDCHECK.C*, *LINE2VTX.C* and *PANO_TRACK_VTX.C* are the programs special to the mobile robot localization application. Two batch files are created by *MAKE_SCRIPTS_TRACK.C*, one for each method.

PANO_LANDMARK_EXTRACT.C creates a .VTX and .FEA file from the 3D .SCENE model and the current position (encoded in the last line of a .POS file) for use by either *VTX_TRACK_MASK.C* or *PANO_TRACK_LINE.C*.

PANO_BLINDCHECK.C is used where there are occluding supports, if a tracking program puts a vertex feature in the region defined as a permanent occlusion, it is removed so as not to give a false landmark-feature match for the localization.

PANO_TRACK_VTX.C performs the actual triangulation from a .VTX file and the landmark location .PANO_VTX_TRACK file, and updates the .POS file.

A.4 Stereo Reconstruction Vision Processing

The batch file *RUN.BAT* that runs all the programs on an input PGM image is listed below in Table A.6, the individual programs are described above. The C programs are all command line driven. *RUN.BAT* also contains standard UNIX commands like copy and delete, and programs such as *PANO_DRAW* that are not essential to the operation but

useful for analysis and debugging.

RUN.BAT (Table A.6) was created by *MAKE_SCRIPTS.C* which all produced all .GEO and .CFG files, and a *PREPARE.BAT* batch file to create the .LOOKUP file.

A.5 Vertex Tracking Vision Processing

The batch file *TRACK_VTX_RUN.BAT* (Table A.8) performs the mobile robot localization using the first method, that of tracking vertex corner projections in the quasi-cylindrical warped image. The position is entered as a line in the .POS file, which grows with a new line as every image in the .PGM_LIST file is processed. The .POS file is started with an initial position.

PANO_LANDMARK_EXTRACT.C creates a .VTX list of predicted vertices according to the WORLD.SCENE file and the estimated position which is the last entry in the .POS file. *VTX_TRACK_MASK.C* is used to track the vertices, it creates the quasi-cylindrical warp, and outputs a new .VTX file after matching. *PANO_BLINDCHECK.C* filters the list to remove vertices that landed in an occluding region from a support. *PANO_TRACK_VTX.C* uses the .VTX file and the constant PANO_VTX_TRACK file to update the position and place a new line entry in the .POS file.

These programs are run in succession for every image. The batch file is created by *MAKE_SCRIPTS_TRACK.C* from a .GEOMAIN file and an *a priori* .SCENE file, with an initial start position. *MAKE_SCRIPTS_TRACK.C* is given the .GEOMAIN file, a .SCENE file, a start position, the name of an image sequence (A_FIX.PGM_LIST in the example below) and the name for the experiment (Oct_5 in the example). From this the batch file *TRACK_VTX_RUN.BAT*, the initial .POS file with the start position and the .PANO_VTX_TRACK file is created.

A.6 Line Junction Tracking Vision Processing

This is the second panoramic tracking approach, which uses the Panoramic Hough Transform to track line segment projections instead of corners. No quasi-cylindrical image is created as in the previous method, the tracking is done directly on the panoramic image.

PANO_TRACK_LINE.C is used as the functional program instead of *VTX_TRACK_MASK.C*. *LINE2VTX.C* is used to convert the .FEA file to a .VTX file so that the *PANO_TRACK_VTX.C* program could be used without modification. *LINE2VTX.C* is responsible for recognizing the intersection of correct features to declare vertex primitives.

PANO_LANDMARK_EXTRACT.C creates a .FEA file as well as a .VTX list, either one or the other is ignored depending on which method is used.

The first part of a batch file *TRACK_LINE_RUN.BAT* is shown below in Table A.9. The setup is performed by *MAKE_SCRIPTS_TRACK.C* as with the vertex tracking method. In fact everything is set up for both methods, and one can choose which batch file to run.

A.7 Optic Flow Vision Processing

The Shape-Through-Motion application of the PHT was not developed as far, basic proof of concept work only was done.

OPTIC.C compares two images by correlating image patches of various size between them, and generates statistics of the correlation. The match distribution is expressed as a distribution center and a description of the spread. The spread is define a major and minor axis length, with the angle of the major axis specified. Due to the *aperture problem*, corners, edges and uniform regions will all match different distributions/confidence levels. See Chapter 10 for more information on correlation matching . Five values: an I and J offset, a major axis angle, and the major and minor axis lengths are reported for each pixel (and it's surrounding neighbourhood) in a .FLOW file. This output information is transmitted via a .FLOW file, which is really a .PGM format file (a kludge for convenience).

This correlation matching was done in a brute force fashion for this thesis, taking hours to calculate each .FLOW file, but as was mentioned this development was a proof of concept, not an optimized complete system as the stereo reconstruction and mobile robot localization systems are.

ITERATE.C attempts to correct the aperture problem by propagating optic flow from pixels with a defined flow to neighbours with less determinate image motion. For example a corner would give a well defined flow, represented by a small major and minor axis. An edge has a defined flow normal to the edge only so *ITERATE.C* would try to orient the flow parallel to that of the nearby corner. *ITERATE.C* takes a .FLOW file as an input and outputs another .FLOW file. The .FLOW files are listed in a .FLOW_LIST file for use by the next stage.

TRACK_FLOW.C extracts a list of trajectories from the the .FLOW files listed in the .FLOW_LIST file. This is done by starting at a seed point in the first frame, and following a chain of high confidence flow vectors. A trajectory ends if it lands in an area of low flow direction confidence in a given frame. Trajectories below a threshold length are purged. The resultant trajectories are listed in the ASCII .TRAJ file.

PH_TRAJ.C applies the Panoramic Hough Transform data encoded in a .LOOKUP file

and extracts trajectory sections consistent with a horizontal 3D line. For ones that satisfy this, a point is added to a .SCENE file. Note that the 3D structure is not recovered, just a model correct to a scale factor. With further development and a known camera trajectory, the 3D model could have a correct scale.

The whole process is shown in Fig. A.16 from PGM file sequence to 3D model. The image sequence is input to the system as PGM files, and a PGM_LIST file listing them. *OPTIC.C* operates on each subsequent pair producing a .FLOW file. *ITERATE.C* acts on each .FLOW file.

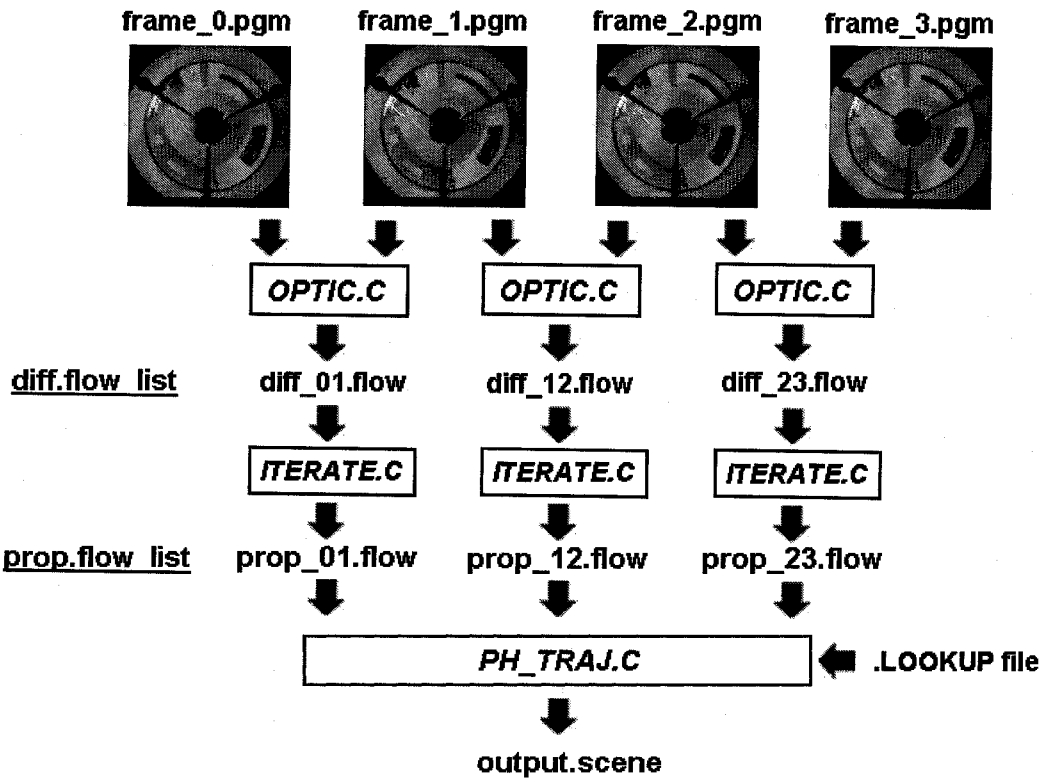


Figure A.16: *Stages of the Shape-Through-Motion application of the PHT in this thesis.*

A.8 Setup Programs

The whole environment for the stereo reconstruction and mobile robot localization vision systems is highly automated. Batch files run the operations which are divided into separate programs for development and analysis purposes.

These "run-time" batch files, plus other batch files for preparation and synthetic image and synthetic image sequence generation are created by the setup programs *MAKE_SCRIPTS.C*

and *MAKE_SCRIPTS_TRACK.C*. The former sets up the stereo reconstruction environment, the latter sets up the robot localization environment.

Both programs also set up other files, such as the .GEO and .CFG files needed, as well as a preparatory script PREPARE.BAT which creates the .LOOKUP file needed by programs that implement the Panoramic Hough Transform. Both setup programs also create .POV and batch files to run the freeware *Povray* ray tracing program to generate synthetic imagery.

```
rm -f oin*.fea
pano_seg3 first_inner_lobe.cfg 0 44
cp in.fea oin0.fea
pano_seg3 first_inner_lobe.cfg 45 89 cp in.fea oin1.fea pano_seg3 first_inner_lobe.cfg 90 134
cp in.fea oin2.fea
pano_seg3 first_inner_lobe.cfg 135 179
cp in.fea oin3.fea
pano_seg3 first_inner_lobe.cfg 180 224
cp in.fea oin4.fea
pano_seg3 first_inner_lobe.cfg 225 269
cp in.fea oin5.fea
pano_seg3 first_inner_lobe.cfg 270 314
cp in.fea oin6.fea
pano_seg3 first_inner_lobe.cfg 315 359
cp in.fea oin7.fea
echo ———
rm -f thu_in*.fea
rm -f inner.fea
rm -f oct_in*.pgm
combine_fea_inner oin0.fea oin1.fea oin2.fea oin3.fea oin4.fea
oin5.fea oin6.fea oin7.fea thu_in.fea
pano_draw thu_in.fea oct_in.pgm
pano_track_line_inner thu_in.fea in.pgm thu_in_tr1.fea 4 10
pano_vrect_hypothesize thu_in_tr1.fea thu_in_tr2.fea
pano_track_line_inner thu_in_tr2.fea in.pgm thu_in_tr.fea 4 10
pano_draw thu_in_tr.fea oct_in_tr.pgm
pano_vrect_inner thu_in_tr.fea thu_in_tr_v1.fea
pano_fea_purge thu_in_tr_v1.fea thu_in_tr_v.fea
pano_draw thu_in_tr_v.fea oct_in_tr_v.pgm
cp -f thu_in_tr_v.fea inner.fea
echo see in.pgm oct_in.pgm oct_in_tr.pgm oct_in_tr_v.pgm
echo ——==========————
echo ——==== DONE INNER ====——
echo ——==========————

rm -f sout*.fea
pano_seg3 first_outer_lobe.cfg 0 22
cp in.fea sout0.fea
pano_seg3 first_outer_lobe.cfg 23 44
cp in.fea sout1.fea
pano_seg3 first_outer_lobe.cfg 45 67
cp in.fea sout2.fea
pano_seg3 first_outer_lobe.cfg 68 89
cp in.fea sout3.fea
pano_seg3 first_outer_lobe.cfg 90 112
cp in.fea sout4.fea
```

Table A.6: RUN.BAT *batch file for stereo reconstruction (continued in Table. A.7).*

```
pano_seg3 first_outer_lobe.cfg 113 134
cp in.fea sout5.fea
pano_seg3 first_outer_lobe.cfg 135 157
cp in.fea sout6.fea
pano_seg3 first_outer_lobe.cfg 157 179
cp in.fea sout7.fea
pano_seg3 first_outer_lobe.cfg 180 202
cp in.fea sout8.fea
pano_seg3 first_outer_lobe.cfg 203 224
cp in.fea sout9.fea
pano_seg3 first_outer_lobe.cfg 225 247
cp in.fea sout10.fea
pano_seg3 first_outer_lobe.cfg 248 269
cp in.fea sout11.fea
pano_seg3 first_outer_lobe.cfg 270 292
cp in.fea sout12.fea
pano_seg3 first_outer_lobe.cfg 293 314
cp in.fea sout13.fea
pano_seg3 first_outer_lobe.cfg 315 337
cp in.fea sout14.fea
pano_seg3 first_outer_lobe.cfg 338 359
cp in.fea sout15.fea
echo ———
rm -f slices_out*.fea
rm -f outer.fea
rm -f slices_out*.pgm
combine_fea sout0.fea sout1.fea sout2.fea sout3.fea sout4.fea sout5.fea sout6.fea
sout7.fea slices_outer.fea
combine_fea sout8.fea sout9.fea sout10.fea sout11.fea sout12.fea sout13.fea sout14.fea
sout15.fea slices_outer.fea slices_outer.fea
pano_draw slices_outer.fea slices_out.pgm
pano_track_line slices_outer.fea in.pgm slices_out_tr1.fea 5 25
pano_track_line slices_out_tr1.fea in.pgm slices_out_tr2.fea 4 20
pano_track_line slices_out_tr2.fea in.pgm slices_out_tr3.fea 3 15
pano_vrect_hypothesize slices_out_tr3.fea slices_out_tr4.fea
pano_track_line slices_out_tr4.fea in.pgm slices_out_tr5.fea 4 20
pano_draw slices_out_tr5.fea slices_out_tr.pgm
pano_vrect slices_out_tr5.fea slices_out_tr_v1.fea
pano_fea_purge slices_out_tr_v1.fea slices_out_tr_v.fea
pano_draw slices_out_tr_v.fea slices_out_tr_v.pgm
cp -f slices_out_tr_v.fea outer.fea
echo see in.pgm slices_out.pgm slices_out_tr.pgm slices_out_tr_v.pgm

echo ——====================——
echo ——==== DONE OUTER ====——
echo ——====================——

pano_match2a inner.fea inner.geo outer.fea outer.geo
```

Table A.7: RUN.BAT *batch file for stereo reconstruction.*

```
pano_track_vtx_initial base.scene oct5_initial.pano_vtx_track
rm oct5.pos
echo ——— a_fix_0 ———
pano_landmark_extract single_only_lobe.cfg oct3.geomain only base.scene initial.pos
vtx_track_mask pano_landmark_extract.vtx a_fix_0.pgm a_fix_0.vtx ideal 30 10 annotate
cp vtx_track_mask_cyl.pgm a_fix_0_cyl.pgm
cp vtx_track_mask_ccd.pgm a_fix_0_ccd.pgm
pano_blindcheck_vtx a_fix_0.vtx initial.pos oct5_initial.pano_vtx_track a_fix_0.vtx
pano_track_vtx oct5_initial.pano_vtx_track a_fix_0.vtx oct5.pos
cp pano_track_vtx.pgm a_fix_0_pos.pgm
echo ——— a_fix_1 ———
pano_landmark_extract single_only_lobe.cfg oct3.geomain only base.scene oct5.pos
vtx_track_mask pano_landmark_extract.vtx a_fix_1.pgm a_fix_1.vtx ideal 30 10 annotate
cp vtx_track_mask_cyl.pgm a_fix_1_cyl.pgm
cp vtx_track_mask_ccd.pgm a_fix_1_ccd.pgm
pano_blindcheck_vtx a_fix_1.vtx oct5.pos oct5_initial.pano_vtx_track a_fix_1.vtx
pano_track_vtx oct5_initial.pano_vtx_track a_fix_1.vtx oct5.pos
cp pano_track_vtx.pgm a_fix_1_pos.pgm
echo ——— a_fix_2 ———
pano_landmark_extract single_only_lobe.cfg oct3.geomain only base.scene oct5.pos
vtx_track_mask pano_landmark_extract.vtx a_fix_2.pgm a_fix_2.vtx ideal 30 10 annotate
cp vtx_track_mask_cyl.pgm a_fix_2_cyl.pgm
cp vtx_track_mask_ccd.pgm a_fix_2_ccd.pgm
pano_blindcheck_vtx a_fix_2.vtx oct5.pos oct5_initial.pano_vtx_track a_fix_2.vtx
pano_track_vtx oct5_initial.pano_vtx_track a_fix_2.vtx oct5.pos
cp pano_track_vtx.pgm a_fix_2_pos.pgm
```

Table A.8: *First three sections of* TRACK_VTX_RUN.BAT *processing an image sequence called A_FIX.PGM_LIST.*

```
pano_track_vtx_initial base.scene tue_initial.pano_vtx_track
rm tue.pos
echo ——— a_fix_0 ———
pano_landmark_extract single_only_lobe.cfg oct3.geomain only base.scene initial.pos
pano_track_line pano_landmark_extract.fea a_fix_0.pgm a_fix_0.fea 15 8
pano_track_line a_fix_0.fea a_fix_0.pgm a_fix_0.fea 3 3
line2vtx a_fix_0.fea a_fix_0.vtx
annotate_pano a_fix_0.vtx a_fix_0_line.pgm
pano_blindcheck_vtx a_fix_0.vtx initial.pos tue_initial.pano_vtx_track a_fix_0.vtx
pano_track_vtx tue_initial.pano_vtx_track a_fix_0.vtx tue.pos
cp pano_track_vtx.pgm a_fix_0_pos.pgm
echo ——— a_fix_1 ———
pano_landmark_extract single_only_lobe.cfg oct3.geomain only base.scene tue.pos
pano_track_line pano_landmark_extract.fea a_fix_1.pgm a_fix_1.fea 15 8
pano_track_line a_fix_1.fea a_fix_1.pgm a_fix_1.fea 3 3
line2vtx a_fix_1.fea a_fix_1.vtx
annotate_pano a_fix_1.vtx a_fix_1_line.pgm
pano_blindcheck_vtx a_fix_1.vtx tue.pos tue_initial.pano_vtx_track a_fix_1.vtx
pano_track_vtx tue_initial.pano_vtx_track a_fix_1.vtx tue.pos
cp pano_track_vtx.pgm a_fix_1_pos.pgm
echo ——— a_fix_2 ———
pano_landmark_extract single_only_lobe.cfg oct3.geomain only base.scene tue.pos
pano_track_line pano_landmark_extract.fea a_fix_2.pgm a_fix_2.fea 15 8
pano_track_line a_fix_2.fea a_fix_2.pgm a_fix_2.fea 3 3
line2vtx a_fix_2.fea a_fix_2.vtx
annotate_pano a_fix_2.vtx a_fix_2_line.pgm
pano_blindcheck_vtx a_fix_2.vtx tue.pos tue_initial.pano_vtx_track a_fix_2.vtx
pano_track_vtx tue_initial.pano_vtx_track a_fix_2.vtx tue.pos
cp pano_track_vtx.pgm a_fix_2_pos.pgm
```

Table A.9: *First three sections of* TRACK_LINE_RUN.BAT *processing an image sequence called A_FIX.PGM_LIST.*

Appendix B

Edge Detection

Step intensity changes, a.k.a. edges, are a salient image feature to detect for vision systems assuming a polyhedral world. Edge detection is typically one of the first steps used in image segmentation. This appendix is a brief review of edge detection techniques, and attempts to explain and unify the convolution-based approaches.

Edge detection is a low level task, usually done after some pre-processing but before line extraction. Edge detection seeks to find pixels that correspond to object boundaries in an image. Typically the greyscale image is converted to a binary image indicating edge existence, with each '1' pixel sometimes accompanied with an edge strength and/or direction. These extracted pixel locations, mini-edges are called *edgels*. The terms edge detectors and edge operators are used interchangeably below.

The boundary of an object is usually recognized by a large, sudden change in intensity within the span of a few pixels. An ideal edge, the one for which much of the edge operators are designed, is a step edge with constant intensity on either side. Of course many edges are not a step discontinuity due to both object corners not being sharp and imperfections in the image capture process. Edges caused by round objects are typically graded, and even polyhedral objects in the real world rarely have knife-sharp corners. Even if the incident light that a perfect ideal camera would see as a step, the blurring caused by the lens (point-spread function) and the sampling of the image irradiance on the imaging plane will typically spread this discontinuity out to more than a pixel wide.

Edge detectors are classically used not to define the existence of lines or boundary curves themselves, but to provide information on where an image edge may be, it is usually the first stage in extracting lines with the *edge aggregation* stage making decisions on where a true line or curved boundary lies by considering many of these edgels, deciding on a best fit and rejecting outliers.

Edge detectors usually only look within a small region, and so do not see the 'big picture' and so are sensitive to noise and small variations that do not correspond with a prominent feature that we are interested in finding. Researchers have tried edge detectors that consider different sizes of an image section, and found as one might expect, larger windows give better immunity to noise (less false edgels) for image features equal or larger to the size of the window, but are more computationally expensive and give less precise edge position confidence (see Nalwa [80] Chapter 2).

Edge detectors have been divided into two main categories, *Difference* and *Parametric* edge detectors. Difference operators work on the principle of differentiation, image discontinuities can be detected by detecting a large spatial derivative. Parametric or *model-matching* edge detectors as they are also called, attempt to fit a neighbourhood of pixels to one or more prescribed edge models, and decide if the center pixel qualifies as an edgel. (Nalwa [80] Chapter 2) claims difference detectors typically fail by giving too many false positives

(falsely indicating an edgel where one is not there) where as parametric detectors tend to err on the side of not recognizing an edgel (so-called false negatives).

Difference edge detectors are typically more suited to an FIR (Finite Impulse Response) operation, convolving the image with a NxM mask (especially the ones that do not use normalization). This reduces the computation complexity and time and is much more amenable to real-time hardware implementation, and the maintenance of a constant rate of data flow through the lower stages of a vision system.

Difference operators can be divided into first-derivative, second-derivative and combination first and second-derivative operators. The first-derivative operators can be further divided into linear and non-linear (involving a division) operators.

B.1 First-Derivative Difference Operators

B.1.1 Linear

Difference operators use subtraction of image intensities in a neighbourhood to estimate the derivative. This is usually done is typically two or more separate operators, usually two to correspond to finding edges in the two image dimensions.

These operators take an image in the form 'analog-like' (high number of level gradations, such as 64, 256 or 1024) two-dimensional arrays and provide a similar 'analog-like' image as a response. This can be followed with a thresholding operation to provide a binary edgel image.

The simplest edge detector pair is to just subtract a neighbouring pixel (such as the left neighbour for example) from a given pixel to estimate delta-X, and likewise vertically. We call this detector the *Neighbour detector* in this paper for lack of a universally defined name. The horizontal and vertical Neighbour detector can be expressed as convolution masks as below:

$$\begin{bmatrix} -1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{B.1}$$

Alternatively we can try restrict detected edgels as being more horizontal or vertical nature by looking at two pixels on each side, our *Double Neighbour detector*:

$$\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \tag{B.2}$$

The Robert's, or alternatively called Cross detector finds differences in diagonal pixels.

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{B.3}$$

Whenever a pair of edge detectors are used at orthogonal directions, we can calculate the magnitude by the square-root of the sum of the results squared (Euclidean), can approximate by simply summing (similar to finding the Manhattan distance) or simply take the larger of the two. The direction can be found by calculating an inverse tangent of the two responses (being careful to watch for divide-by-zero cases). This researcher used two operators for X and Y but kept the data separate until after edgel aggregation.

Because differentiation enhances noise, most difference edge operators are preceded by a smoothing filter, or have one built in. Researchers use average filters, or more sophisticated low pass filters such as the Gaussian because it is better behaved in the frequency domain and has some other nice mathematical properties. The Gaussian is a negative exponential function with the square of the radius as its exponent and produces the well-known bell curve in statistics.

Below is the 3x3 *Prewitt* edge detection mask as represented by a convolution of the Neighbourhood operator with a 3x2 averaging mask (linear convolution operations can grouped to provide an equivalent single mask):

$$\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{B.4}$$

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{B.5}$$

The Prewitt masks were originally defined as the result of finding the best least-squares error fit of a planar surface to the neighbourhood (this can be visualized by imagining the Z-direction as being image intensity and fitting a plane to describe an edge as a constant intensity ramp). If one uses this definition, one can also create Prewitt masks of different sizes. Below is a 4x4 Prewitt mask:

$$\begin{bmatrix} 3 & 1 & -1 & -3 \\ 3 & 1 & -1 & -3 \\ 3 & 1 & -1 & -3 \\ 3 & 1 & -1 & -3 \end{bmatrix} \tag{B.6}$$

Likewise the notable *Sobel* edge detector 3x3 masks can be represented as a sequential convolution of the *Double Neighbourhood* operator with a 2x2 averaging mask:

$$\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{B.7}$$

$$\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{B.8}$$

A similar mask that is reported to provide better edge angle preservation (is said to be more isotropic) is the following un-named mask pair.

$$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} \tag{B.9}$$

Also of note are the *Nevatia-Babu* edge templates [84], they are six 5 x 5 templates each at 30 degree increments rather than two and one determines the direction of an edge by observing which of the six provides the largest response, or interpolating between the two largest responses to obtain a more precise edge direction estimate. These masks are large and contain large weight numbers (up to 100), and the reader is directed to the original paper (or to Pratt [93] pg 512) to find these masks.

This leads to a secondary approach to finding edges, rather than using only two convolution masks geared towards the two principle directions and estimating the direction and magnitude from combining the results, one can make a large group of edge masks, each resembling a basic pattern rotated by some degrees between each mask. The direction of the edge can then be given by which mask gave the highest response.

Robinson suggests two families of masks, called 3-level and 5-level which are the 3x3 Prewitt and Sobel masks respectively rotated in 45 degree increments. Kirsch proposed a family of 8 masks, each also representing a 45 degree increment. The first 4 are repeated below.

$$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix}$$
$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix} \tag{B.10}$$

As mentioned earlier, the larger masks provide better immunity to noise, but lose locational confidence precision. More pixels provide a high response in the neighbourhood of an edge and so one has to perform some thinning on a binary image produced with the larger masks to produce a single edgel. This is demonstrated dramatically in an experiment in Nalwa [80] page 88-89 where the Roberts, Prewitt and Nevatia-Babu operators are tested on a synthetic image with differing levels of noise. The edge detection disappears in noise almost all together for practical automatic consideration at the higher noise levels with the smaller masks, and the larger region of detected edges is evident also.

Thresholding the output of these edge operators to produce a binary edgel image can be problematic, selecting this threshold level must depend on the noise level, the strength of edges, and the tolerance of higher processing to noisy results. This can be set as a constant after viewing many samples of the images a vision system is intended to work for. However the quality of images - illumination, the type of camera used, the qualities of objects to be observed cannot be changed too radically and still always expect good results.

One fallback of the constant threshold method is that regions of higher intensity give greater responses if the edge sizes are proportionally larger. And if noise is correspondingly larger in bright areas of images (as when the illumination is noisy, or more esoterically, in SAR satellite imagery, where noise is multiplicative), then the threshold will have to be set higher to reject this 'brighter' noise and edges in lower intensity images will be lost. Selecting a local threshold within that area of the image can introduce a bit more computation and complexity but can address this problem.

An alternate related method of edge detection is to use the same or similar masks but normalize the result, divide by some combination of the intensities and/or their variances.

B.2 Non-linear - Normalized

If the response to a linear operator is divided by some measure of the neighbourhood brightness, such as average or variance of intensity, then the noise immunity of brighter edges can be maintained while not missing edges in darker regions.

One way of looking at this is the vector angle approach usually attributed to Frei and Chen [41] is to consider the N x M neighbourhood as a N x M cartesian space and examine the projections of this neighbourhood onto vectors that describe ideal edges.

A 3 x 3 neighbourhood, for example, could be represented as a 9-dimensional space with each axis the intensity of one of the neighbourhood pixels. One can create a new basis set of 9 vectors (that span this space) and find the coords of this neighbourhood in term of this new coordinate system. This is achieved by finding the dot product (inner product) of this vector with each of the basis vectors. I.e. we can represent a neighbourhood by a linear combination of these vectors.

The Frei and Chen 9 mask set is shown below, they form a complete basis for a 9-dimensional space. Note that the first two are the square-root versions of the Sobel mask. Other ones in the set can be used as line detectors, note that the 3rd and 2nd last are versions of the laplacian, to be mentioned in the second-derivative edge detector section below.

Four of the Frei and Chen masks are called the basis of edge subspace, they represent

four orthogonal vectors corresponding to edges (out of the 9-dimensional space).

$$\begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$$
$$\begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} \sqrt{2} & -1 & 1 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix} \tag{B.11}$$

Four of the Frei and Chen masks are called the basis of line subspace, they represent four orthogonal vectors corresponding to edges (out of the 9-dimensional space).

$$\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$
$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix} \tag{B.12}$$

The final mask is the "average subspace", and finishes off the set of orthogonal basis vectors.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{B.13}$$

The similarity a 3x3 pixel neighbourhood has with a given basis vector has to include more than just a convolution, some adjustment to account for the input neighbourhood magnitude must be done. The similarity can be expressed as the cosine of the angle between an input neighbourhood (expressed as a 9-dimensional vector) and a basis vector by finding the inner product and then dividing by the magnitudes. Note that the linear non-normalized procedures of the previous section are analogous to performing this mask convolution (inner/dot product) without the normalizing division.

To use this set to just find edges, it is only necessary to find the projection onto the first four masks, or even just the first two for horizontal and vertical edges and then normalize this by dividing by the square root of the sum of all the neighbourhood pixels individually squared, we obtain the cosine of the angle between the neighbourhood's vector and that of the two reference edge vectors. Thresholding this cosine result (it is not necessary to convert this to degrees or radians) provides us an apparently more robust edge detector. Now edges in small regions in dark areas will be more defined. We can have the problem of finding too weak edges, which are really noise and so a good compromise might be to logical and the binary image produced by this approach and that provided by a previous linear method.

An adaptation of this method for corners is [115].

In summary, dividing the result of a linear convolution with some measure of the image intensity can make the edge detection more adaptive before the thresholding occurs. The Frei and Chen edge detectors are mask convolutions divided by the square root of the sum of intensities squared, thus providing 2 or 9 "scores" of the probability of a fit to each of the basis vectors. This procedure is sometimes performed using only the first two vectors listed, which can be referred to as the 'square-root' Sobel convolution masks due to their similarity to the Sobel set.

B.3 Second-Derivative Difference Operators

The second derivative of a signal can be used to find edges as well, although is much more noisy, requires care in implementation but has merit in that it can locate the center of a gradual edge and can provide single pixel wide edgels.

The second derivative gives us information about when the intensity is getting steeper or shallower. It provides two response peaks around an edge, one positive and one negative. Thresholding this response either positive or negative is not suggested as that it will provide a response on one side of the edge and not in the center.

The motivation for finding the second derivative of the intensity in a neighbourhood is usually to find the points of maximum slope, i.e. where the 2nd derivative equals zero. The three methods mentioned below provide a 2-D image of the second derivative and it is necessary to process it further to find the pixel locations where this is crossing zero. The easiest way to accomplish this is to threshold this 2nd-derivative image with '0' and produce a binary image where values above zero are given one intensity, and others below zero another. A simple binary edge detector can produce a one-pixel wide binary image of the nearest pixel location where these zero-crossings occurred.

The *Laplacian* is defined as the sum of the second derivative in the X-direction and that in the Y-direction. Below are 3x3 masks which estimate the second derivative in the X and Y direction correspondingly, followed by their sum to produce the Laplacian:

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{B.14}$$

Here is another Laplacian mask composed of the sum of second derivative masks in the X and Y, but with the 2nd derivatives averaging 3 pixels on each side of the center pixel:

$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} + \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix} \tag{B.15}$$

Here is a third example which produces the laplacian in a form seen in many image processing packages, such as for remote sensing:

$$\begin{bmatrix} -0.5 & -2 & -0.5 \\ -1 & 4 & -1 \\ -0.5 & -2 & -0.5 \end{bmatrix} + \begin{bmatrix} -0.5 & -1 & -0.5 \\ -2 & 4 & -2 \\ -0.5 & -1 & -0.5 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \text{(B.16)}$$

The laplacian is very noisy since it corresponds on differentiating the image twice and noise is amplified by differentiation. Usually the image is first smoothed before applying the Laplacian. The *Marr-Hildreth* edge operator [74] is a mask which consists of the laplacian convolved with the gaussian - also known as the *Laplacian-of-Gaussian* filter. This integrates both smoothing and the laplacian operator. The second derivative however gives no weight to the 2nd derivative in the edge direction and so is said to be more susceptible to noise.

The *Gaussian* is a smoothing function, where the attentuation fades with radial distance as shown below. α is the standard deviation.

$$h(x,y) = exp(-\frac{r^2}{2\alpha^2}) = exp(-\frac{x^2+y^2}{2\alpha^2}) \quad \text{(B.17)}$$

The Laplacian of the Gaussian is below:

$$LG(x,y) = (\frac{r^2-\alpha^2}{\alpha^2})exp(-\frac{r^2}{2\alpha^2}) \quad \text{(B.18)}$$

A refinement on this is the *Canny* edge operator which first calculates the edge direction in a neighbourhood by using one of the first derivative operators in the section above, and finds the second derivative along the this direction and is hence more noise immune. This is accomplished by finding the second derivative in the X and Y independently (using one of the mask pairs above) and calculating the sum of: (Eqn.B.19)

$$2ndDerivative = D2Xcos(\theta) + D2Ysin(\theta) \quad \text{(B.19)}$$

Where D2X and D2Y are the estimated 2nd derivatives in the X and Y direction respectively. Theta is the edge angle calculated by finding the inverse tangent of the first derivative of $\frac{Y}{X}$. Note that the Marr-Hildreth method is equivalent to merely summing D2X and D2Y.

Note that the Canny edge detector is defined as these calculations on the neighbourhood after being smoothed with the Gaussian as in the Marr-Hildreth operator. A discussion of practical issues in its implementation is given in [68].

The advantage to second-derivative methods is that there is no arbitrary threshold, other than zero when producing the binary image to find zero-crossings. These methods

have drawbacks in that a contour will always be found, where there are real edges, but also in slightly noisy constant regions as that zeros of the second derivative will exist there too. The contours tend to 'swim around' and are not always at the correct edge position, and produce many false contours in uniform image regions.

This leads to the technique of only using one of these second derivative techniques to fine-tune an edge location after it has been located by other means such as first derivative methods. Thus we can declare a pixel is an edge pixel if it satisfies two requirements: the first derivative is above a threshold, and the second derivative has a zero crossing.

B.4 Parametric Edge Filters

Model-Matching Edge detectors attempt to fit some model of an edge to a neighbourhood and determine the confidence of the fit. This assumes some knowledge or assumption of what defines an edge and Nalwa [80] claims this type of filter tends to err on the side of false negatives, i.e. more not recognizing a real edge than falsely reporting an edge where there is none.

Huekel [59] is mentioned in a few vision texts and his method attempts to match a neighbourhood to a step edge with the unknown parameters of edge direction, position and average intensity values on both sides of the edge (total of four variables).

Nalwa and Binford [81, 80] discriminate edges based on whether the neighbourhood matches an edge model better than it matches a quadratic surface (assuming a area of quadratically sloping intensity is an area where false edges may be reported with other methods).

Indeed the Frei and Chen method of determining the fit of a neighbourhood to an edge type by normalizing the projection of one to another can also be viewed as a model-matching type of edge detector.

Others such as Prewitt and Haralick developed edge filters that attempt to fit a surface to the data (as mentioned above with the Prewitt masks being the result of a least squares fit of a planar surface).

B.5 Edge Detector Performance

Many texts provide examples of edge detected images, potentially before and after some thresholding operation, so that one can qualitatively analyze an edge detector's performance. One can scan the image looking for edges and see if they are reported as binary one's or

strong responses in the output image and see how many falsely chosen edgels there are.

Some researchers have attempted to quantify the performance of edge detectors by devising some numerical tests. The *Signal-to-Noise-Ratio* (SNR) concept from signal processing is applied, with different metrics suggested to related the number of false positives and negatives to the standard deviation of a Gaussian noise source.

Other tests include analyzing how the edge direction is maintained - determining the so called *isotropic* nature of an edge detector. A definition of the term isotropic is 'direction-invariant' meaning in this context that if an edge operator is isotropic it gives the same response to an edge regardless of that edge's orientation.

Pratt in his image processing textbook [93] goes into great depth investigating and reporting investigations in the literature of edge detector performance.

A variable resolution [2], a neural net approach [92] and other edge detection methods and approaches are given in [97, 100, 117, 94].

Related sources on finding outlines and boundaries are given in chapter 2 on the Hough transform and its applications and [72, 67, 75].

Appendix C

An Automatic Single Plane Calibration System

Calculation of Intrinsic and Extrinsic Camera Parameters is important for computer vision tasks such as stereo reconstruction. This appendix details a working automatic system to gather all parameters except scale factor (aspect ratio) from one or more images of a given single-plane calibration pattern places placed in the field of view - and a simple extension of this method to calculate the scale factor is also presented. One image - as long as it is not fronto-parallel of the calibration pattern is sufficient to calculate all parameters (except scale factor), but it is shown that improved performance can be achieved by using several images of the calibration plane. No physical measurements are necessary, and the coordinate system is defined by the location of the calibration pattern. Tests with synthetic and real images are presented.

C.1 Introduction

To convert the location of a feature in an image to a representation of the corresponding ray in space, the image capture device (digital camera or video camera & frame-grabber in most cases) must be calibrated. The intrinsic parameters of focal length, scale factor (aspect ratio) and radial distortion parameters must be done at least once for a camera, and the extrinsic (position,orientation) parameters must be calculated every time the camera is moved. Addressing radial distortion is especially important for use with low cost off-the-shelf cameras as that the field of view is typically very curved.

There is much literature on camera calibration [63]. Roger Tsai's method [110] is the most oft-quoted in vision publications on calibrating cameras, taking into account radial distortion. The work of Faugeras and Toscani [38] is also mentioned as an important calibration paper.

This paper details the theory, practical issues and experiments using a single-plane automatic calibration system. The term single plane refers to the calibration pattern being entirely on a flat surface, as opposed to other systems using two calibration planes (such as Wei and Ma [113]).

The usage of the system presented here can be divided into two main parts: calculating intrinsic and extrinsic parameter sets. This system can simultaneously capture both from one image (as is shown in the stereo reconstruction example), but the suggested paradigm is to calibrate the intrinsic parameters separately. When the calibrated camera is placed in the scene to be modeled, pose (extrinsic parameters: position and orientation) is more accurately found (and the non fronto-parallel constraint removed) if one already has the intrinsic information available.

The first part of the intrinsic calibration stage is to model the imperfections of the

camera/lens system. The motivation is to obtain image coordinates of a feature in an image as they would be captured by an ideal pinhole perspective projection. The science of photogrammetry defines both thin prism and radial distortion as main effects. Tsai [110] claims from experience that only radial distortion need be considered with most available video cameras/digital cameras available for computer vision. The system presented in this paper relies on the fact that a two-dimensional pattern of lines will have the straightness of these lines maintained only by correct correction, and so numerical iterative methods can be used to obtain suitable parameters.

The focal length and pose calculation are obtained by fitting the sample points from the calibration pattern to an equation reflecting an affine transform (the nature of a perspective projection). The image coordinates of the sample points (after correction for radial distortion) are fitted with the least-squares method to this equation. The coefficients found contain the desired pose information, but these desired values need to be separated. If the focal length is available this can be done directly and the location of the focal point of the camera, and direction and rotation of the optical axis found. A method of using vanishing points is used to calculate this focal length if not known- and this explains why a fronto-parallel orientation of the calibration panel is problematic as that there are no finite vanishing points for the lines in the calibration pattern.

In the system developed, the world coordinate system is defined by the axes on the calibration pattern. This system is useful in the following example scenarios:

1-Stereo reconstruction. Several calibrated or uncalibrated (or some mix thereof) cameras are placed arbitrarily around a scene. A flat panel with the calibration pattern is placed somewhere in the the scene such that it is visible from all cameras. Images are then taken and processed by this system. As long as the scale-factor was known before-hand, all intrinsic and extrinsic parameters are then calculated automatically. Now the panel is removed and as long as the cameras are not moved, the relationship between image coordinates and a description of the line that feature must lie upon is well approximated. After identifying and matching features, the 3D coordinates (relative to the axes defined by the calibration pattern) can be readily obtained.

2-Motion tracking of markers on persons/objects relative to an image sensor's imaging plane require the intrinsic parameters only. Capturing one image of the calibration pattern at a non fronto-parallel pose is enough to get a working approximation of these parameters. Alternatively, two images could be captured for greater accuracy and remove the condition of the calibration pattern being non fronto-parallel. One image of the calibration panel in a fronto-parallel (or roughly close to) position could be used to automatically calculate radial distortion parameters, and a second image with the panel at some angle 30^o-60^o could be used to calculate the focal length more accurately.

Thus this paper details a working system that can be used to quickly and conveniently calibrate cameras with no physical measurements and the only special equipment being the preparation of this calibration panel (or the utilization of some known surface with features located along straight lines such as a building facade).

C.2 Stages of Automatic Calibration

There are four distinct stages used: 1:Corner Extraction, 2:Mesh Finding, 3:Calculating Radial Distortion Parameters and 4:Extracting Focal Length and Camera Position/Orientation. The Corner Extraction takes the image of the calibration panel as input, and produces a list of corner locations. The Mesh Finding stage uses this list and the image itself to locate the markers identifying the origin of the calibration pattern and matches the corners to their location on the calibration plane, producing a list of of X, Y, Z, U, V point sets which are processed by the last stage to if necessary find the focal length foc and the pose parameters (position and orientation).

Different combinations of the stages can be used in different calibration strategies. They may all be used to calculate everything in one image (as is done in the example stereo reconstruction at the end of this paper). Or if the radial distortion parameters only are being found, the first three stages only would be used. To calculate the focal point, with the radial distortion parameters already obtained in a previous calibration run, the 3rd stage can be skipped, and only foc retained from stage 4. If all the intrinsic parameters are known, stage 3 can be skipped, and foc provided to the stage 4 processing.

C.2.1 Automatically Calculating Radial Distortion parameters

It would be desirable to be able to treat the image from a camera as that of a perspective projection, however most optical systems, and certainly (inexpensive non-metric off-the-shelf video cameras and digital cameras) exhibit some distortions from this ideal. Photogrammetry texts (example [77])typically refer to two main effects that need attention: *Radial* and *Thin Prism Distortion.* Tsai [110] claims it is necessary in his experience to consider only the radial distortion.

Radial distortion is a phenomenon whereby the lens(es) bend the light such that the difference between the captured image and that of an ideal pinhole projection is that the location of points are changed in their radius from some central point. A point (U_w, V_w) in the captured image is a warped version of the same point (U_i, V_i) in an ideal projection. This function is a non-linear function of radius, and it represented here as the ideal radius R_i being a polynomial function of the warped (captured) radius R_w from the center of

distortion (U_{center}, V_{center}).

$$U_w = Scale_x \cdot R_w \cdot cos(theta) + U_{center}, \quad V_w = R_w \cdot sin(theta) + V_{center} \qquad \text{(C.1)}$$
$$R_i = R_w + K_2 \cdot R_w^2 + K_3 \cdot R_w^3 + K_4 \cdot R_w^4$$
$$U_i = R_i \cdot cos(theta), \qquad V_i = R_i \cdot sin(theta)$$

This system was implemented initially with only coefficients for the 2nd,3rd and 4th power which was found to be sufficient for the tests done, however photogrammetry references and Tsai claims only odd powers (3rd,5th,7th) are needed. The tests so far have been on low-resolution cameras and a 4th order polynomial appears sufficient but the system is being expanded to handle up to the 7th power to accommodate the anticipated better accuracy required for higher resolution sensors.

Once a grid is identified by the previous of mesh finding, a measure can be made of the "straightness" of the points along each line, the sum-of-squares from a least squares line fitting was used. These errors tend to be greater further away from the optical center as the field curvature gets more pronounced, this motivates this part of the technique whereby the center of distortion (U_{center}, V_{center}) is found by linearly interpolating between the lines of least and second least errors. Tsai and others have reported that the accuracy of this location is not too important, with synthetic images tested for this paper this center was usually found within 10 pixels in the 400,400 images tested.

After the center has been identified, iterative numerical techniques are used to search the possible solution spaces with the goal to minimize the sum of squared error measures from all the lines. All combination of solutions types are tried: with no radial correction, with K_2 only, K_3 only, K_4 only, $K_2 and K_3$, $K_2 and K_4$, $K_3 and K_4$, and finally with $K_2, K_3 and K_4$. The one with the least error is chosen. Since it is impossible to get a straight line with anything other than the correct radial distortion parameters, it is believed (but not proved here) that convergence is certain.

C.2.2 Automatically Calculating Camera Pose

It is assumed at this stage we have a list of X, Y, Z, U, V point sets [1] (known correspondences between a point X,Y,Z and its projection U,V). We first obtain a least-squares fit of these points to appropriate equations with intermediate variables, and then extract the desired position and orientation parameters, and focal length of the camera if it is not yet known.

[1] Note that the Z coordinate in the calibration points is 0 since this is a single-plane calibration technique and the calibration pattern lies upon the XY plane.

C.2.3 Extract parameters of Affine Transform

The perspective projection of a point (X_c, Y_c, Z_c) in "camera coordinates" to the location on the image plane (U, V) is $U = -foc\frac{X_c}{Z_c}$ and $V = -foc\frac{Y_c}{Z_c}$. The camera coordinate system is defined with the origin at the focal point of the ideal pinhole and the Z-axis being the central optical axis pointing down the line of sight. The X-axis and Y-axis are parallel, but in the opposite direction, to the U-axis and V-axis (on the inverted image plane) respectively. The U,V axis are on the imaging plane located a distance equal to foc pixels behind the focal point, normal to the Z-axis. The vectors $I_x = 1,0,0$, $I_y = 0,1,0$ and $I_z = 0,0,1$ define the orientation axis in camera coordinates. The focal point is located at (X_w, Y_w, Z_w)=0,0,0, and points on the imaging plane are located at $(-U, -V, -foc)$. The "world coordinate system" is different from that of the camera(s), with points (X_w, Y_w, Z_w). The extrinsic parameters of the camera are defined as vectors in world coordinates with the focal point located at C_x, C_y, C_z. and the I_x, I_y, I_z vectors now being represented as

$$I_x = [i_{xx}, i_{xy}, i_{xz}], \quad I_y = [i_{yx}, i_{yy}, i_{yz}], \quad I_z = [i_{zx}, i_{zy}, i_{zz}]$$

The conversion of a 3-D point in world coordinates to image coordinates is shown below, first with the intermediate step of conversion to camera coordinates. The coordinates in camera coordinates are the projection of the vector from the camera center to the point, onto the orientation axis I_x, I_y, I_z.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} i_{xx} & i_{xy} & I_{xz} \\ i_{yx} & i_{yy} & I_{yz} \\ i_{zx} & i_{zy} & I_{zz} \end{bmatrix} \begin{bmatrix} X_w - C_x \\ Y_w - C_y \\ Z_w - C_z \end{bmatrix}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} i_{xx} & i_{xy} & i_{xz} \\ i_{yx} & i_{yy} & i_{yz} \\ i_{zx} & i_{zy} & i_{zz} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - \begin{bmatrix} i_{xx}C_x + i_{xy}C_y + i_{xz}C_z \\ i_{yx}C_x + i_{yy}C_y + i_{yz}C_z \\ i_{zx}C_x + i_{zy}C_y + i_{zz}C_z \end{bmatrix}$$

The perspective equation then follows.

$$U(X,Y,Z) = -foc\frac{X_c}{Z_c} = -foc\frac{i_{xx}X + i_{xy}Y + i_{xz}Z - (i_{xx}C_x + i_{xy}C_y + i_{xz}C_z)}{i_{zx}X + i_{zy}Y + i_{zz}Z - (i_{zx}C_x + i_{zy}C_y + i_{zz}C_z)} \quad \text{(C.2)}$$

$$V(X,Y,Z) = -foc\frac{Y_c}{Z_c} = -foc\frac{i_{yx}C_x + i_{yy}C_y + i_{yz}Z - (i_{yx}C_x + i_{yy}C_y + i_{yz}C_z)}{i_{zx}C_x + i_{zy}C_y + i_{zz}Z - (i_{zx}C_x + i_{zy}C_y + i_{zz}C_z)} \quad \text{(C.3)}$$

The principle ray, determined by the location of the focal point and the direction of the I_z vector intersects the XY plane at a point defined as X_0, Y_0. Since the calibration pattern lies upon the XY plane, this point X_0, Y_0 is what is viewed on the image plane at $U, V = 0, 0$. The location of the camera's focal point can be defined relative to this point by extending "backwards" along the I_z axis by a length w. Thus,

$$C_x = X_0 - i_{zx}w, \quad C_y = Y_0 - i_{zy}w, \quad C_z = -i_{zz}w \quad \text{(C.4)}$$

If equation (C.4) is subbed into (C.2),(C.3) and recognizing dot products between the unit orientation vectors I_x, I_y, I_z as being either 0 or 1, the constant terms can be simplified and U, V expressed as follows;

$$U(X,Y,Z) = = -foc\frac{i_{xx}X + i_{xy}Y + i_{xz}Z - i_{xx}X_0 - i_{xy}Y_0}{i_{zx}X + i_{zy}Y + i_{zz}Z - i_{zx}X_0 - i_{zy}Y_0 + w} \quad \text{(C.5)}$$

$$V(X,Y,Z) = = -foc\frac{i_{yx}X + i_{yy}Y + i_{yz}Z - i_{yx}X_0 - i_{yy}Y_0}{i_{zx}X + i_{zy}Y + i_{zz}Z - i_{zx}X_0 - i_{zy}Y_0 + w} \quad \text{(C.6)}$$

After the intersection points are recognized in the image producing a list of X, Y, Z, U, V point sets (known correspondences between a point X,Y,Z and its projection U,V), it is desired to use find a least squares fit of equations (C.5),(C.6). This cannot be achieved since the unknown coefficients are multiplied together, and so the following substitutions are made;

$$M = i_{zx}X_0 + i_{zy}Y_0 - w \quad \text{(C.7)}$$

$$A = \frac{foc \cdot i_{xx}}{M}, \quad B = \frac{foc \cdot i_{xy}}{M}, \quad C = \frac{foc \cdot i_{xz}}{M} \quad \text{(C.8)}$$

$$D = \frac{foc(i_{xx}X_0 + i_{xy}Y_0)}{M}$$

$$E = \frac{foc \cdot i_{yx}}{M}, \quad F = \frac{foc \cdot i_{yy}}{M}, \quad G = \frac{foc \cdot i_{yz}}{M}$$

$$H = \frac{foc(i_{yx}X_0 + i_{zy}Y_0)}{M}$$

$$J = \frac{i_{zx}}{M}, \quad K = \frac{i_{zy}}{M}, \quad L = \frac{i_{zz}}{M}$$

Using equations (C.7),(C.8) we can rewrite equations (C.5),(C.6) to express in the following form for which a solution to the least squares curve fitting could be performed for the X, Y, Z, U, V point sets.

$$U(X,Y,Z) = -\frac{A \cdot X + B \cdot Y + C \cdot Z + D}{J \cdot X + K \cdot Y + L \cdot Z + 1} \quad \text{(C.9)}$$

$$V(X,Y,Z) = -\frac{E \cdot X + F \cdot Y + G \cdot Z + H}{J \cdot X + K \cdot Y + L \cdot Z + 1} \quad \text{(C.10)}$$

We can obtain closed equations for A, B, D, E, F, H, J, K but not C, G, L since the $Z = 0$ for all our calibration points. However since $I_x, I_y and I_z$ are perpendicular and foc and M are scaling factors,

$$A \cdot E + B \cdot F + C \cdot H = 0 \quad \text{(C.11)}$$

$$A \cdot J + B \cdot K + C \cdot L = 0$$

$$E \cdot J + F \cdot K + G \cdot L = 0$$

we can solve for C, G, L;

$$C = -\frac{(A \cdot J + B \cdot K)}{\sqrt{-\frac{(E \cdot J + F \cdot K) \cdot (A \cdot J + B \cdot K)}{(A \cdot E + B \cdot F)}}} \quad \text{(C.12)}$$

$$G = -\frac{(E \cdot J + F \cdot K)}{\sqrt{-\frac{(E \cdot J + F \cdot K) \cdot (A \cdot J + B \cdot K)}{(A \cdot E + B \cdot F)}}}$$

$$L = \sqrt{-\frac{(E \cdot J + F \cdot K) \cdot (A \cdot J + B \cdot K)}{(A \cdot E + B \cdot F)}}$$

The equations for A, B, D, E, F, H, J, K are not given (as that there are almost 200 terms for each), but after these parameters (and C, G, L are obtained, we now have a way to relate world coordinate points to and from image points. But to calculate the focal length, position and orientation of the camera it is necessary to separate out the original parameters.

C.2.4 Calculate Focal Point Using Vanishing Points

If the focal point is not known, it can be calculated in the following fashion, otherwise this step is skipped.

I_x, I_y, I_z in camera coordinates (referred to herein as I_{xc}, I_{yc}, I_{zc} can be estimated by finding the vanishing point(s) for equations (C.2,C.3). The coordinates of the vanishing points aren't necessarily within the image boundaries and will have some finite value unless the calibration is fronto-parallel in which case A, B, J, K are all zero. Best results are obtained for this stage if the calibration panel is located at some angle 30^o-60^o.

$$I_{xc} = \begin{bmatrix} Lim_{X->0} \frac{A \cdot X + B \cdot Y + C \cdot Z + D}{J \cdot X + K \cdot Y + L \cdot Z + 1} \\ Lim_{X->0} \frac{E \cdot X + F \cdot Y + G \cdot Z + H}{J \cdot X + K \cdot Y + L \cdot Z + 1} \\ foc \end{bmatrix} = \begin{bmatrix} \frac{A}{J} \\ \frac{E}{J} \\ foc \end{bmatrix}$$

$$I_{yc} = \begin{bmatrix} Lim_{Y->0} \frac{A \cdot X + B \cdot Y + C \cdot Z + D}{J \cdot X + K \cdot Y + L \cdot Z + 1} \\ Lim_{Y->0} \frac{E \cdot X + F \cdot Y + G \cdot Z + H}{J \cdot X + K \cdot Y + L \cdot Z + 1} \\ foc \end{bmatrix} = \begin{bmatrix} \frac{B}{K} \\ \frac{F}{K} \\ foc \end{bmatrix}$$

I_{xc} and I_{yc} represent orthogonal vectors and so the dot-product is zero.

$$I_{xc} \cdot I_{yc} = 0$$

$$\frac{A}{J} \cdot \frac{B}{K} + \frac{E}{J} \cdot \frac{F}{K} + foc \cdot foc = 0$$

This allows us to calculate foc from A, B, E, F, J, K.

$$foc = \sqrt{-\frac{A \cdot B}{J \cdot K} - \frac{E \cdot F}{J \cdot K}} \tag{C.13}$$

C.3 Extraction of Position and Orientation Information

Having the focal length foc, the orientation parameters $i_{xx}, i_{xy}, ...i_{zz}$ can be extracted by first finding the scale factor M. Since $I_x, I_y and I_z$ are unit vectors,

$$i_{xx}^2 + i_{xy}^2 + i_{xz}^2 = 1, \qquad i_{yx}^2 + i_{yy}^2 + i_{yz}^2 = 1, \qquad i_{zx}^2 + i_{zy}^2 + i_{zz}^2 = 1$$

$$A^2 + B^2 + C^2 = foc^2 \cdot M^2, \quad E^2 + F^2 + G^2 = foc^2 \cdot M^2, \quad J^2 + K^2 + L^2 = M^2$$

And M can be calculated in three ways (labelled $M_x, M_y, M_z M$). It was found with the tests done that the average of $M_x and M_y$ produced the most reliable results. M can be either positive or negative, the sign that makes i_{zz} negative is chosen since the camera is assumed to be facing towards the calibration pattern, and on its front side.

$$M_x = \sqrt{\frac{A^2 + B^2 + C^2}{foc^2}} \tag{C.14}$$

$$M_y = \sqrt{\frac{E^2 + F^2 + G^2}{foc^2}}$$

$$M_z = \sqrt{J^2 + K^2 + L^2}$$

$$M = \pm\frac{M_x + M_y}{2} \tag{C.15}$$

With foc and M equations (C.8) can be used to extract the orientation vectors $I_x = (i_{xx}, i_{xy}, i_{xz}), I_y = (i_{yx}, i_{yy}, i_{yz}, I_z = (i_{zx}, i_{zy}, i_{zz})$. They are normalized and I_z recalculated as the cross-product of $I_x and I_y$ to provide a completely normalized, orthogonal basis.

X_0, Y_0, the point at which the I_z intersects the calibration plane, can be calculated by solving;

$$A \cdot X_0 + B \cdot Y_0 + D = 0 \tag{C.16}$$

$$E \cdot X_0 + F \cdot Y_0 + H = 0 \tag{C.17}$$

This can be derived by setting equations (C.5,C.6) both to 0. w can be calculated from M in (C.7) and the position (C_x, C_y, C_z) found from (C.4).

C.4 Fine-tuning Position and Orientation Information

The parameters $foc, w, X_0, Y_0, I_x, I_y, I_z$ will most likely not be precise, and applying equations (C.2,C.3) will produce minor mismatches between the observed (U, V) and (U, V) calculated from the $(X_w, Y_w, 0)$ world point. Numerical iterations testing close values of foc, w, X_0, Y_0 and small rotations of I_x, I_y, I_z according to a cost function such as the sum-of-absolute-differences or sum-of-squared-differences was found to improve the results.

C.5 Experimental Results

Calibration tests were performed for synthetic images with known intrinsic and extrinsic parameters, and with two image capture devices.

The synthetic images were generated as 400x400 pixels images with varying radial distortion, focal length and pose parameters. The localization of the optical center was correct within 5% and radial distortion parameters were found within 3%. The camera position (C_x, C_y, C_z) was found within 4% and the orientation within 1^o in yaw, pitch and roll. The explanation offered as to why the results were not exact is that of the corner detector providing only integer values of detected corners and other quantization error effects. Four example source images, and reconstructed views of the calibration pattern with the extracted parameters are given.

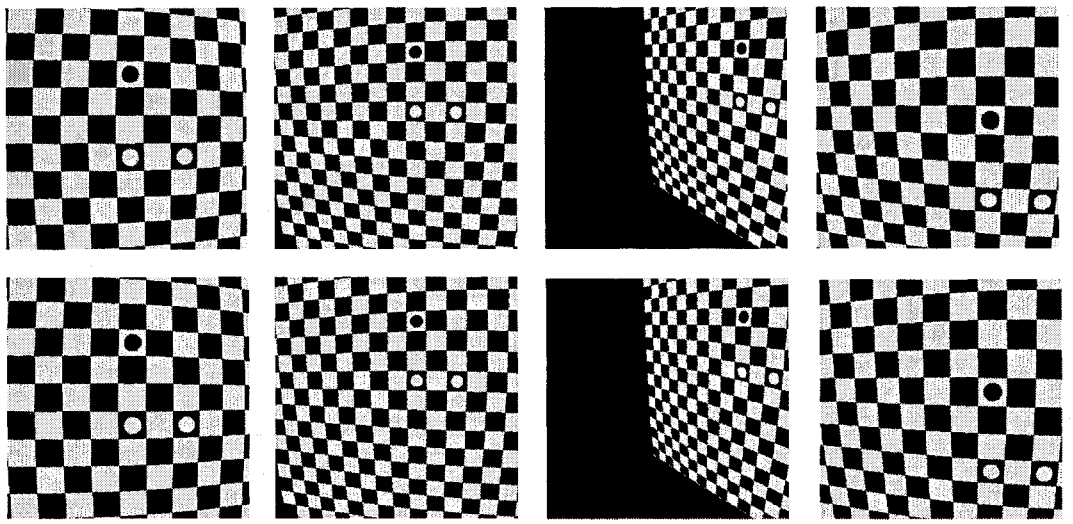


Figure C.1: *Synthetic test images: top row is original image, bottom row is reconstructed view using automatically determined parameters from above image.*

Calibration experiments were performed on the 2 Quickcams from different views and the results of these shown below in tables C.1 and C.2. Figure C.5 demonstrates the setup used to calibrate an inexpensive Greyscale Connectix Quickcam digital camera. This type of camera provides an image of 320x200 pixels with a depth 6-bits in intensity, is of low price and convenient to attach to a computer, but its consumer-grade optics provide high radial distortion that is quite different from unit to unit.

As expected, it was found in both the synthetic and real images that better accuracy for the radial distortion parameters can be achieved with a more fronto-parallel configuration of camera and calibration panel. Likewise a more accurate focal length can be calculated with the panel closer to 45^0 (with the radial distortion parameters already known).

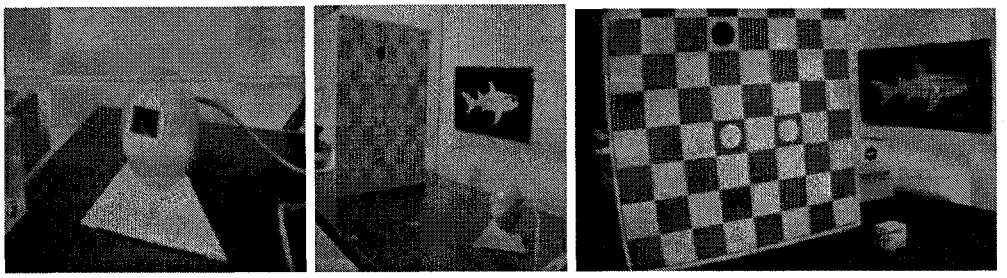


Figure C.2: *Example of calibration of a greyscale Connectix Quickcam digital camera.*

Num Sample Pts	Image Center	K_2	K_3	K_4	Foc	Error/pixel
49	191,88	0.0	0.0	0.0000000172	408.3	1.027
40	188,82	0.0	0.0	0.0000000244	—	—
29	181,86	0.0	0.0	0.0000000172	410.1	0.849
41	326,172	-0.00111	0.0	0.000000006	1050.2	4.480
42	186,94	0.0	0.000004	0.0	345.3	0.929
42	208,87	0.0	0.0	0.0000000128	—	—
32	193,99	0.0	0.0	0.0000000191	319.2	0.648

Table C.1: Results of Calibration on "Left" Connectix Quickcam Digital Camera. In some cases the pose/foc algorithms didn't find a foc as that the scene was too fronto-parallel. Error/pixel refers to a successful fitting of position/orientation to the sample points. The first row is for the image used in the figure C.5.

It was expected and found that the best results for modeling the effects of radial distortion were obtained with the calibration pattern occupying the full field of view of the camera, otherwise parameters will be found that minimize only the error of those points in part of the field of view, and large errors can result from extrapolating outside that range. Also it also improves the accuracy to choose the calibration pattern according to the sensor resolution to maximize the number of sample points.

Figure C.5 shows a stereo reconstruction of a scene in which a calibration panel was placed for automatic calibration. The panel was removed and a stereo pair of images captured without moving the cameras. The captured images, as well as reconstructed views of

Num Sample Pts	Image Center	K_2	K_3	K_4	Foc	Error/pixel
45	142,104	0.0	0.0	0.0000000221	—	0.653
37	134,92	0.0	0.0	0.0000000212	140.2	2.812
29	159,102	0.0	0.0	0.0000000208	877.3	3.331
36	159,89	0.0	0.0000040	0.0	490.4	1.160
46	151,91	0.0	0.0	0.0000000214	342.4	0.659
39	175,41	-0.00111	0.0	0.0000000190	336.3	5.710
36	150,84	0.0	0.0	0.0000000204	344.6	0.723

Table C.2: Results of Calibration on "Right" Connectix Quickcam Digital Camera. In some cases the pose/foc algorithms didn't find a foc as that the scene was too fronto-parallel. Error/pixel refers to a successful fitting of position/orientation to the sample points. The first row is for the image used in the figure C.5. Note that this view was too fronto-parallel to extract foc and so $foc = 408$ was used for the stereo reconstruction.

the calibration pattern and models of some of the objects are shown. The feature points were chosen manually in the two captured images.

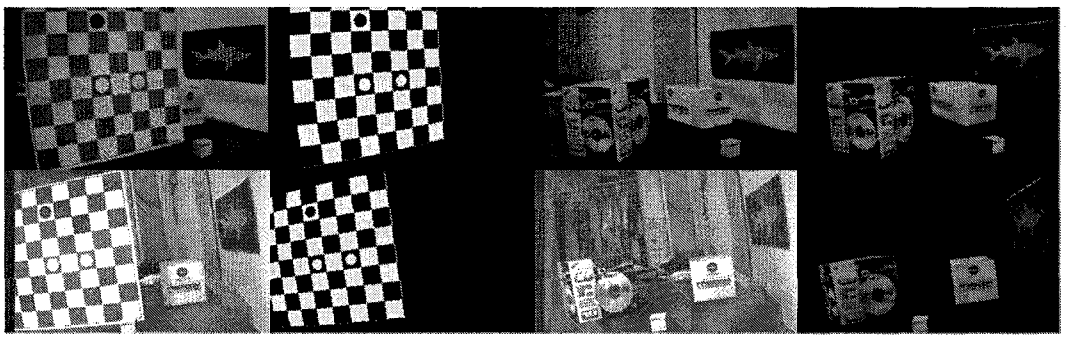


Figure C.3: *Reconstruction test images: top row for the left camera, bottom row for the right. Each row left-to-right: the captured image of the calibration panel, the synthetic image reconstruction of the panel given extracted parameters, the captured scene image, and the reconstructed view from that position from the model.*

A 3rd viewpoint of this model is shown in figure C.5.

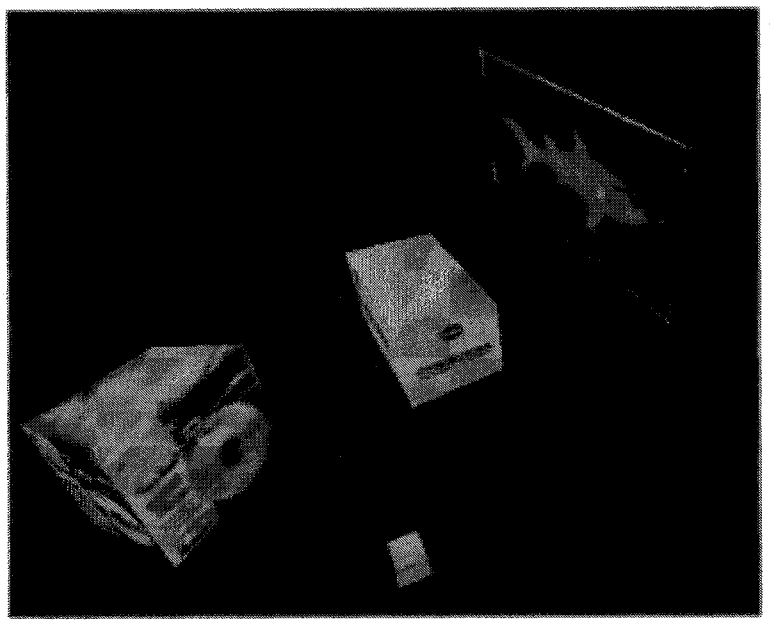


Figure C.4: *Top view of model created using the automatically calibrated camera setup. The features were chosen manually in the two images.*

C.6 Conclusions

A method for automatically calibrating cameras from one single-plane image has been demonstrated in theory and practice, with the suggested application of using these procedures in parts on several images to improve accuracy. Specifically, calculating the distortion parameters from a fronto-parallel (or roughly fronto- parallel) image, and following this with the focal point calculation with an image from an oblique viewpoint provides the best accuracy for intrinsic parameters.

The single plane calibration pattern obviates the need for the preparation and maintenance of an optical calibration range or preparation of precise 3D calibration patterns This paper demonstrates this calibration system applied in a very convenient and easy manner, just placing the calibration pattern in the field of view of the image sensor allows rapid calculation of quite precise calibration results.

Appendix D

Previous Work: Image Processing and Panoramic Imaging

Several years of work into real time image processing, remote operated vehicle image capture and surface creations and real time panoramic viewing systems for telepresence were carried out by the author.

D.1 Remote Underwater Vehicle

A control and positioning system was designed and built by the author to facilitate the capture of underwater images and compositing into a large model for underwater inspection tasks. The vehicle and some hardware developed are shown in Fig. D.1. The work relevant to this thesis was the calibration of the imagery provided by the wide-angled lens on the on-board video camera to correct for radial distortion and the image registration so that multiple images could be patched together to form a larger surface.

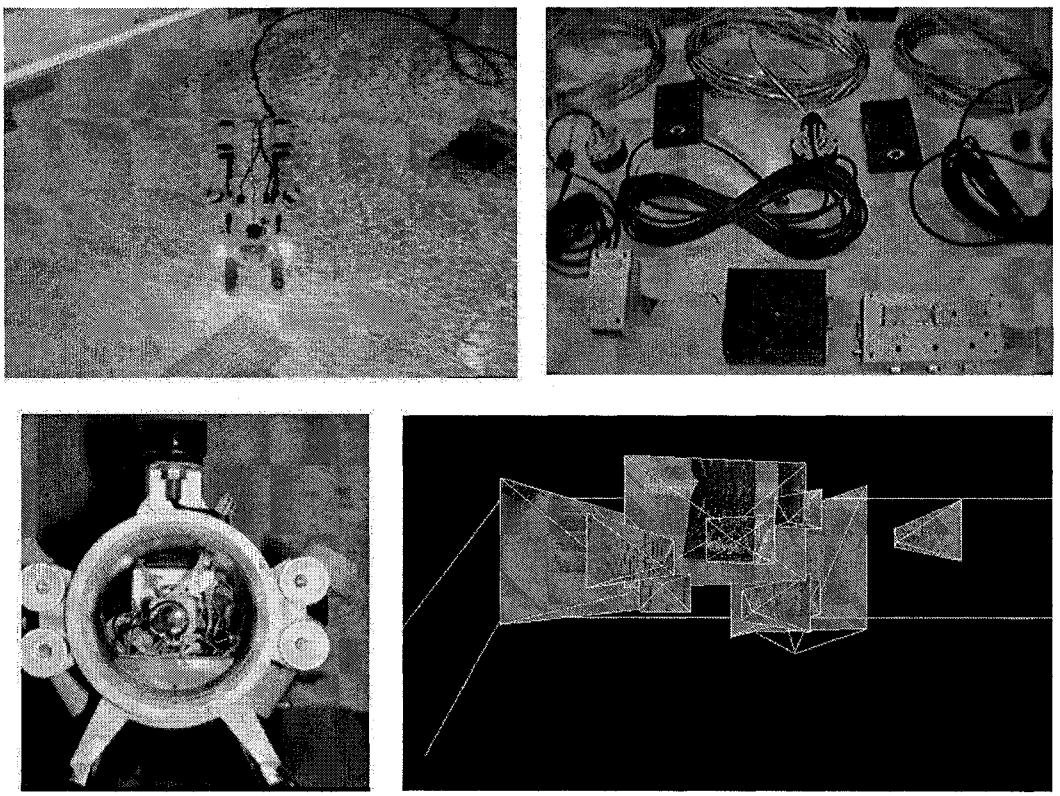


Figure D.1: *Remote Underwater Vehicle project components.*

Some basic camera calibration had to be applied to find the field of view and correct for radial distortion. Radial distortion correction was necessary to correct the gross radial distortion that the wide-angled lenses produced. Below (Fig. D.2 are some examples of image correction once the radial distortion parameters were found:

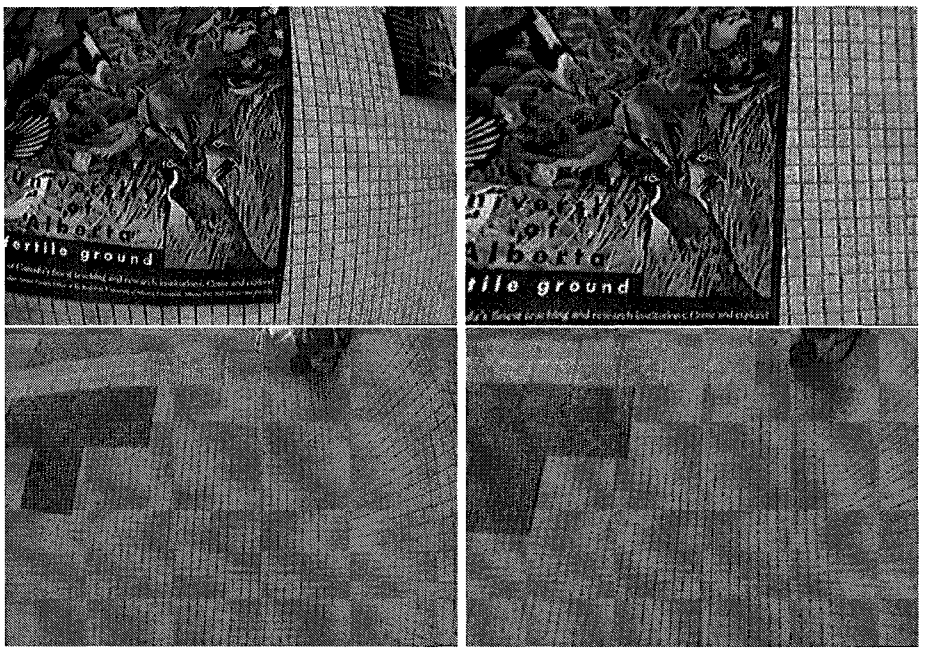


Figure D.2: *Examples of radial distortion from images from ROV where the air-water interface created a large radial distortion effect. The left images show the original images, the right show the images after radial distortion correction has been applied. Once the radial distortion parameters have been found, the same correcting warp can be applied to all images from the camera.*

D.2 Backscatter Elimination Circuit

A real-time video processing system was also developed to test a theory to attempt to remove back-scatter effects of small particles floating in front of a camera in underwater inspections. This involved the temporal filtering of images according to a simple non-linear greyscale decision function.

D.3 Panoramic Telepresence

The real time telepresence panoramic viewing systems were designed and built by the author in conjunction with a university research relationship with an external firm attempting to develop panoramic telepresence products. These involved the design and manufacture of several Image Transformation Engines (ITE's) and two families of panoramic video capture devices.

The first was built for PVSI and for a proof-of-concept system for bi-lobed panoramic stereo reconstruction. This work was done in late 1995, Fig. D.3 shows the same bi-lobed mirror used in the recent high-resolution experiments, but with an NTSC video camera and

the first ITE circuit which remapped the pixels to two horizontal panoramic strips.

This was a real-time hardware system designed and built by the author which could perform a general purpose remapping. It could be applied to create a quasi-cylindrical views from the curved images provided by several different panoramic optics such as conical mirrors, single lobed mirrors and this stereo panoramic system. The circuit was intended for use in panoramic viewing for telepresence, with the spatially warped image being presented to a human viewer.

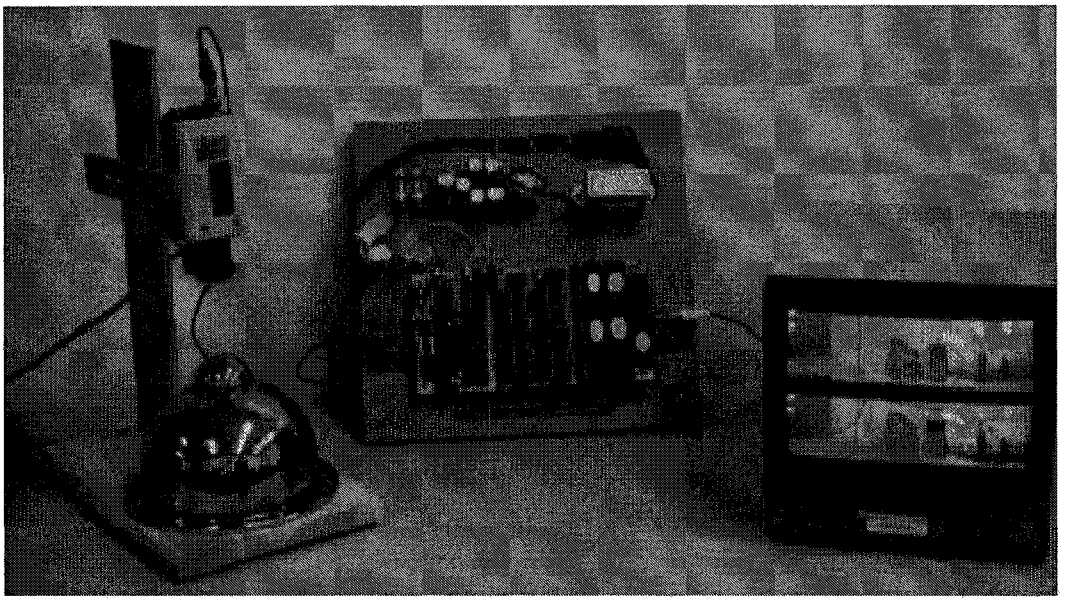


Figure D.3: *Stages of Line Extraction.*

D.4 Hardware Description of ITE No.1 - NTSC greyscale version: Fixed Spatial Transform

A more detailed hardware description is given for the above system. The incoming video is digitized, stored in a frame buffer, and an output image is generated from this stored frame using a mapping lookup table stored in erase-programmable memories. The circuit simultaneously stores the present incoming video frame, and generates the output image from the previous frame. This allows a 30 frames/second video signal to be continuously flowing out, delayed only one frame. The video is sampled at 10 MHz to provide a 512 x 480 pixel image with 8 bits of greyscale. There are two 256K x 8 bits frame storage buffers of fast SRAM (Static Ram) each capable of storing one frame. The data coming from the video ADC (Analog to Digital Converter) is written to one frame storage, while the other holds the previous frame and is being read from to generate data for the video DAC (Digital

to Analog Converter). A 256 K x 18 bits EPROM (Erasable Programmable Read Only Memory) array provides the X and Y coordinates of the stored pixel corresponding to each pixel in the output image. It is an inverse mapping, so that a single pixel in the input image can appear many times in the output image, thus producing a magnifying effect. The frame storage buffers alternate between being written to and read from at every frame, and all the data is carried through a common bus — the most efficient implementation. This results in a bus speed of 20 MHz. Since this circuit was realized discretely (using only standard off the shelf components), the 74F logic family was chosen for most of the logic. A software package was written to generate and test the lookup tables in software before programming the table into the EPROM array. One program takes a lookup table and remaps a test raster image file, another takes the lookup table and generates the files needed by the EPROM programmer. A program written specifically for this application generates the remapping table used; being the only part that must be changed for arbitrary remapping. This circuit is shown below in Fig. D.4.

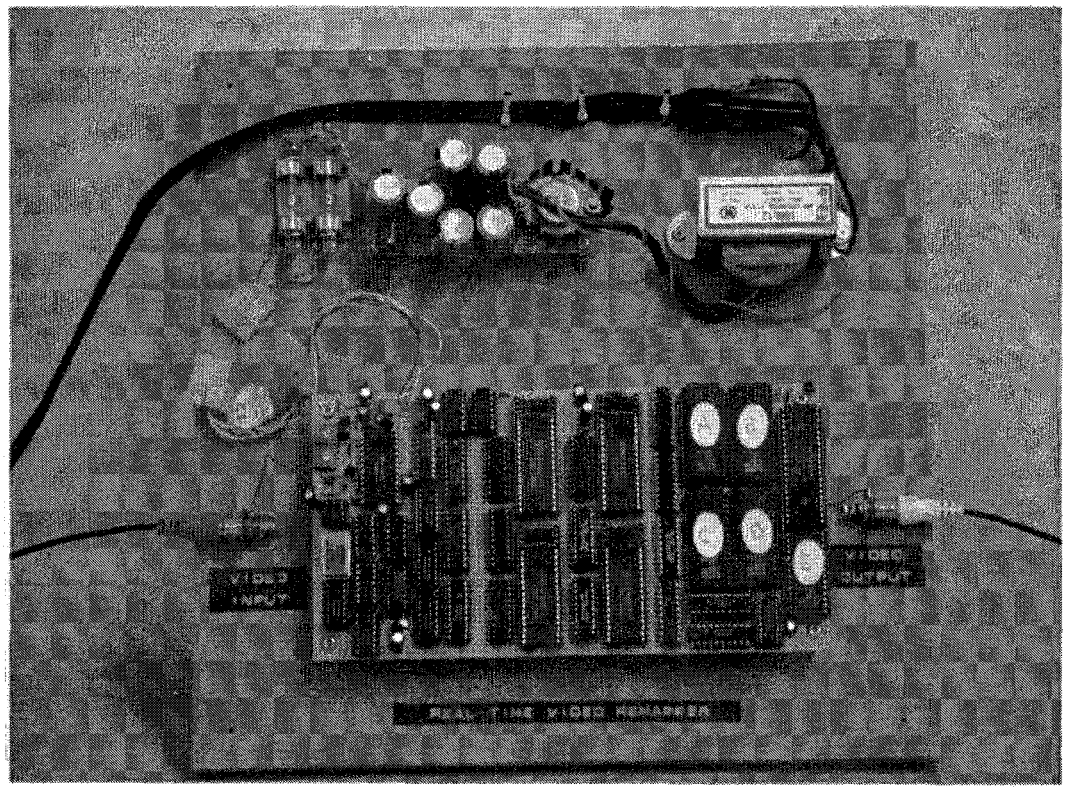


Figure D.4: *NTSC greyscale ITE (Image Transformation Engine).*

D.5 ITE No.2 - 24-bit Colour VGA Fixed Spatial Transform

The next system used a colour digital video camera and a new circuit designed and built by this author that produced a fixed transform of this digital video and provided it to a VGA monitor. This allowed the presentation of a full-colour, real time video panoramic view captured from a conical mirror and warped into a linear panorama. See Fig. D.5

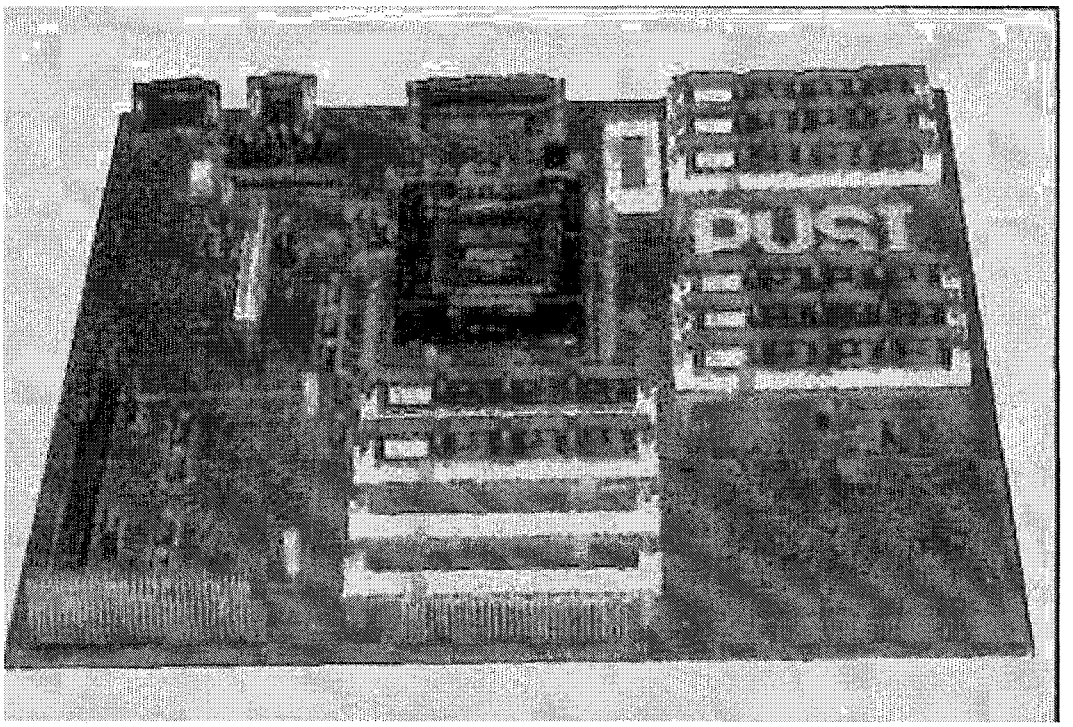


Figure D.5: *VGA Colour ITE (Image Transformation Engine).*

D.6 Multi-CCD Digital Camera

The author then went on to create a novel system involving a pyramidal mirror and 5 image capture devices to create a much higher resolution panoramic image than that obtained with the reflective optics D.6. The distances were chosen to make all the five image sensors have the same effective focal point to eliminate parallax distortion between images. The five video frames could be effectively stitched together to produce a hemispherical view with 1.25 Million RGB colour pixels at a rate of 30 frames/second. This was a vast improvement over the estimated 90,000 pixels or less that could be obtained by a narrow field-of-view camera of NTSC (standard video) resolution and reflective conical optic. This design was later

patented by Bell Labs (part of Lucent Technologies) but the issue of intellectual property was never investigated.

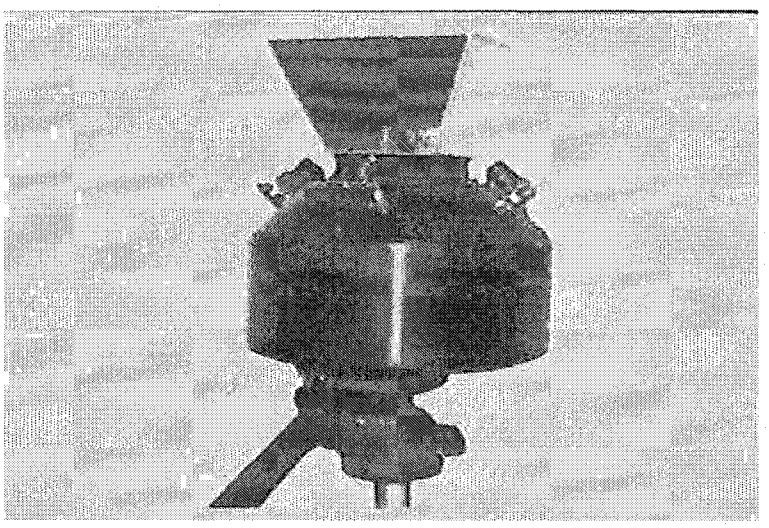


Figure D.6: *Planar SVP Panoramic Video Camera: Multi-CCD system.*

D.7 Development of Telepresence Paradigm

The author devised the vision of the product the external company was to develop. A previous *video clipper* approach had been put forth that would display only a section of a panorama to a viewer according to view direction, a sort of "virtual periscope". The author extended that to a head-mounted display based design that would provide the correct perspective view to a viewer according to the position that the head mounted display orientation tracker provided. This paradigm and the necessary transforms and math was provided by the author, and it was successfully incorporated into a system that allowed a working prototype of this system to be developed.