Terrain Mapping from a Mobile Platform Through Optical Flow Techniques

by

Jonathan Mark Walton

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

Bachelor of Science in Electrical Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1995

Signature of Author .....
Department of Electrical Engineering and Computer Science
May 12, 1995

Approved by .....
Dave Hauger
Technical Supervisor
The Charles Stark Draper Laboratory

Certified by .....
Berthold K. P. Horn
Thesis Supervisor
Professor of Electrical Engineering and Computer Science

Accepted by .....
Frederic R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Terrain Mapping from a Mobile Platform Through Optical Flow Techniques

by

Jonathan Mark Walton

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 1995, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Electrical Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes a camera-based range detection scheme designed as part of a complete terrain-mapping program for use on a six-degree-of-freedom cartesian robot in the Simulation Lab at C. S. Draper Lab. This range detection scheme uses least-squares optical flow techniques, but with several improvements to account for high-frequency images and for radial lens distortion.

A simple optical flow algorithm is first developed, assuming purely translational motion and using least-squares minimization to calculate the pixel shifts undergone by different regions of an image pair. Standard filtering, downsampling, and frame-averaging techniques are used to take each picture of the image pair, but a shift-biasing method is also developed to improve the derivative calculations and consequently the distance estimate. The standard optical flow equations are also altered to account for radial distortion. Furthermore, the optical flow algorithm is also used to calculate the radial distortion parameters, thereby supplying a self-contained means of analyzing the radial distortion of a lens and of removing its influence from the optical flow calculations.

The complete terrain-mapping program is also described, and a methodology for choosing values for the parameters is presented. Examples of processing individual image pairs have been included, as well as results from using the overall terrain-mapping program.

Technical Supervisor: Dave Hauger
Title: Principal Member of Technical Staff, Charles Stark Draper Laboratory

Thesis Supervisor: Berthold K. P. Horn
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank all of the member of the Simulation Lab for a great working environment, both technically and socially. In particular, I owe Mario Santarelli a great deal for believing in me and for insulating me from any bureaucracy I might have encountered. Thanks to Tim Kalvaitis for giving me both moral and technical support, even though his ties to Draper ended before this thesis was even begun. To Dave Hauger and Chris Winters, I owe my appreciation for the creation of both useful and reliable simulation software, without which this thesis would not have been possible. Dave has also been a great source of moral support and a much-appreciated sounding board. Thanks to Tim, Bill McKinney, and Torsten Berger for their contributions and extensions to the simulation software which, amongst other things, made it possible for me to use the robot and camera with ease.

Thanks also to my family and friends for their emotional support. To those of you who have tolerated too much of my stressful presence over the past few months and to those of you whom I have neglected, thank you for bearing with me.

Finally, I would like to thank Professor Horn for his insightful comments and realistic attitude. Without his useful suggestions, I would no doubt still be searching in vain both for a range-detection method to meet my requirements and for an overall focus for this thesis.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Internal Research and Development Project #13116, "Precision Vehicle Positioning and Autonomous Work Systems".

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis develops a range-sensing algorithm for use with a precision-motion mobile camera system. This chapter presents the motivation behind the research, as well as the reasons for choosing an optical flow technique. Chapter 2 explains the theory behind the optical flow algorithm. Chapter 3 provides frequency analysis of the optical flow system in order to understand and overcome problems with high-frequency optical flow analysis. Chapter 4 examines the effects of camera lens distortion and develops a method to remove its effects from the optical flow system. Chapter 5 unites the previous chapters into a complete range-sensing algorithm, as well as describing how this algorithm fits into a terrain-mapping utility for a simulation environment at C. S. Draper Lab.

1.1 Draper Lab Proximity Operations Project

The motivation behind this thesis was supplied in part by an internal development project at C. S. Draper Lab. This project is focused around a high-fidelity simulation of an unmanned underwater vehicle (UUV). The UUV simulation also involves a simulation of the surrounding environment and some simple sensors to interact between the submarine and the environment. This includes simulations of the ocean floor and

several different sonar systems for sensing that floor. At its current stage in development, the UUV simulation is being expanded through the addition of some visual sensors. In order to enforce a certain level of reality in the visual sensor system, these simulation sensors are being implemented through actual hardware as CCD cameras mounted on a high-precision robot environment. The robot, a six-degree-of-freedom cartesian gantry system, is controlled by the UUV simulation and slaved to the location of the vehicle in the simulation (The hardware is described in detail in Section 1.3.1). As the simulated vehicle moves, the gantry robot also moves, positioning the camera over the laboratory workspace. Images of the workspace environment are then taken and returned to the UUV simulation for processing, and the results of that processing are used to influence the control system on the submarine, thereby influencing its motion and subsequently the motion of the gantry-camera system.

In order to run a valid simulation, the simulated sensors (such as the sonar system) and the real hardware of the camera system must return corresponding information. It would be disastrous for the simulation if the different sensors returned conflicting information, such as the sonar giving an altitude measurement of 20 feet while the ground visible through the camera system was 50 feet away. However, the simulation would lose a lot of usefulness if it were restricted to always traveling over the same small plot of simulated terrain. Instead, an automated terrain mapping system is needed, so that an arbitrary terrain surface can be sculpted in the robot workspace, and a digitized version can automatically be generated for use in the UUV simulation.

The underlying motivation behind this thesis was to supply such a terrain mapping system. The system had to satisfy the following criteria:

- Implementable – The system must be built. A theoretical or impractical system would not suffice.
- Automated control – The system should be easy to use, as it will be used often. It should make all decisions from startup through to the end of the mapping process. It should need an operator only to begin the program.

- Accurate and robust – The system should be reliable and repeatable.

In addition to these hard limits, it seemed wise to go for something simple and economical. The nature of the goal itself guarantees that the terrain mapping system will not be the only program to be using this gantry robot. As a result, to keep the utility easy to use, the terrain-mapper should not require any special-purpose hardware to be mounted on the apparatus, as the hardware would probably have to be removed each time another program used the gantry. Combined with the desire to keep the project as inexpensive as possible, this suggests that the terrain mapping system should be built around the hardware currently available, without the purchase of any new equipment.

1.2 Automated Terrain Mapping

Having laid out the motivation behind an automated terrain mapping system, it seems reasonable to examine what such a system would involve. The floor of the workspace being mapped is approximately 75 square feet, which is too large to be mapped with reasonable accuracy from one single location. Instead, it seems reasonable to develop a local-area terrain mapping utility and then use the gantry robot to move this local mapper around the larger workspace. Technically, this terrain-mapper could take on many forms, such as a tactile system which moved over the surface and pressed some sort of force sensor down until it felt any resistance below it. However, a range-sensing system taking measurements from a fixed height seems to be far more practical. To map the terrain through range estimation, the system can combine the position and direction of the range-sensor with the range data acquired. By moving around the workspace in a methodical manner, a dense map of the entire terrain can be developed.

Inherent in the above idea for range sensing is the idea that one know the exact location of the gantry robot. This is necessary in order to use range estimates to determine the actual height of the terrain and also to piece together the local terrain

estimates into a single comprehensive map. To perform a methodical scan of the workspace, precise control over the position of the sensing instrument as well as control over the instrument itself is needed.

1.3 Draper Simulation Lab Environment

Considering the requirements and considerations laid out in the last two sections, it seems reasonable to examine the available resources in the lab and use that as a basis to help narrow down the possible implementations.

1.3.1 Available Hardware

The gantry robot is a six-degree-of-freedom cartesian robot. The gantry is constructed as a large frame over which the robot can move (see Figure 1-1). The base consists of two parallel tracks in the X direction, which support two more parallel tracks in the Y direction. On top of this is a platform with a large piston which moves up and down in the Z direction. At the bottom of the piston, the heading, pitch, and roll angles control the final axis, which holds a camera (see Figure 1-2). The limits for each gantry motor, as well as the available accuracy for commanding that motor, are shown in Table 1.1. The ranges for the X and Y axes dictate the maximum size of the workspace, which is approximately 11.5 feet by 6.5 feet. The motor accuracies reflect how precise the motors can be commanded, but do not account for any inaccuracies introduced by the drive and gearing systems. In general, these gearing errors can be avoided by limiting the maximum acceleration and by approaching all imaging points from the same axis direction. The camera is an 8-bit grayscale CCD camera with 640 pixels by 480 pixels of resolution. All motor, camera, and support signals are routed out of the gantry and into a large Simulation Interface Unit (SIU). The SIU houses the motor controllers along with analog and discrete inputs and outputs, which are all connected to an SGI workstation via a VME bus interface (the SIU and contents are visible in Figure 1-2). The robot can be controlled in its entirety from the SGI

Motor	Minimum	Maximum	Accuracy
X	-73.71"	67.92"	2.36×10^{-5} inches
Y	-42.66"	33.44"	2.36×10^{-5} inches
Z	-16.94"	18.04"	3.79×10^{-5} inches
Heading	N/A†	N/A†	1.44×10^{-3} degrees
Pitch	-99.66°	11.49°	2.04×10^{-2} degrees
Roll	-117.14°	119.90°	2.04×10^{-2} degrees

Table 1.1: Gantry Motors – Limits and Accuracy
These are the six degrees of freedom of the gantry robot, in order from robot base to camera. The default scaling from the simulated environment to the gantry is 12:1, so that 1 inch of gantry space is equal to 1 foot of simulated space
†Note: The heading motor is connected via a slip-ring, and so has unlimited travel.

workstation, and camera images can be captured from there as well.

1.3.2 Available Software

The SGI interface to control the robot is currently handled through Draper Simulation Lab's Simulation Framework. This Framework, written in C, is the basis for the Proximity Operations UUV simulation described above, as well as for several other sea, land, air, and space vehicles. The Framework provides easy access to the gantry and camera, and also provides an excellent development, debugging, and execution environment for the terrain-mapper code. For image analysis, the Framework include routines to handle large arrays, including a graphical browser to examine array values. In addition, it includes several general-purpose image processing and display capabilities. The Framework is capable of defining and using aliases as well as running script-files, and so is ideal both for experimentation and for batch operation. For a complete description of the Simulation Framework, see Appendix A.

1.4 Range Estimation – Why Optical Flow

After considering the existing hardware and software in the lab, the terrain-mapping system was implemented as a mobile vision-based system, using the camera mounted

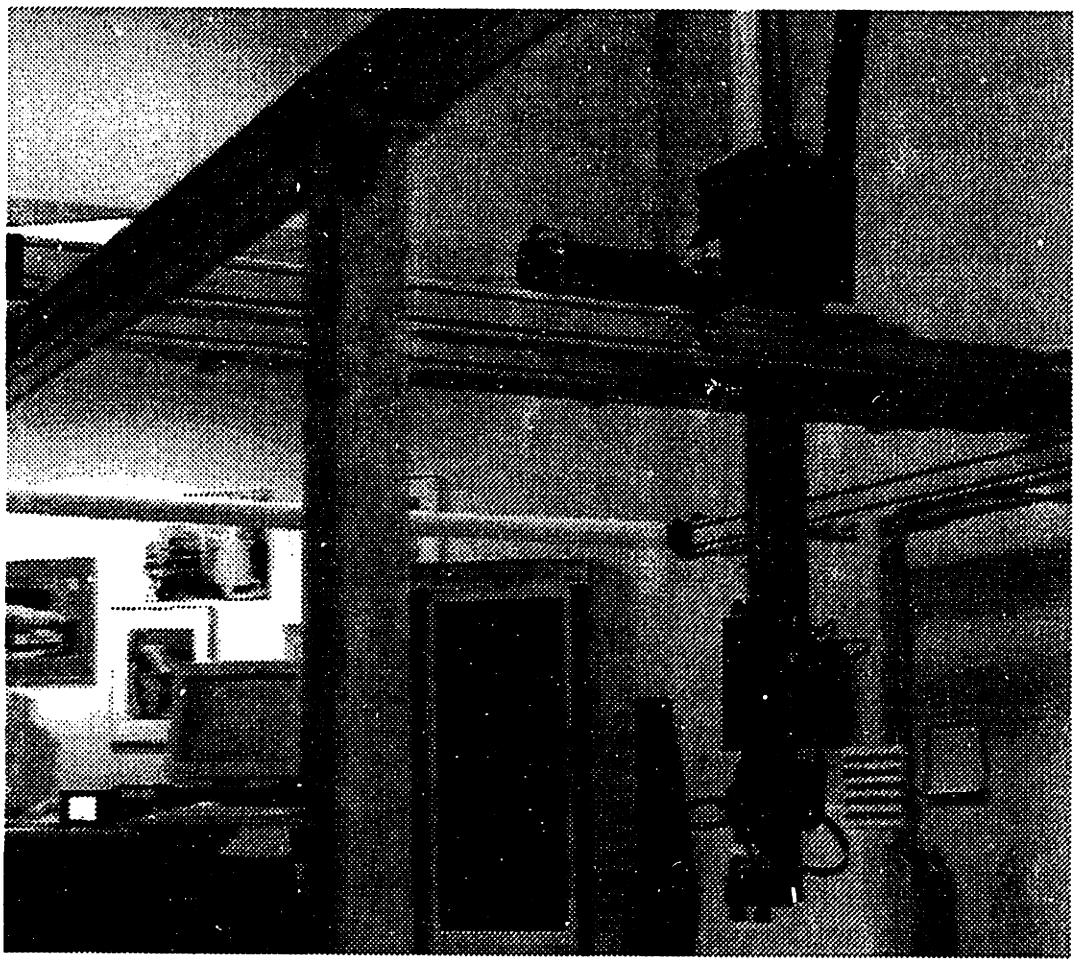

Figure 1-1: Gantry Frame Showing X, Y, and Z Axes
The gantry frame is visible in the foreground, with the X axis coming out of the image and to the right. The Y axis is attached on top of the two X rails (extending across the center of the image towards the left) and the Z axis points down towards the ground. The Z motor, brake, and clutch box are visible above the Y rail, and the camera and slip-ring assembly is visible below. The Simulation Interface Unit (SIU) sits against the wall in the background, and the SGI workstation which controls the whole setup is visible beside it.

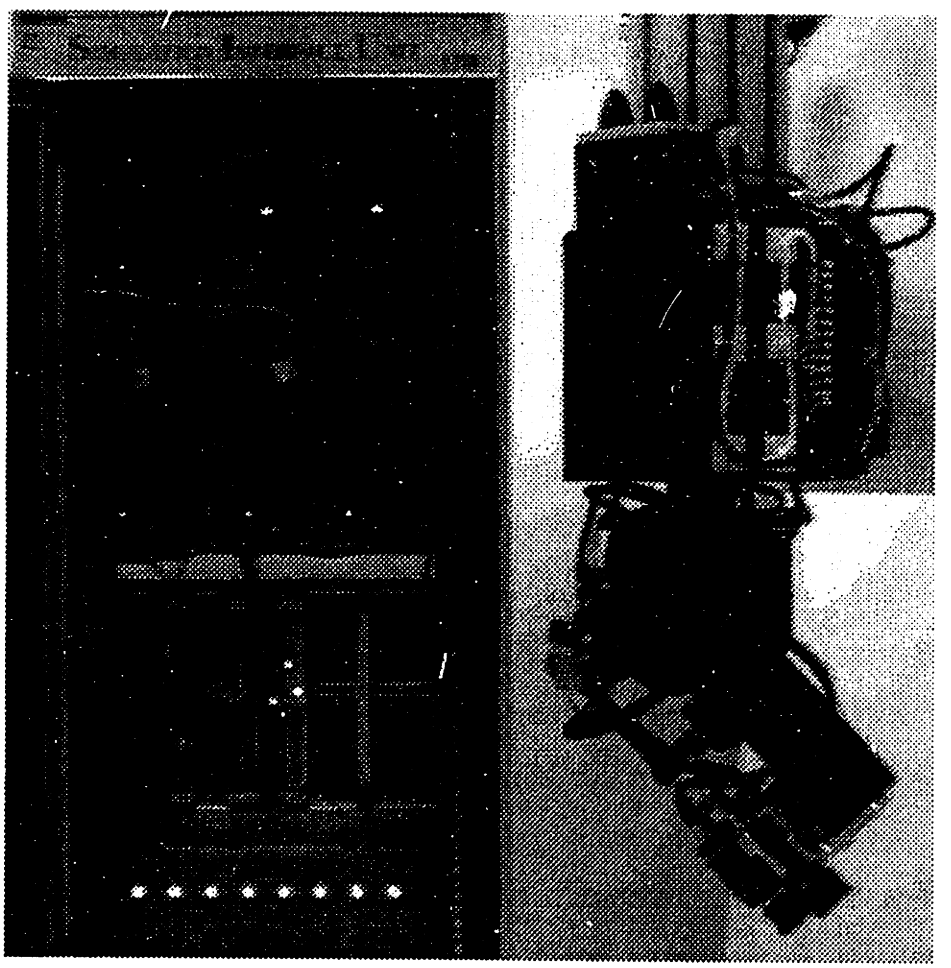

Figure 1-2: Gantry Cameras with Slip-Ring and SIU
The gantry camera and slip-ring assembly is visible in the foreground, with the Simulation Interface Unit (SIU), which holds the VME bus cards and motor controllers, visible in the background

at the end of the gantry. Both the gantry hardware and the software interface already existed. Although there are actually stereo cameras mounted on the gantry at this time, there was only a single camera when this thesis project was started, and the hardware and software to calibrate and interact with the stereo cameras is still not fully developed. As a result, only single-camera imaging methods were considered.

There are actually a wide variety of single-camera imaging methods in existence (see [Ski91, pp. 5-23] or [Fau93] for comprehensive lists). However, a lot of these methods are active imaging methods, requiring some control of, or at least knowledge of, the exact lighting situation. Such methods would require additions or alterations to the gantry hardware, and so were less preferable. After eliminating all methods which required control of lighting, there were essentially two options: stereo correspondence and optical flow.

Stereo correspondence can be carried out using a single camera by taking two different pictures, one after the other. The result of taking a picture, moving, and then taking a second picture with the same camera is almost identical to that of taking two pictures simultaneously through two different cameras. There are two major methods for solving the stereo correspondence problem, the Marr-Poggio-Grimson (MPG) method and the Pollard-Mayhew-Frisbee (PMF) method. Both methods are extremely computationally intensive, and can only work where feature points are detected. After taking a pair of images and identifying feature points, a correspondence is found between the feature points in one image and the feature points in the other image. Knowing this correspondence, it is trivial to calculate a measure of how far the feature point displaced between images. Then, using this displacement and the induced camera motion which caused this displacement, simple geometry can be used to determine the distance from the camera to the feature point.

Optical flow techniques are almost identical in method to stereo correspondence techniques. Optical flow techniques, however, are designed around the fact that the two images are taken from almost (but not quite) identical locations. Because the

image shift is known to be very small, the derivatives of the image brightness can be used to determine the image shift. This eliminates the need to identify feature points or solve the correspondence problem. The calculated image shift can then be used in the same manner as the displacement in the stereo correspondence problem in order to determine the distance between camera and object.

Skifstad develops an intensity-gradient analysis (IGA) method which seems to be a variant of optical flow (see [Ski91] for details). The mathematics is similar to that of optical flow, but Skifstad's algorithm essentially involves taking a picture and selecting a desired optical flow, then moving and taking images continuously until that flow is achieved. By doing this, Skifstad claims that the mathematics is much simpler and faster than typical optical flow methods, but the system is required to take more images.

In the application under consideration, none of the above methods are ideal. The stereo correspondence methods depend entirely on a good set of feature points. However, the lab workspace, as a simulation of the sandy-bottomed ocean where the actual UUV submarine is run, is covered with a light burlap cloth. This cloth gives a very smooth texture which doesn't produce reliable feature points. The math behind solving the correspondence problem also seemed rather complicated for something to be implemented in the scope of a Master's Thesis. The optical flow techniques are also rumored to be unpredictable on textured surfaces, such as the burlap. However, by careful analysis of the image frequencies and the optical flow equations, it seems that the texture issue can be identified and avoided (see Chapter 3 for such an analysis). Skifstad's IGA method also suffers on textured surfaces but promised to be a faster algorithm. However, because of the software filtering and frame-averaging which was needed to get clean pictures out of the CCD cameras, the algorithm would not have been the speed bottle-neck. Even for the optical flow algorithm which was eventually implemented, the slowest part of the method is capturing and cleaning up the images, not calculating the optical flow. As a result, even though the IGA algorithm

may have been faster, the overall method, which requires taking many more pictures, would have been much slower.

After considering all the options, the optical flow techniques seemed most promising. Chapter 2 describes and develops the basic method, while subsequent chapters describe analysis and variations needed to make the method work in the tested lab environment. This previous discussion is by no means meant to suggest that optical flow was the only method that would have worked, nor that it was necessarily the best choice. The discussion was, however, supposed to give some insight into why the optical flow method was chosen as the only method to be explored in this thesis.

Chapter 2

Optical Flow

This chapter develops the optical flow method of range detection. It starts with basic theory, then details the implementation used as well as specific optical flow issues. Those familiar with optical flow techniques may wish to skip the theory in Section 2.1 and may even wish to skip Section 2.2 which describes the least-squared implementation used in this thesis.

2.1 Theory

2.1.1 What Is Optical Flow

Optical Flow ranging is based on a simple assumption that, from nearly identical locations, the scene stays constant through short periods of time. By comparing two pictures of the same scene taken from slightly different locations, the algorithm identifies the motion of each part of the scene which occurs between the images. Based on the amount of camera motion between images and the amount of motion any particular point in the scene undergoes between the same two images, simple geometry can be used to determine the range to that point. Mathematically, optical

flow is based on the constant-brightness assumption, which is written as:

$$E_x u + E_y v + E_t = 0 \tag{2.1}$$

In the above equation, E is the brightness of the image, as measured by the CCD camera, E_x, E_y, and E_t are the derivatives $\frac{\delta E}{\delta x}$, $\frac{\delta E}{\delta y}$, and $\frac{\delta E}{\delta t}$ respectively, x and y are any orthogonal pair of direction vectors on the surface of the terrain, and t is the number of the image. Conceptually, it is simplest to consider t to be time, with the first image taken at time $t = 0$ and the second image taken at time $t = 1$. The other two variables, u and v, are described in terms of this time t. That is, $u = \frac{\delta x}{\delta t}$ and $v = \frac{\delta y}{\delta t}$, the x- and y-components of the optical flow. By considering t to be time, the components of optical flow can be thought of as describing the velocity of each individual point in the image. For an in-depth derivation of optical flow, see Horn [Hor86, pp. 278-293].

Note that for the rest of this chapter, it is sufficient to consider the x and y directions to be the horizontal and vertical directions in the image, respectively i and j. However, as examined in Chapter 4, lens distortion destroys any chance of having such a simple one-to-one correspondence.

2.1.2 Converting Optical Flow To Distance

After having calculated the components of optical flow, those components need to be converted into a range estimate, which is essentially a guess of the distance away from the camera along the camera axis. Because a known amount of camera motion is induced, simple geometry can be used to calculate what optical flow is expected as a result of viewing a point from any distance.

The math is far simpler when restricted to purely translational camera motion. By considering the lens of the camera to be at the origin of the world coordinate system, a translational motion of $[U, V, W]^T$ causes a point located at any location $[X, Y, Z]^T$ to move by $[\delta X, \delta Y, \delta Z]^T = [-U, -V, -W]^T$ (all in the world coordinate

system). The conversion between the world coordinate system and the coordinates of the camera can be explained through similar triangles, so that $\frac{x}{f} = \frac{X}{Z}$ and $\frac{y}{f} = \frac{Y}{Z}$ where f is the distance between the camera lens and the image plane, known as the principal distance. From this, the expected optical flow ($u_{expected}$, $v_{expected}$) can be calculated to be

$$u_{expected} = \frac{\delta x}{\delta t} = \frac{\delta}{\delta t}(f\frac{Y}{Z}) = f\frac{Z\frac{\delta X}{\delta t} + X\frac{\delta Z}{\delta t}}{Z^2}$$

which leads to

$$u_{expected} = \frac{-fU + xW}{Z}$$

Similarly, calculations for $v_{expected}$ give that

$$v_{expected} = \frac{-fV + yW}{Z}$$

These expected optical flow components can then be matched with the observed components to give the most likely distance for the point.

2.2 Implementation

2.2.1 Calculating Derivatives

Before actually trying to solve the constant brightness equation, it helps to examine the components separately. As seen in Equation 2.1, the constant brightness equation contains the optical flow components u and v which are being solved for, and the brightness derivatives E_x, E_y, and E_t. How these derivatives are calculated from the two images can make a large difference in the result. As suggested by Horn [Hor86, p. 289], it is best to calculate the derivatives for points halfway between pixels in the x, y, and t directions. Also, the effects of noise can be minimized if the same brightness measurements are involved in the calculations of all three derivatives. The derivatives should therefore be calculated as:

$$
\begin{aligned}
E_x(x,y,t) &= E(x{+}1,y{+}1,t{+}1) - E(x,y{+}1,t{+}1) + E(x{+}1,y,t{+}1) - E(x,y,t{+}1) + \\
&\quad E(x{+}1,y{+}1,t) \quad - E(x,y{+}1,t) \quad + E(x{+}1,y,t) \quad - E(x,y,t) \\
E_y(x,y,t) &= E(x{+}1,y{+}1,t{+}1) - E(x{+}1,y,t{+}1) + E(x,y{+}1,t{+}1) - E(x,y,t{+}1) + \\
&\quad E(x{+}1,y{+}1,t) \quad - E(x{+}1,y,t) \quad + E(x,y{+}1,t) \quad - E(x,y,t) \\
E_t(x,y,t) &= E(x{+}1,y{+}1,t{+}1) - E(x{+}1,y{+}1,t) + E(x,y{+}1,t{+}1) - E(x,y{+}1,t) + \\
&\quad E(x{+}1,y,t{+}1) \quad - E(x{+}1,y,t) \quad + E(x,y,t{+}1) \quad - E(x,y,t)
\end{aligned}
$$

These derivative calculations work well, but actually can be improved upon, as explored in Section 2.3.

2.2.2 Calculating Optical Flow

Theoretically, Equation 2.1 holds separately for each pixel in the image. However, for an $N \times N$ image, there are $2(N \times N)$ unknowns but only $N \times N$ equations. The constant brightness constraint alone is not sufficient to solve for the image. There are several assumptions that can be made at this point to constrain the system further. One popular assumption is that the scene being imaged is relatively smooth, and therefore that the optical flow itself is relatively smooth. This assumption leads to an iterative technique for estimating the optical flow. Another popular assumption, and the one used in this thesis, is that the image can be broken down into regions of approximately constant distance, implying that the optical flow throughout the region is constant. By assuming constant optical flow for a region, there are only two unknowns, but still a large number of equations, which can be solved by finding the least-squared-error of the constant brightness constraint. Essentially, this means solving

$$min_{w.r.t.\ u,v} \sum (E_x u + E_y v + E_t)^2 \tag{2.2}$$

for u and v, where all variables are the same as in Equation 2.1, and where the summation is over all pixels in the region of constant distance. This assumption turns out to be rather convenient for other reasons – eventually, the range estimates are all going to be assembled into a single terrain map which describes the whole

robot workspace. This map need only be accurate to about 1-inch of height on a 1-inch by 1-inch grid. This means that by selecting regions to correspond to those inch-squares in the workspace, the optical flow can be calculated by region without losing any desired accuracy. This definition for the regions has the added benefit that it does not require any filtering or averaging of the range estimates before turning them into workspace terrain data.

By swapping the order of the sums and differentiation, carrying out the minimization of Equation 2.2 leads to the two equations

$$u\sum E_x^2 + v\sum E_xE_y + \sum E_xE_t = 0$$
$$u\sum E_xE_y + v\sum E_y^2 + \sum E_yE_t = 0$$

These two linear equations can be solved simultaneously through any of several methods. One option is to use the method of determinants, which results in

$$u = \frac{\begin{vmatrix} -\sum E_xE_t & \sum E_xE_y \\ -\sum E_yE_t & \sum E_y^2 \end{vmatrix}}{\begin{vmatrix} \sum E_x^2 & \sum E_xE_y \\ \sum E_xE_y & \sum E_y^2 \end{vmatrix}} \quad v = \frac{\begin{vmatrix} \sum E_x^2 & -\sum E_xE_t \\ \sum E_xE_y & -\sum E_yE_t \end{vmatrix}}{\begin{vmatrix} \sum E_x^2 & \sum E_xE_y \\ \sum E_xE_y & \sum E_y^2 \end{vmatrix}}$$

which can also be expressed as

$$u = \left(\frac{\sum E_xE_y \sum E_yE_t - \sum E_xE_t \sum E_y^2}{\sum E_x^2 \sum E_y^2 - \sum E_xE_y \sum E_xE_y}\right)$$
$$v = \left(\frac{\sum E_xE_y \sum E_xE_t - \sum E_x^2 \sum E_yE_t}{\sum E_x^2 \sum E_y^2 - \sum E_xE_y \sum E_xE_y}\right) \tag{2.3}$$

2.2.3 Calculating Range Estimates

The u and v calculated for each region still needs to be turned into a range estimate. One simple method is to minimize the squared error between the measured and the

expected optical flows.

$$min_{w.r.t.\ Z}((u - u_{expected})^2 + (v - v_{expected})^2)$$

Using the definitions of $u_{expected}$ and $v_{expected}$ found in Section 2.1.2 to carry out the minimization leads to the equation

$$Z = \frac{(-fU + xW)^2 + (-fV + yW)^2}{u(-fU + xW) + v(-fV + yW)} \tag{2.4}$$

which is expressed entirely in terms of known and calculated parameters.

2.3 Optical Flow Issues

As shown above, using the constant brightness constraint equation can lead to an estimate of distance from two distinct images. However, the technique is prone to several problems. Some of these problems are present in many image processing techniques, but all are especially important for optical flow. While examining these problems, it is important to remember that the optical flow equations use derivative information calculated from the images. Derivative calculations tend to be extremely sensitive both to discretization (of both the domain and the range of the function) and noise. Even when using the derivative methods described in Section 2.2.1 (or the one to be developed in Section 3.4), discretization, quantization, and noise in the imaging step cause the biggest problem for the optical flow algorithm.

2.3.1 CCD Camera Issues

The general imaging issues all tend to stem from the actual CCD camera structure and physics, as well as from the environment in which the camera is used. Inherent in the construction of CCD cameras are two issues: quantization and noise – the two things which cause trouble for derivative calculations. Each cell in a CCD camera

sensor array, as a discrete component, has both finite size and finite sensing ability. Because of the finite size, the sensor does not measure values of brightness for each point on the image, but instead get an average brightness for a pixel-sized region of the image. Furthermore, this measured value is quantized to one of a small number of brightness values before being stored. Worse yet, the actually number of quantization levels which get used is affected drastically by the environmental lighting - both direction and brightness. For most non-reflecting surfaces, having the light source coming from the same direction as the camera causes much of the image to wash out - only a narrow range of the quantization levels are used. However, if the light source comes from an angle to the camera, there is more contrast in the image, so that more of the quantization levels are used. A common method for overcoming the quantization issues in the image is to use histogram modification. Histogram modification essentially stretches out the distribution of intensities to guarantee that the more extreme quantization levels are used. Although this does miracles for the visual appearance of the image, stretching out the distribution *after* quantizing does nothing for the accuracy of the measurements, thereby giving no benefit to the optical flow calculations. In fact, standard histogram methods involve a nonlinear transformation which corrupts the derivative calculations even further, instead of improving them.

An even bigger issue with CCD cameras stems from the physics behind them. Each cell in the CCD camera array actually senses and counts photon collisions with that cell. This is important because in reality the arrival of photons is very stochastic, with each photon carrying a discrete amount of energy. No matter how sensitive the CCD cell is, probability dictates that some photon quantization and random-arrival noise will show up in the image. This noise causes random graininess in the image, which is especially noticeable in darker areas of an image (where fewer photons are being collected). Again, the derivatives calculated for optical flow are very sensitive to this noise. For a thorough discussion of the problems of real-world sensors, refer to Skifstad [Ski91, 55-76].

It is impossible to avoid any of these issues entirely, but by careful choice of lighting their effects can be minimized. In this thesis implementation, careful control of lighting was made extremely difficult by the decision to use only currently-available hardware. The lights have to be far enough to the side of the camera to supply contrast while supplying enough light to reduce the stochastic photon noise. But, if the lights are too bright some image pixels may saturate, which corrupts any relative brightness measurements made using those pixels. Special care also has to be taken to insure that the motion of the camera and robot system does not cast moving shadows on the surface immediately visible in an image, as those shadows confuse the optical flow algorithms. Even with ideal lighting conditions, the CCD camera noise is still noticeable, but can be reduced by taking multiple images and averaging the results. In order to increase the averaging effect, the image can also be downsampled (using the appropriate antialiasing filters). Because downsampling occurs in both the i and j directions, downsampling by N has approximately the same averaging effect as taking N^2 pictures and averaging them together. Furthermore, downsampling actually reduces the size of the image, thereby reducing memory and time requirements for later processing steps. So, as long as downsampling does not throw out important information, it is more efficient than frame-averaging.

2.3.2 Algorithm Issues

For typical downsampling by N, an antialiasing filter with cutoff frequency $\frac{\omega}{N}$ would be used. This antialiasing filter is simply a lowpass filter which removes any high-frequency components which can not be properly represented in the new downsampled image. However, the standard antialiasing lowpass filters are not sufficient for optical flow. In fact, even without downsampling, the optical flow images should be frequency-limited below what the image could maximally represent in order to allow the discretization of derivatives to appropriately represent the actual derivative. This issue is explored in detail in Chapter 3.

A final item which proved to be an issue in the laboratory environment used for this thesis was radial lens distortion. There is basically no such thing as a perfect lens, as lenses often fail to have ideal characteristics throughout the entire image. For cheaper or wide field-of-view lenses, radial distortion often exists, but is not necessarily troublesome. When using the equipment available during the testing of this thesis, this distortion was noticeable, as were its effects on the optical flow calculations. Chapter 4 explores this issue in depth and develops an optical flow-specific algorithm for overcoming the distortion.

Chapter 3

Optical Flow Frequency Analysis

When working with data on modern computers, there is an inherent assumption that the data is discrete. For images, this means that the smallest possible distance between image features is the width of a single pixel. This translates to an upper bound on the frequencies which can be represented in the image. In discrete signal processing (DSP) terms, this maximum frequency in a discrete-space image is defined to be the frequency π *radians/sec*, and corresponds to having a row of pixels which has alternating white and black pixels repeated in every pixel pair. The general relation between the frequency ω *radians/sec* of a pattern and its period of repetition l *pixels* is $(\omega = \frac{2\pi}{l})$.

Although the pixel description of the frequencies from above makes the image appear like a square wave of frequency ω, the equivalent continuous-space image which had been appropriately frequency-limited would actually be a sinusoid. In fact, as typically done in fourier-transform analysis, an image can be considered to be made up entirely of sines and cosines (or, to make the math easier, complex exponentials and inverse complex exponentials) of the appropriate frequencies. By analyzing the effects of taking optical flow measurements of pure sinusoids with frequencies between 0 and π, the maximum acceptable frequency to be included in the image can be determined. For a discussion of Fourier analysis in images, see [Lim90, pp. 22-49].

3.1 Discretizing an Arbitrary Derivative

Mathematically, it is easiest to consider the breakdown of the image into complex exponentials such as $f(x) = e^{i\omega x}$. The discrete version is now defined to be $f(n) = e^{i\omega n}$, where n is an integer. When concentrating only on the sample points of n, this discretizing process is exact. But even though the discretization is exact, the derivatives are no longer exact. This can be seen through the Taylor expansion, which states that

$$f(x+\Delta x) = f(x) + \Delta x \frac{\delta f}{\delta x} + \frac{\Delta x^2}{2!}\frac{\delta^2 f}{\delta x^2} + \frac{\Delta x^3}{3!}\frac{\delta^3 f}{\delta x^3} + \ldots$$

Rearranging the terms gives that

$$\frac{\delta f}{\delta x} = \frac{f(x+\Delta x) - f(x)}{\Delta x} - \frac{\Delta x}{2!}\frac{\delta^2 f}{\delta x^2} - \frac{\Delta x^2}{3!}\frac{\delta^3 f}{\delta x^3} - \ldots \tag{3.1}$$

By considering this equation in the limit as $\Delta x \to 0$, the higher order terms drop out leaving the familiar definition for the derivative:

$$\frac{\delta f}{\delta x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \tag{3.2}$$

This is not the most accurate statement of the derivative, however. For a finite Δx between samples, the derivative method as discussed in Section 2.2.1 is much more accurate, and is actually akin to averaging a version of Equation 3.1 using $\frac{\Delta x}{2}$ with a version using $\frac{-\Delta x}{2}$ to obtain

$$\frac{\delta f}{\delta x} = \frac{f(x+\frac{\Delta x}{2}) - f(x-\frac{\Delta x}{2})}{\Delta x} - \frac{\Delta x^2}{2^2 3!}\frac{\delta^3 f}{\delta x^3} - \frac{\Delta x^4}{2^4 5!}\frac{\delta^5 f}{\delta x^5} - \ldots \tag{3.3}$$

which leads to the definition of the derivative as

$$\frac{\delta f}{\delta x} = \lim_{\Delta x \to 0} \frac{f(x+\frac{\Delta x}{2}) - f(x-\frac{\Delta x}{2})}{\Delta x} \tag{3.4}$$

In the limit, Equations 3.2 and 3.4, as well as Equations 3.1 and 3.3 are equivalent. However, in a discretized image, where Δx is the size of a pixel and cannot be changed, Equation 3.4 is much more precise. However, the higher order terms are still too large to be blindly discarded. From now on, this analysis will concentrate on the definition of the derivative in Equation 3.3 and on the largest discarded term.

Returning to the complex exponential $f(n) = e^{i\omega n}$, one can calculate the third derivative to be $\frac{\delta^3 f}{\delta n^3} = -i\omega^3 e^{i\omega n} = -i\omega^3 f(n)$. In order for Equation 3.4 to be accurate in the discrete world, where restricted to a finite Δn, the discarded terms must be much smaller than the retained terms. This means that

$$\left|\frac{\Delta n^2}{2^2 3!}\frac{\delta^3 f}{\delta n^3}\right| \ll \left|\frac{\left(f(n+\frac{\Delta n}{2}) - f(n-\frac{\Delta n}{2})\right)}{\Delta n}\right|$$

Plugging in the derivative from above, using the fact that

$$e^{i\omega\frac{\Delta n}{2}} - e^{-i\omega\frac{\Delta n}{2}} = 2i\sin(\omega\tfrac{\Delta n}{2})$$

and remembering that n is an element of the set of integers (implying that $\Delta n = 1$), the limit reduces to

$$\frac{\omega^3}{2^2 3!} \ll 2\sin(\frac{\omega}{2}) \tag{3.5}$$

There is no way to solve this directly, but the discretization error is known to be more prevalent at higher frequencies. Using the fact that ω^3 drops off much faster than $\sin(\frac{\omega}{2})$ at the higher image frequencies (around $\omega = \pi$), the limit simplifies to $\frac{\omega^3}{2^2 3!} \ll 2$, which further simplifies to

$$\omega \ll \sqrt[3]{48} \tag{3.6}$$

In order to translate Equation 3.6 into an actual limit on the frequency, it is important to keep in mind that the error-term is cubic in ω. To limit the error to be less than five percent of the actual measurement (five percent seems to be a reasonable

limit for "much-less-than"), ω must be less than $\sqrt[3]{\frac{1}{20}}\sqrt[3]{48}$, which means the highest frequency allowed in the image is $\omega = 1.3389$. This corresponds to a period of $\frac{2\pi}{1.3389} = 4.69$ pixels. The beginning of this chapter stated that the maximum frequency possible in the image corresponds to a 2-pixel period. However, these calculations show a necessary limit of 4.69 pixels, which is just under half of the maximum frequency. This can be achieved by low-pass filtering the image with a filter of cutoff frequency 1.3389 (with π being the maximum frequency) before performing any other processing.

3.2 Discretizing the Optical Flow Constraint Equation

The previous section concentrated entirely on the derivative in space, but the optical flow constraint equation as stated in Equation 2.1 has derivatives in space and time, and also includes the flow velocity, which is another derivative. In order to discretize the optical flow constraint equation, these other derivatives must also be discretized to reasonable values.

Because the time axis is not really time but is actually just an index for the images taken (see Section 2.1.1), the actual camera path followed between the two images is irrelevant. When considering the flow velocities, the only important information is the location of the camera when each image is taken. From there, it is easiest to assume the simplest possible motion, which is that of constant velocity. For a constant velocity, all of the higher derivatives are zero, so $\frac{\Delta x}{\Delta t} = \frac{\delta x}{\delta t}$ and the discretization of the derivative is exact.

Unfortunately, the time derivative of the brightness ($\frac{\Delta E}{\Delta t}$) is not so easy. If the image were to shift by exactly one pixel, then the brightness derivative would be identical to the spatial derivative. In this case, limiting the frequency components of the image to restrict the error in the spatial derivative would cause an identical restriction in the error of the time derivative. If the image were to shift by exactly two pixels, then the brightness derivative would be equivalent to having calculated the spatial derivative with Δn equal to 2 pixels instead of one. Because this scaling

factor takes effect before taking the cube-root in Equation 3.6, this multiplies the error term by a power of 8. So, if the image shifts by 2 pixels, it must be frequency limited half as much as calculated in in Equation 3.6 or else the magnitude of the error term for the higher frequencies would be 8 times larger than expected, which makes it comparable in amplitude to the actual derivatives.

Unfortunately, because of the nature of the optical flow imaging scheme, the exact pixel shift which will occur in an image is not known ahead of time. The above calculations suggest that this shift be as small as possible, and that anything above a full pixel may require the original image to be band-limited even further than the previous section suggests. The next section explores other issues which affect the desired pixel shift, and Section 3.4 develops a method to address these issues.

Before examining this pixel shift more, it is helpful to examine the frequency content of some actual images. Figure 3-1 shows a typical image of a sheet of burlap cloth taken from above. The fine horizontal and vertical striations in the image are due to the texture of the burlap cloth and are brought out by placing the light source to the side of the camera and far away (in this case, off the right-hand side of the image). Histogram modification was used to further emphasize the texture for display purposes. By varying the height of the camera above this surface, the size of the burlap texturing in the image changes, thereby changing the frequency content. Table 3.1 lists the frequency and period (in pixels) of this texturing for images taken at a variety of heights above the burlap cloth.

. Notice that the safe frequency limit calculated in Section 3.1 is 1.3389, which falls somewhere around the height of 16" (when downsampling by 3) . This means that the burlap texturing is of a sufficiently low frequency to allow good imaging only when the burlap is less than 16" away from the camera. Table 3.2 shows the results of some actual height calculations using the basic optical flow algorithm on the burlap scene from various distances. The images are frame-averaged by four and downsampled by three (with antialiasing) before processing to remove some of

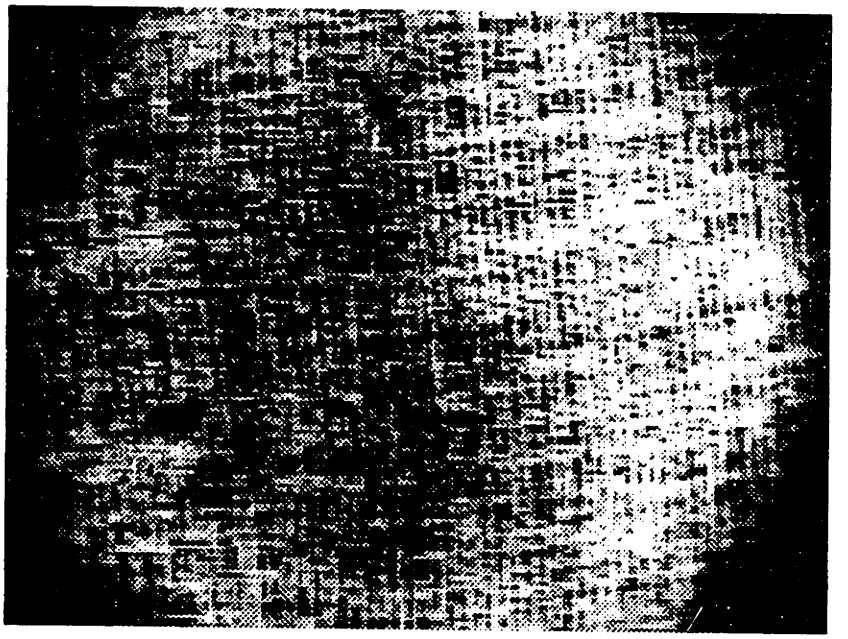

Figure 3-1: Burlap Image Showing Burlap Texturing
This histogram-modified image of burlap cloth is taken from 20 inches away, directly overhead, while downsampling by 3 and frame-averaging by 2. By varying the height of the camera above the same burlap scene, the frequency of the burlap hash can be altered. The histogram modification draws attention to the fact that the brightness drops off at the corner of the image. Careful attention is paid to not include any of these regions in the actual optical flow calculations.

Camera Height	Burlap Frequency	Burlap Pixels
10"	0.816	7.704
12"	0.997	6.303
14"	1.178	5.333
16"	1.359	4.622
18"	1.541	4.078
20"	1.661	3.782
22"	1.752	3.586
24"	1.994	3.152
26"	2.175	2.889
28"	2.477	2.537
30"	2.507	2.506

Table 3.1: Burlap Frequency and Period vs. Camera Height using 3×Downsample
At each of the specified heights, the burlap frequency is the observed frequency (with π as the maximum) of the burlap texture grid while downsampling by 3, and the burlap pixels is the corresponding number of pixels in one period of this grid.

Actual Height (inches)	Camera Shift (inches)	Measured Height (inches)	Absolute Error (inches)	Percent Error
10	0.01750	10.66	0.66	6.60
12	0.02099	13.22	1.22	10.17
14	0.02449	16.14	2.14	15.29
16	0.02799	19.56	3.56	22.25
18	0.03149	23.16	5.16	28.67
20	0.03499	26.25	6.25	31.25
22	0.03849	28.94	6.94	31.55
24	0.04199	31.14	7.14	29.75
26	0.04549	33.53	7.53	28.96
28	0.04899	37.16	9.16	32.71
30	0.05249	40.44	10.44	34.80

Table 3.2: Measured Height vs. Actual Height – Basic Method
At each of the specified heights, two images were taken, downsampled by 3 and frame-averaged by 2, using the specified camera shift between them. This camera shift was calculated to create an optical flow of exactly 0.75 pixels at the given height. At these camera shifts, the basic optical flow algorithm is used to calculate the height, and absolute and percent errors are determined. Based on the frequencies in the image (as listed in Table 3.1), the measurements should get more noisy as the height increases, which seems to be the actual case.

the noise. The processing was also restricted to a 3" by 3" square of the burlap cloth, so that the same terrain information was used in all of the images. Without shift-biasing (described in Section 3.4), this basic algorithm is rather unstable, and cannot reliably handle image shifts above approximately 1.5 pixels (due to the higher-frequency derivative errors already discussed). Because of this, a camera motion which would produce a 0.75 pixel shift in each pair of images was used. The table shows a large increase in the percent error when moving from 14" to 18" away, and also shows a general increase in the percent error throughout. When much farther than 16" from the camera, the burlap texturing starts to be filtered out by the (non-ideal) antialiasing filter, so that its contribution to the error of the optical flow algorithm does not increase much.

3.3 Optimal Optical Flow Shift

After taking two images and calculating the optical flow between them, Equation 2.4 would be used to convert this to an estimate of distance. This equation highlights that the imaging process could result in any of a range of pixels shifts, based on what distance was detected. It is important to remember that this range of pixel shifts, along with the subsequent distance estimate, is affected by the induced camera motion. The equation for converting from pixel shifts and camera shifts to distance estimate (Equation 2.4) is a single fraction – the estimate is improved when both the numerator and denominator are as large as possible. This requires large camera shifts and pixel shifts, where Section 2.1.2 shows that pixel shifts are actually linearly related to the induced camera shifts. Simply put, to get more accurate estimates for distance from the optical flow method, one should use as large a camera motion as possible.

At some point, there is a balance between the larger camera motion desired for accurate optical-flow-to-distance calculations and the small camera motion desired for accurate derivative calculations to estimate the optical flow from the original images. No matter where this point is, the magnitude of the optical flow will be limited by a certain value, with both positive and negative image shifts being acceptable. However, it is physically impossible to measure negative distances with a camera, and so it is physically impossible to have both positive and negative image shifts at the same time (the allowed sign of the image shift is determined by the sign of the induced camera motion). This means that only half of the numerically clean image shifts are physically available. In addition, the previous section highlighted that optical flows closer to zero are less susceptible to discretization error, but these flows always correspond to things far away. Typically, one knows a little bit about the imaging environment ahead of time, giving a suitable range for the distance measurements. At the very least, it is physically impossible to measure distances behind the camera, beyond walls, or through the floor. This translates to an absolute minimum (of

zero as previously mentioned) and an absolute maximum (dependent on the imaging environment) measurable distance. The next section develops a method which gives preference to the range of image shifts expected to occur while also allowing for use of the other half of the numerically clean image shifts.

3.4 Shift-Biasing the Derivative Calculations

3.4.1 What is Shift-Biasing

If an expected or average possible distance is known, the second image can be shift-biased to favor that distance. This will clean up the derivatives and subsequent optical flow calculations as described later. The first step when shift-biasing the second image is to calculate the expected optical-flow for each region of the image pair, based on what distances are expected in the image. That calculation was carried out in Section 2.1.2. However, the results in that section have a term dependent on the location in the image (x, y) and on the camera motion parallel to the camera axis (W as defined in Section 2.1.2). If W is non-zero, the expected flow is not constant throughout the region, so the shift-biasing must instead start with the average expected flow. Although the situation is simplified and improved results are obtained by restricting W to be zero (as done in the implementation in Chapter 5), it is not necessary to do so. By rounding these estimates to the nearest integer, the second image can be physically shifted by $[-u_{expected}, -v_{expected}]^T$ in memory before taking the derivatives. Then, the images can be processed as before, but will result in optical flow calculations $[u_{measured}, v_{measured}]^T$ centered near to $[0, 0]^T$ instead of being centered near $[u_{expected}, v_{expected}]^T$. Sub-pixel optical flow shifts are more precise, as stated in [DF95], and the smaller values of optical flow are more precise because of the time-derivative component as mentioned in the previous section. In addition, both positive and negative values for the optical flow are acceptable, as it is $(u_{expected} + u_{measured})$ and $(v_{expected} + v_{measured})$ which are now constrained to be always positive or always

negative. Before converting the optical flow to a distance, the expected flow can be added to the measured flow to get the real optical flow. As explained in Section 4.4, this method alone does not actually work properly for images which suffer from radial distortion, and a little extra thought is needed. To avoid this issue, the rest of this chapter assumes that there is no lens distortion in the system.

Because the real optical flow was a result of large camera motion, it benefits from having better accuracy. In addition, the bias reduces the magnitude of the calculated optical flow, reducing the error in E_t which comes from higher frequency components of the image.

3.4.2 The Mathematics Behind Shift-Biasing

In order to understand mathematically what happens when shift-biasing an image, it is helpful to examine how the shift affects the constant brightness equation (Equation 2.1). Figure 3-2 shows a one-dimensional simplification of the situation. Each plot shows the brightness contour of a specific region at two distinct instants in time. The top plot shows both the original image (Image1) and the image that results after the camera has been moved (Image2). The bottom plot shows Image1 and the shift-biased version of Image2. The spatial derivative E_n is akin to E_x and E_y in the constant brightness equation. It is obvious that the component of E_n which comes from Image1 alone is the same in the two situations, since the image itself does not change. The component of E_n from Image2 changes based on the amount of shift and the frequency components of the image. However, for images that are predominantly lower-frequencies, this component is also pretty similar between the images. For lower-frequency images or for small shifts between images, this value more closely matches the component of E_n measured from Image1. This is important because the actual derivative used (from Section 2.2.1) is the sum of the components of E_n from the two images, which more closely resembles the real derivative when the image shift is smaller.

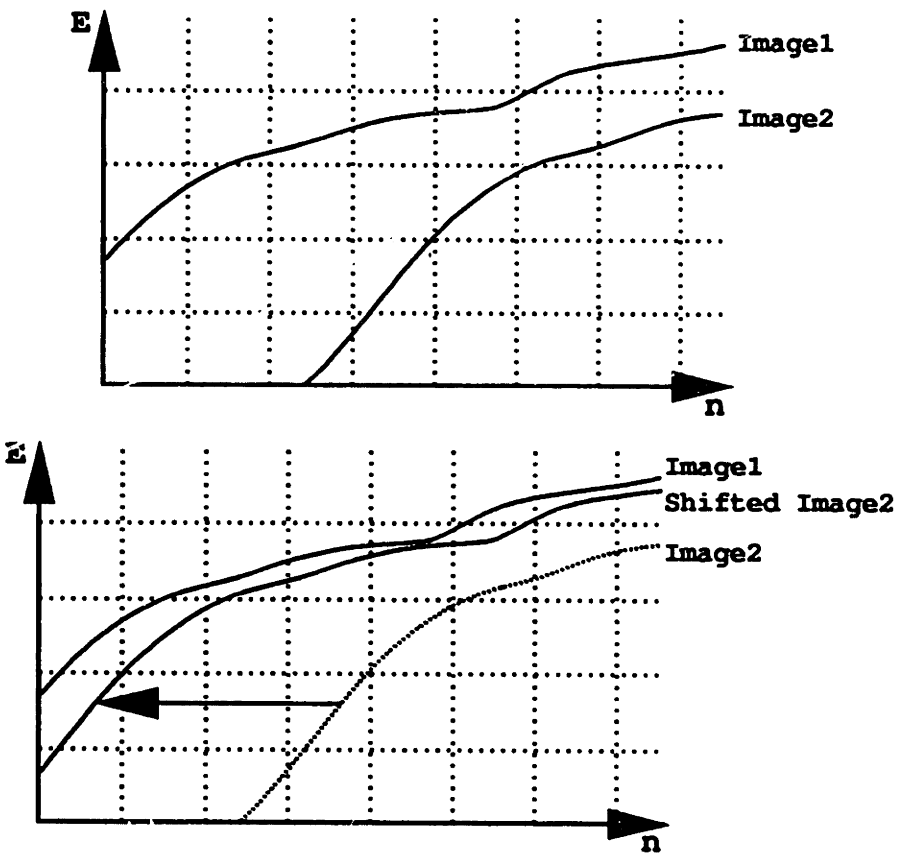

Figure 3-2: Shift-Biasing – Original Image Pair and Shifted Image Pair
In the top figure, Image1 is taken at the original location and Image2 is taken after shifting the camera.
In the bottom figure, Image2 has been shift-biased to make it more similar to Image1

Any change in E_n brought about by shift-biasing the image is overshadowed by the change in E_t. After shifting Image2, the two images are far more similar in appearance, and so the change in brightness between the two images at any one pixel is much smaller. For low-frequency images, there is little difference between solving the system with the original E_t versus solving it with the shift-biased E_t and adding the shift-bias back at the end.

The large change in E_t brought about by the shift-biasing causes the new optical flow calculations to be centered around zero. However, this change is almost exactly (but not quite) what is needed to produce a new flow estimate offset by exactly the shift-biasing amount. It is the small changes in E_n and E_t due to having slightly cleaner derivatives which makes the shift-biasing method worthwhile.

Considering the one-dimensional constant-brightness equation $E_n u + E_t = 0$ and rewriting it to use the shift-biased measurements gives $E_n^s u^s + E_t^s = 0$. In this case, E_n^s is the spatial derivative measured after shift-biasing the image, and should be equal to E_n (but a cleaner measurement), E_t^s is the time derivative measured after shift-biasing (it should be quite different from E_t), and u^s is the optical flow measured after shift-bias, which should be equal to $(u - u_{expected})$ where $u_{expected}$ is the integer-valued shift-bias used. Comparing the two equations gives that $E_t^s = E_t + E_n u_{expected}$ (or $E_t = E_t^s - E_n^s u_{expected}$). Starting again with the unshifted constant brightness equation ($E_n u + E_t = 0$) and using the shifted derivatives and the replacement for E_t gives ($E_n^s u + E_t^s - E_n^s u_{expected} = 0$). Solving this would lead to a calculation for optical flow which had the benefit of the cleaner image derivatives, and which already had the expected flow added back into it.

Section 3.4.1 stated that the proper optical flow could be calculated after shift biasing by measuring the flow in the shift-biased images and then adding the expected optical flow back in with that measured flow. However, this derivation suggests that the proper method for calculating the optical flow is to use the expected flows to alter

Actual Height (inches)	Measured Height (inches)	Absolute Error (inches)	Percent Error
10	9.60	-0.40	-4.00
12	11.59	-0.41	-3.42
14	13.50	-0.50	-3.57
16	15.07	-0.93	-5.81
18	16.49	-1.51	-8.39
20	17.93	-2.07	-10.35
22	19.65	-2.35	-10.68
24	21.44	-2.56	-10.67
26	23.22	-2.78	-10.69
28	25.08	-2.92	-10.43
30	26.67	-3.33	-11.10

Table 3.3: Measured Height vs. Actual Height – Shift-Biased Method
These measurements were made in exactly the same manner as those in Table 3.2, except that shift-biasing was used when calculating the derivatives. Notice that these measurements are much closer than those without the shift-biasing, and that the expected trend of increased error with increased frequency content is still prevalent.

E_t through the equation

$$E'_t = E_t - E_x u_{expected} - E_y v_{expected} \tag{3.7}$$

This new E'_t should then be used instead of E_t in the constant brightness equation (Equation 2.1) and in the subsequent derivations in Section 2.2.2. When considering lens distortion in Chapter 4, it is necessary to use this method of altering the expression for E_t.

3.4.3 Examining Shift-Biasing Results

In order to test out the effects of shift-biasing, the imaging tests carried out in Section 3.2 were repeated with the shift-biased optical flow algorithm. The results of these new tests, shown in Table 3.3, show a marked improvement in the height estimates.

It is interesting to note that the height estimates in the original situation were

all too large, whereas those in this situation are all too small. In general, when the algorithms fail, they tend to estimate optical flows which are smaller than the actual flow. In the original case, that always results in a smaller overall optical flow, which results in a farther distance. However, when using shift-biasing, the result is dependent on the bias. In these tests, the cameras were moved so as to produce an optical flow of 0.75 pixels. The shift biasing step requires an integer bias, so uses an expected flow of 1 pixel. Failure of the algorithm in this case produces an estimate which is closer to the "expected" bias flow of 1 pixel, and is therefore larger than the actual flow. This results in distance estimates which are too small, as seen in the table.

Chapter 4

Optical Flow and Camera Lens Distortion

There are many kinds of camera and lens distortion which can show up in an imaging system, and the types that afflict an imaging system depend entirely on the hardware in use. After examining a few images in the test environment used for this thesis, it became apparent that the system suffers from radial distortion, as well as from a few other shortcomings. The radial distortion is rather noticeable, and can be seen simply by pointing the camera at any straight-edged surface and observing the bow in the imaged version. It is a common problem for cheap lenses, and has to be adjusted for. The other ailments seem to manifest themselves only at the image edges and corners, and so are easier just to discard. These edge effects include the fact that the image intensity drops off drastically within about 20 pixels of the corner, and also that the frame-grabber used for this thesis fails to grab parts of the edge 10 pixels on several sides of the image. The intensity-drop seems to be another lens-related issue, and could be fixed by scaling the intensity of each image pixel based on its location in the image. However, the missing pixels in the frame-grabber can't be adjusted for, so the bad rows and columns must be discarded. It is much easier (and doesn't lose much information or accuracy) just to discard parts of the bad corners at the same time

and to ignore any other distortions which are retained. Davis and Freeman [DF95] mention a strategy for removing the image brightness drop-off, which they refer to as fixed-pattern noise. Though a worthwhile addition, the method was not added to this thesis, as it seemed likely to have only a small effect on the overall distance estimate.

4.1 What is Radial Distortion

Essentially, the radial distortion issue is the only one that needs special attention. Radial distortion is a product of cheap lenses where the magnification is different at the edges from at the center of the lens. Because of lens shapes, radial distortion is also a function of the lens's field of view – radial distortion is usually not an issue for telephoto lenses but is quite prevalent in wide-angle lenses. Although a function of the radius, the effects of the distortion is noticeable through most of the image. Discarding the affected areas would therefore leave little of the image left for processing.

There are two kinds of radial distortion: barrel distortion, which occurs when the off-axis magnification is weaker than at the center, and pin-cushion distortion, which occurs when the opposite is true. The lens used in this thesis suffers from barrel distortion, which affects an image as shown in Figure 4-1. The pattern on the left is an undistorted pattern of concentric circles and grid lines. The right image shows what barrel distortion would do to this pattern. Radial distortion is most easily understood in a polar coordinate system, with the origin somewhere near the image center. Objects at small radii appear as they should, but further from the center of the lens and image the perceived radius is smaller than the actual radius. At the same time, polar coordinate angles are unaffected, and so circles whose center corresponds to the center of the image remain circular (with a slightly smaller radius) and lines through the center of the image remain straight. Lines which do not pass through the image center tend to bow, however.

Lens aberrations and distortions have been know about for quite some time. There

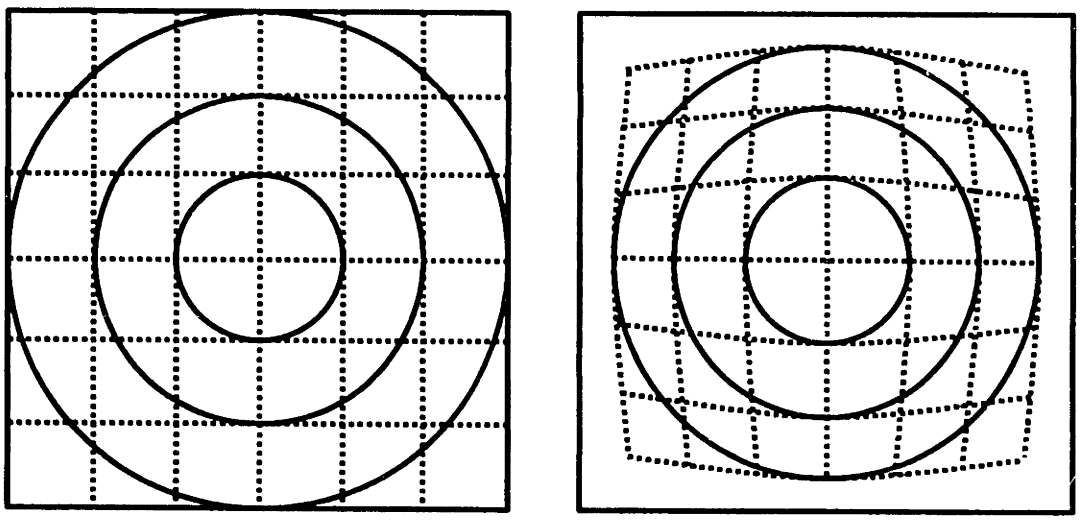

Figure 4-1: Radial Distortion Example
The left image is an undistorted grid pattern, and the right image is the same pattern viewed after radial distortion

is a reasonable list of them in The Focal Encyclopedia of Photography, 1965 Edition [KK65, p. 5]. This encyclopedia includes both kinds of radial distortion and states that their effect on images is a function of the image radius cubed. Polynomial approximations of the radial distortion are known to follow the form

$$r = \rho + \alpha\rho^3 + \beta\rho^5 + ... \tag{4.1}$$

where r is the undistorted image radius, ρ is the radius in the distorted image, and α and β are coefficients of radial distortion. Only odd powers of ρ exist, and the distortion can usually be approximated using only the first and third power of ρ. In addition, although the center of the radial distortion may not be exactly at the center of the image, using the image center seems to be sufficient for the purposes of this thesis. Wolf [Wol74] explains some methods used to calculate and correct for radial distortion in aerial photography, and Stein [Ste93] describes a method used in a lab environment.

By some simple algebraic manipulation, Equation 4.1 can also be written as

$$\rho = r - \alpha_2 r^3 - \beta_2 r^5 + ... \tag{4.2}$$

where the variables all stand for the same as in Equation 4.1, but the radial distortion coefficients may differ. This algebraic manipulation, known as reversion of series, is covered in several standard math texts [AS64]. This equation can again be approximated using only the first and third power terms. In both the original equation and its reversion, if the radial distortion is barrel distortion, as seen in Figure 4-1 and the test apparatus, α is greater than zero. In fact, α is the same for both equations. On the other hand, β and the subsequent coefficients do not have to be any particular sign, nor are they likely to be the same in both equations. The exact relationship between the coefficients is covered by Abramowitz and Stegun [AS64, p. 16]. However, for the precision desired in this thesis, all terms beyond α can be discarded.

The next section develops a method to correct for radial distortion but shows that even the approximated equations

$$r = \rho + \alpha\rho^3 \tag{4.3}$$

and

$$\rho = r - \alpha r^3 \tag{4.4}$$

are a little too complex to use separately in correcting the optical flow equations. However, a reasonable system can be developed by using the two equations together. Before continuing, however, it is important to note that for barrel distortion, the terms discarded to make Equation 4.4 are far more important than those discarded to make Equation 4.3. Without those discarded terms, Equation 4.4 is actually physically impossible. As shown in Figure 4-2, as the $-\alpha r^3$ term gets big enough, the distorted radius ρ actually reaches a maximum and then doubles back. In an imaging

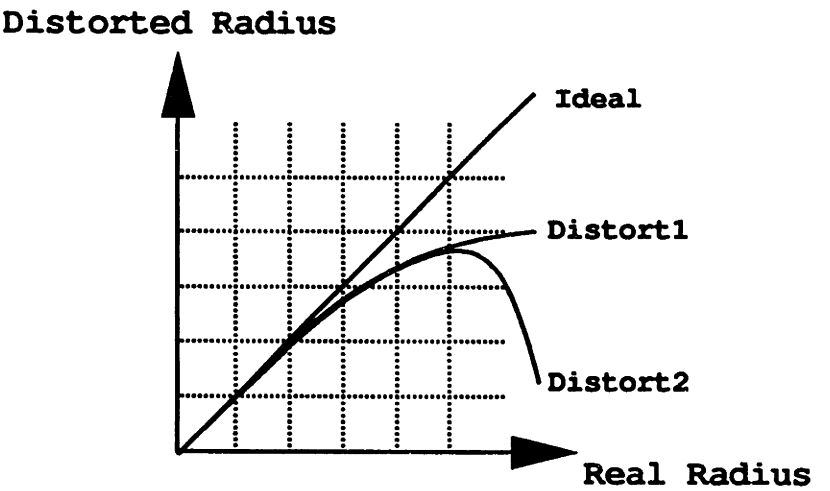

Figure 4-2: Radial Distortion Approximation Methods
Approximating using the equation $r = \rho + \alpha\rho^3$ (Distort1)
compared to approximating using the equation $\rho = r - \alpha r^3$ (Distort2)
For small enough α and r, the approximations are almost identical, but at large α or r, the approximation from the second method is physically impossible

situation, this would correspond to having an object which is actually further away from the center of the scene appear to be closer to the center of the image. In contrast, Equation 4.3 represents a more reasonable physical situation, where the function relating ρ to r is monotonically increasing everywhere.

Although a lot of effort was taken to avoid using Equation 4.4 because of this realization, it turns out that the "ρ as a function of r" form of the radial distortion equation is easier to work with. Fortunately, in the lens and camera system used to test this thesis, the actual α is small enough that this discrepancy is not an issue inside the image. However, the point where this does become an issue is only just beyond the edge of the image.

4.2 Removing Radial Distortion

Although the radial distortion problem is noticeable when looking at whole images, the problem is hard to detect when looking at local regions of the image. This is especially true for the optical flow calculations, which are interested only in relative

distances, not absolute locations. Empirically, for any small region, the optical flow system is only off by a few percent. However, this few percent causes obvious errors in the distance estimate which manifest themselves in the same way for each image pair. Regions of the image which are far away from the center always appear to have smaller-than-expected optical flow measurements, and as a result the distance estimate is too large. When this effect is mapped to the whole image, it makes the edges of the image fall away from the camera – if the image is of a flat surface, the optical flow and subsequent distance calculations make it appear like the top of a hill.

The general solution for this problem is to undistort the image before processing. Because the value of α is a product of the lens instead of the image or scene, the value can be determined by careful analysis of a test image similar to that shown in Figure 4-1. For a complete description of how to determine the radial distortion (among other distortions) through such methods, see Stein [Ste93, pp. 18-21]. By using this value for α and one of distortion equations presented in the previous section, the image can be put through a new distortion which should correct for the radial distortion of the lens. Unfortunately, the discretized samples occur on a regular grid in the distorted image, and would therefore follow an irregular grid on the corrected image. In order to use the corrected image, a regular grid must be superimposed over the distorted one, with interpolation used to find the actual values. The interpolation process tends to be computationally intensive and also introduces quite a bit of noise into the image, which causes problems in later processing. Shirai gives a brief overview of how to resample an image, including any interpolation which might be necessary [Shi86].

4.3 Distorting the Optical Flow Equations

Instead of using the generic solution of distorting the image a second time, another option is to distort the optical flow equations. Remember that the constant brightness equation (Equation 2.1) holds for the undistorted image, the undistorted flow is used in Equation 2.4 to estimate the distance, any derivative measurements are obtained

from the distorted image. So, instead of working with the equation $E_x u + E_y v + E_t = 0$, it should be rewritten into something of the form $A(E_i, E_j)u + B(E_i, E_j)v + E_t = 0$, where i and j represent the orthogonal coordinate system in the distorted image but u and v are still undistorted pixel shifts.

In order to make use of this equation, the functions A and B need to be determined. In addition, the final form of these functions must be something which can be used to solve for α as well as using them to solve for the optical flow. Introducing the four derivatives needed to convert from the $[x, y]$ coordinate system to the $[i, j]$ coordinate system and expanding the derivatives for E_x and E_y gives

$$E_x = \frac{\delta E}{\delta x} = \frac{\delta E}{\delta i}\frac{\delta i}{\delta x} + \frac{\delta E}{\delta j}\frac{\delta j}{\delta x} = E_i\frac{\delta i}{\delta x} + E_j\frac{\delta j}{\delta x} \tag{4.5}$$

Likewise,

$$E_y = \frac{\delta E}{\delta y} = \frac{\delta E}{\delta i}\frac{\delta i}{\delta y} + \frac{\delta E}{\delta j}\frac{\delta j}{\delta y} = E_i\frac{\delta i}{\delta y} + E_j\frac{\delta j}{\delta y} \tag{4.6}$$

Using these expansions, Equation 2.1 can be rewritten as

$$(E_i\frac{\delta i}{\delta x} + E_j\frac{\delta j}{\delta x})u + (E_i\frac{\delta i}{\delta y} + E_j\frac{\delta j}{\delta y})v + E_t = 0 \tag{4.7}$$

4.3.1 Calculating Optical Flow

This distorted version of the constant brightness equation is not yet usable, since the four coordinate transformation derivatives ($\frac{\delta i}{\delta x}$, $\frac{\delta j}{\delta x}$, $\frac{\delta i}{\delta y}$, and $\frac{\delta j}{\delta y}$) are still unknown. Fortunately, they can be determined from the radial distortion equation presented in the previous section. However, in order to do so, it is necessary to pick which version and which approximation of the radial distortion equation to use. Although many different approximations of Equations 4.1 and 4.2 and methods for deriving the distorted optical flow equations can be used, the simplest one which ended up with reasonable results is outlined below. Other methods attempted are outlined in Appendix B.

The two coordinate systems are defined below. The coordinates x, y, and r correspond to measurements in the theoretical undistorted image, while i, j, and ρ correspond to measurements in the distorted image as seen through the camera-lens system. Because the polar coordinate angle is not affected by radial distortion, θ is the same in the two coordinate systems.

$$\begin{array}{lll} x = r\cos\theta & y = r\sin\theta & r^2 = x^2 + y^2 \\ i = \rho\cos\theta & j = \rho\sin\theta & \rho^2 = i^2 + j^2 \end{array}$$

Eliminating θ from the above equations gives that $\frac{x}{r} = \frac{i}{\rho}$ and $\frac{y}{r} = \frac{j}{\rho}$, which can be written as $x = \frac{r}{\rho}i$ and $y = \frac{r}{\rho}j$. The ratio between r and ρ is determined by the radial distortion equation used. Starting with Equation 4.4, the relations between the distorted and undistorted image can be written as

$$\begin{array}{rcl} \frac{\rho}{r} & = & 1 - \alpha r^2 \\ i & = & \frac{\rho}{r}x = x(1-\alpha r^2) = x - \alpha(x^3 + xy^2) \\ j & = & \frac{\rho}{r}y = y(1-\alpha r^2) = y - \alpha(x^2 y + y^3) \end{array}$$

From here, the four necessary derivatives can be determined.

$$\begin{array}{lll} \frac{\delta i}{\delta x} = 1 - \alpha(3x^2 + y^2) & \frac{\delta i}{\delta y} = -\alpha 2xy \\ \frac{\delta j}{\delta x} = -\alpha 2xy & \frac{\delta j}{\delta y} = 1 - \alpha(x^2 + 3y^2) \end{array}$$

Plugging these derivatives into Equations 4.5 and 4.6 gives

$$\begin{array}{rcl} E_x & = & (1 - \alpha(3x^2 + y^2))E_i - 2\alpha xy E_j \\ E_y & = & (1 - \alpha(x^2 + 3y^2))E_j - 2\alpha xy E_i \end{array}$$

but these equations are still a function of x and y. The summations of Equation 2.3 are performed in the coordinate system of the actual (distorted) image, which are the i and j coordinates. Rewriting the above equations in terms of i and j gives

$$\begin{array}{rcl} E_x & = & E_i - \alpha(\frac{r}{\rho})^2((3i^2 + j^2)E_i - 2ijE_j) \\ E_y & = & E_j - \alpha(\frac{r}{\rho})^2((i^2 + 3j^2)E_j - 2ijE_i) \end{array}$$

which is yet again a function of the nonexistent undistorted image because of the

reference to r. Expressing $\frac{r}{\rho}$ directly from Equation 4.4 fails to remove the appearance of r in the equation, and even though there are ways to solve Equation 4.4 for r as a function of ρ, these equations tend not to be simple polynomials in α. This is not important when solving for the optical flow, but it is very important when actually solving for α. The simplest mathematical solution to this dilemma, and one that yields very nice results, is to use Equation 4.3 as if it were an approximation for Equation 4.4. This gives that $\frac{r}{\rho} = 1 + \alpha\rho^2$, which leads to

$$\begin{aligned} E_x &= E_i - \alpha(1+\alpha\rho^2)^2((3i^2+j^2)E_i - 2ijE_j) \\ E_y &= E_j - \alpha(1+\alpha\rho^2)^2((i^2+3j^2)E_j - 2ijE_i) \end{aligned} \tag{4.8}$$

This can be solved using exactly the same method as in Section 2.2.2 to again yield Equation 2.3, but with E_x and E_y as defined above.

4.3.2 Calculating Alpha

In order to solve for the optical flow in this way, there is now one more piece of information needed than was originally the case – the value of α. This can be obtained by solving a least-squared minimization of the constant-brightness assumption with respect to α. Technically, this should be solved at the same time as minimizing for u and v. However, it is much simpler to derive an estimate of u and v to use when solving for α, and then use that α to solve for the correct u and v. Recalling that α is a function of the lens and not the actual image scene, it is possible to calculate α from any image pair. Realistically, it is much easier to calculate α from an image pair of a flat surface which is perpendicular to the camera axis and parallel to the camera motion. In such a situation, the distance to this surface, as well as the optical flow caused by it, should be constant throughout the image. As a result, a small central region of the screen can be used to calculate the optical flow (where the radial distortion is unnoticeable). Then, this optical flow can be used as the expected flow for the entire image, and α can be calculated by plugging the derivatives from

Equation 4.8 into Equation 2.1 and performing a least-squared minimization of the result. The new form of the constant brightness equation is

$$(E_i u + E_j v + E_t) - [E_i u(3i^2 + j^2) + E_j v(i^2 + 3j^2) + (E_j u + E_i v)2ij]\alpha[1 + \alpha(i^2 + j^2)]^2 = 0 \tag{4.9}$$

Consider this equation to be $A + B\alpha + C\alpha^2 + D\alpha^3 = 0$ with A, B, C, and D set appropriately from above and solve the minimization for α

$$min_{w.r.t.\,\alpha} \sum (A + B\alpha + C\alpha^2 + D\alpha^3)^2$$

which results in

$$\begin{aligned} &\sum(AB) + \sum(2AC + BB)\alpha + \sum(3BC + 3AD)\alpha^2 + \\ &\sum(4BD + 2CC)\alpha^3 + \sum(4CD)\alpha^4 + \sum(3DD)\alpha^5 \quad = \quad 0 \end{aligned}$$

Although there are no guaranteed ways to analytically solve a fifth order equation, it is relatively easy to solve it numerically using Newton's method. Not only is it a relatively straightforward equation, but the physical reality of radial distortion guarantees that the solution for α is a small positive number.

4.4 Accounting for Shift-Biasing

Shift-biasing is the more precise derivative calculation method derived in Section 3.4. In the process of developing this shift-biasing method, it was assumed that there was no radial distortion in the image. This came about by assuming that a shift of $[-u_{expected}, -v_{expected}]^T$ in the second image could be reversed through an optical flow adjustment of $[u_{expected}, v_{expected}]^T$. However, when taking radial distortion into account, that is certainly not the case. The image shift of $[-u_{expected}, -v_{expected}]^T$ occurs in the distorted i and j coordinate system, while the optical flow adjustment of $[u_{expected}, v_{expected}]^T$ occurs in the undistorted x and y coordinate system.

The end of Section 3.4.2 develops a more complex method for adjusting the optical flow calculations to account for the expected flow. This method can be used to account for shift-biasing and radial distortion at the same time. Equation 3.7 shows how to adjust the time derivative to account for the expected flow, and still holds in the presence of radial distortion. However, the equation is written with the derivatives and expected flows all calculated in the undistorted coordinate system. Rewriting the equation with the expected flows spelled out to emphasize this fact gives

$$E'_t = E_t - E_x(\tfrac{\delta x}{\delta t})_{expected} - E_y(\tfrac{\delta y}{\delta t})_{expected} \tag{4.10}$$

When considering radial distortion, the equation should be in terms of the measurable variables E_i, E_j, $(\frac{\delta i}{\delta t})_{expected}$, and $(\frac{\delta j}{\delta t})_{expected}$ instead of the undistorted variables. As with Equations 4.5 and 4.6, the undistorted flows can be expressed in terms of the distorted flows as

$$\begin{aligned}(\tfrac{\delta x}{\delta t})_{expected} &= \tfrac{\delta x}{\delta i}(\tfrac{\delta i}{\delta t})_{expected} + \tfrac{\delta x}{\delta j}(\tfrac{\delta j}{\delta t})_{expected} \\ (\tfrac{\delta y}{\delta t})_{expected} &= \tfrac{\delta y}{\delta i}(\tfrac{\delta i}{\delta t})_{expected} + \tfrac{\delta y}{\delta j}(\tfrac{\delta j}{\delta t})_{expected}\end{aligned}$$

Combining these equations and Equations 4.5 and 4.6 into Equation 4.10 and using the facts that

$$\tfrac{\delta i}{\delta x}\tfrac{\delta x}{\delta i} + \tfrac{\delta i}{\delta y}\tfrac{\delta y}{\delta i} = 1 \qquad \tfrac{\delta j}{\delta x}\tfrac{\delta x}{\delta i} + \tfrac{\delta j}{\delta y}\tfrac{\delta y}{\delta i} = 0$$
$$\tfrac{\delta i}{\delta x}\tfrac{\delta x}{\delta j} + \tfrac{\delta i}{\delta y}\tfrac{\delta y}{\delta j} = 0 \qquad \tfrac{\delta j}{\delta x}\tfrac{\delta x}{\delta j} + \tfrac{\delta j}{\delta y}\tfrac{\delta y}{\delta j} = 1$$

gives the very interesting result that the adjustment is unchanged. Specifically, the distorted replacement for the time derivative is simply

$$E'_t = E_t - E_i(\tfrac{\delta i}{\delta t})_{expected} - E_y(\tfrac{\delta j}{\delta t})_{expected} \tag{4.11}$$

By using this equation to replace E_t along with the replacements for E_x and E_y found in Equation 4.8, Equation 2.3 can be solved while accounting for both shift-biasing and radial distortion.

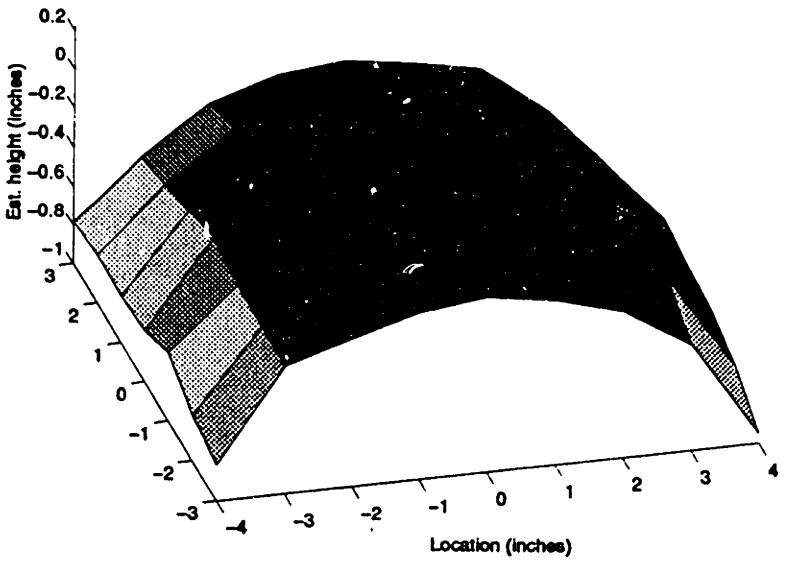

Figure 4-3: Terrain Reconstruction of Flat Surface – Basic Method
This is a terrain reconstruction for a flat surface using the basic optical flow algorithm, without accounting for radial distortion. In this method, the flat terrain appears more like a hill, which falls away at points away from the image center.

4.5 Examining Radial Distortion Correction Results

Adjusting for radial distortion involves two steps, that of calculating α and that of using α to correct the optical flow equations. Because the calculation for α technically only has to be done once, this section focuses on using α to adjust the optical flow equations and totally ignores the issue of calculating α. Unfortunately, however, α is not entirely fixed for an imaging system – it is a function of the focus of the lens itself. For that reason, α must be redetermined every time the lens is refocused.

Figure 4-3 displays a terrain reconstruction of estimated distances as calculated by the basic optical flow algorithm. The image was taken at a height of 20 inches, and the image distances were subtracted from the camera height to calculate the actual terrain height. The effects of radial distortion are clearly visible in the terrain reconstruction, where the distance estimates fall off at points further away from the center of the image.

The distance falloff is more pronounced in the x-direction because the camera

motion is entirely in the x-direction. By moving the camera in the x-direction, the optical flow is calculated by taking the distance between two points with almost identical y locations but with different x locations – the distortion visible is a function of the change in radius in the image, irrespective of angle, but that change in radius due to a constant change in x is not constant throughout the image.

Because the falloff in the x-direction is more prominent, it is simpler to analyze two-dimensional cross-sections in the x-direction alone. Figure 4-4 compares the falloff in the x-direction when using a variety of different α values. The distance estimate displayed is not actually a cross-section, but is instead an average of the y-direction values, to reduce the effects of any erroneous measurements. Also, for sake of comparison, the terrain reconstructions were all adjusted so that the center-most height estimate of the terrain reconstruction (before averaging the y-direction values) was zero. In the figure, it appears that the error in height estimate scales almost linearly with α. Negative α would make the problem worse, and values of α above 2×10^{-6} obviously over-correct for the situation.

As mentioned at the beginning of this section, the radial distortion correction involves not only correcting for the radial distortion, but also calculating the correct α to use for such a correction. The α calculated for the terrain image used in this section is shown as a dotted line in Figure 4-4. A look at the statistics given in Table 4.1 shows that this calculated α value ($\alpha = 1.74 \times 10^{-6}$) has a very low sum of squares term, and so is a reasonable guess for the actual parameter. The calculated value is probably slightly off because of approximations made earlier in this Chapter, which were made in order to have a radial distortion compensation which could be minimized in terms of α as well as in terms of the optical flow. This value is still much closer than any values calculated by the other approximation methods described in Appendix B. Figure 4-5 shows the full terrain reconstruction using this calculated α. A comparison to Figure 4-3 shows how effective the radial distortion adjustment can be.

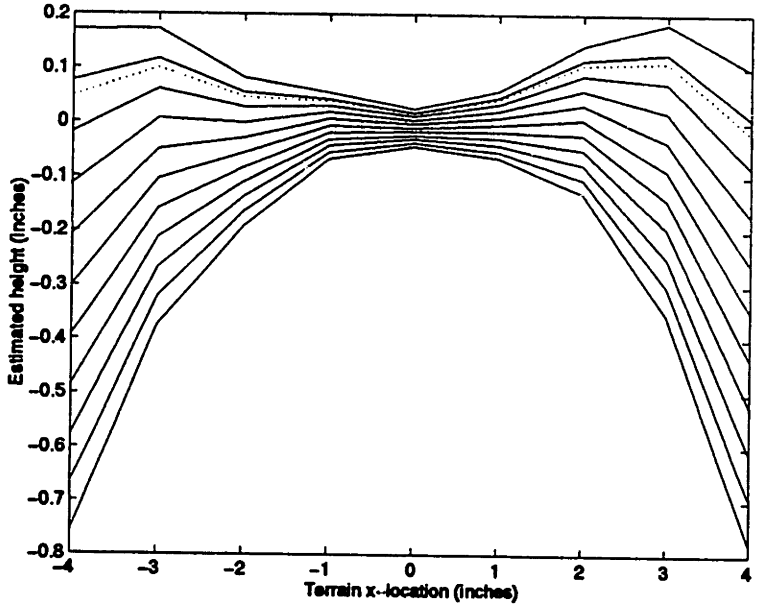

Figure 4-4: Terrain Reconstruction Cross-sections vs. α
These are cross-sections of terrain reconstructions for a flat surface using the radial distortion flow estimate and various α values. The horizontal axis is the terrain floor x-location in inches, and the vertical axis is the estimated height. The bottom plot corresponds to $\alpha = 0$, which is equivalent to using the basic equations without radial distortion adjustments. The top plot corresponds to $\alpha = 2 \times 10^{-6}$. Intermediate plots result incrementing α by 2×10^{-7} each time. The dotted plot corresponds to $\alpha = 1.74 \times 10^{-6}$, which is the value of α estimated by the routines.

α	Minimum (inches)	Maximum (inches)	Average (inches)	Std. Dev. (inches)	Sum Sqrs. (inches^2)
0.00×10^{-6}	-0.95126	0.03001	-0.30674	0.27882	10.82537
0.20×10^{-6}	-0.85885	0.03142	-0.26617	0.24938	8.38136
0.40×10^{-6}	-0.76560	0.03283	-0.22537	0.21988	6.24556
0.60×10^{-6}	-0.67152	0.03824	-0.18434	0.19037	4.42409
0.80×10^{-6}	-0.57661	0.04646	-0.14309	0.16101	2.92317
1.00×10^{-6}	-0.48085	0.06640	-0.10161	0.13206	1.74905
1.20×10^{-6}	-0.38424	0.08637	-0.05989	0.10405	0.90808
1.40×10^{-6}	-0.28677	0.10637	-0.01795	0.07831	0.40668
1.60×10^{-6}	-0.18845	0.14012	0.02422	0.05834	0.25134
1.74×10^{-6}	-0.11914	0.18465	0.05386	0.05182	0.35195
1.80×10^{-6}	-0.08927	0.20383	0.06662	0.05179	0.44861
2.00×10^{-6}	-0.02504	0.26799	0.10925	0.06339	1.00513

Table 4.1: Image Statistics Based on α
For each value of α, the minimum, maximum, average, standard deviation, and sum of squares were collected from the terrain reconstructions. The images were taken from a height of 24 inches above the burlap surface. The error term for the imaging is more like the sum of squares than the standard deviation, which shows that the correct α should probably have been closer to 1.6×10^{-6}. However, the calculated α is still a pretty good estimate.

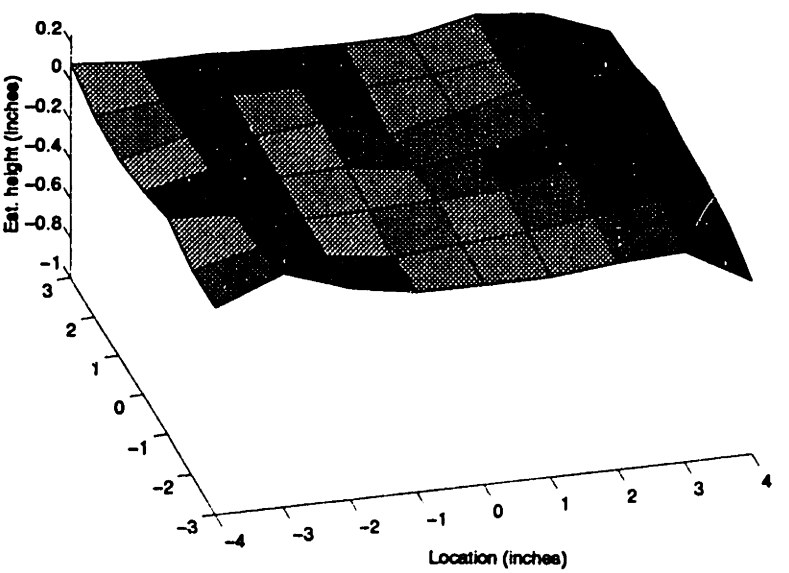

Figure 4-5: Terrain Reconstruction of Flat Surface - Removing Radial Distortion
This is a distance estimate for a flat surface using the radial-distortion adjusted optical flow algorithm and the value of α determined by that algorithm ($\alpha = 1.74 \times 10^{-6}$). Comparisons to Figure 4-3 show a marked improvement in the surface estimate.

Chapter 5

Complete Terrain-Mapping System

Developing a reasonably robust range-sensing algorithm was only part of the driving force behind this thesis. As mentioned in Chapter 1, this work was done at C. S. Draper Lab in order to implement an automatic terrain-mapping system for a specific lab environment. The primary driving force behind this research was therefore to create a working utility for the lab. Although this thesis, so far, has concentrated on developing a reasonably robust general range-sensing algorithm using optical flow techniques, the specific lab environment has always been a consideration. Specifically, the focus of the research has constantly been defined by the issues encountered when using the existing hardware to image the sculpted lab floor. This chapter will now focus on the terrain-mapping system developed, and how it functions in its lab environment.

5.1 Overall Range Sensing System

The process for producing a range estimate in the lab is a rather long sequence of simple steps.

- The camera is moved to the location to be imaged, $24"^{\dagger}$ above the ground, facing down.
- The first image is taken $2^{\dagger}$ times and frame-averaged together.
- The first frame-averaged image is filtered and downsampled by $1^{\dagger}$, resulting in an image with upper frequency of $1.3389^{\dagger}$.
- The camera is moved by $0.125"^{\dagger}$ in the x-direction†.
- The second image is taken $2^{\dagger}$ times and frame-averaged together.
- The second frame-averaged image is filtered and downsampled by $1^{\dagger}$, resulting in a second image with upper frequency of $1.3389^{\dagger}$.
- The estimated height of the camera is used to divide the image into 6" by 8" of regions†, each corresponding to $1"$-squares† of the terrain map.
- The derivatives of each region are calculated and stored, accounting for shift-biasing based on the estimated height.
- The range to each of these regions is calculated and stored in the terrain map.
- The last three steps are repeated once†, using the newest range estimates to divide the image into regions and determine the shift-biasing for the regions.

Note that the flagged items(†) were choices made to produce the best results possible. Descriptions of the tests and analysis used to determine these values are detailed in Appendix C.

Thanks to the flexibility of the simulation framework software, many of the specifics mentioned above are changeable at run-time. The camera height, amount of frame-averaging and downsampling, amount and direction of camera motion between images, number of regioning iterations, and size of the terrain-squares being mapped are all configurable. In addition, whether the derivative calculations should account for shift-biasing or not, as well as the actual method for determining range is also selectable. As noted with the flag(†), many of these values were chosen from test results described in Appendix C. However, other options do work and are occasionally used for the mapping process.

5.2 Converting Range Data to Terrain Data

The steps outlined above will produce range estimates for 6" by 8" of the environment. In order to produce a complete terrain map of the approximately 70" by 140" inches of terrain, this imaging procedure needs to be repeated many times. To do so, the routine starts at one corner of the workspace and calculates range estimates for the first 6" by 8" block. Then it moves the camera to a spot exactly 8" over and calculates a new batch of range estimates. By repeating this all of the way to the other end of the gantry workspace, the mapper produces a terrain strip which is 6" wide and covers the full length of the workspace. It then moves the camera over that 6" and calculates range estimates as it moves the camera back towards the first side of the workspace. By continuing to range in a zig-zag pattern, the terrain-mapper can cover the whole workspace in 12 strips of 18 images each. During this search pattern, the program is capable of stopping and re-imaging a section if it decides that it failed to image properly the first time through, which occurs if an excessively noisy image is captured or if someone walks nearby and casts a shadow on the scene at exactly the place and moment that a picture is taken. The level of tolerance for this re-imaging is user-definable, but even the most stringent limits usually only result in about 5 re-imaging decisions during a run. The entire 216 (or more) iterations of the range estimation process take about an hour in total to complete. Much of this time is spent waiting for the gantry to move and waiting for the camera imaging, frame-averaging, and down-sampling. An example of the overall results, using the options described above, is shown in Figure 5-1.

5.3 Conclusions

The overall system works very well in the lab environment used as a test, as seen in Figure 5-1. However, this success is not as complete as it could have been. As seen through Chapter 3, the camera should be kept within a certain distance of the

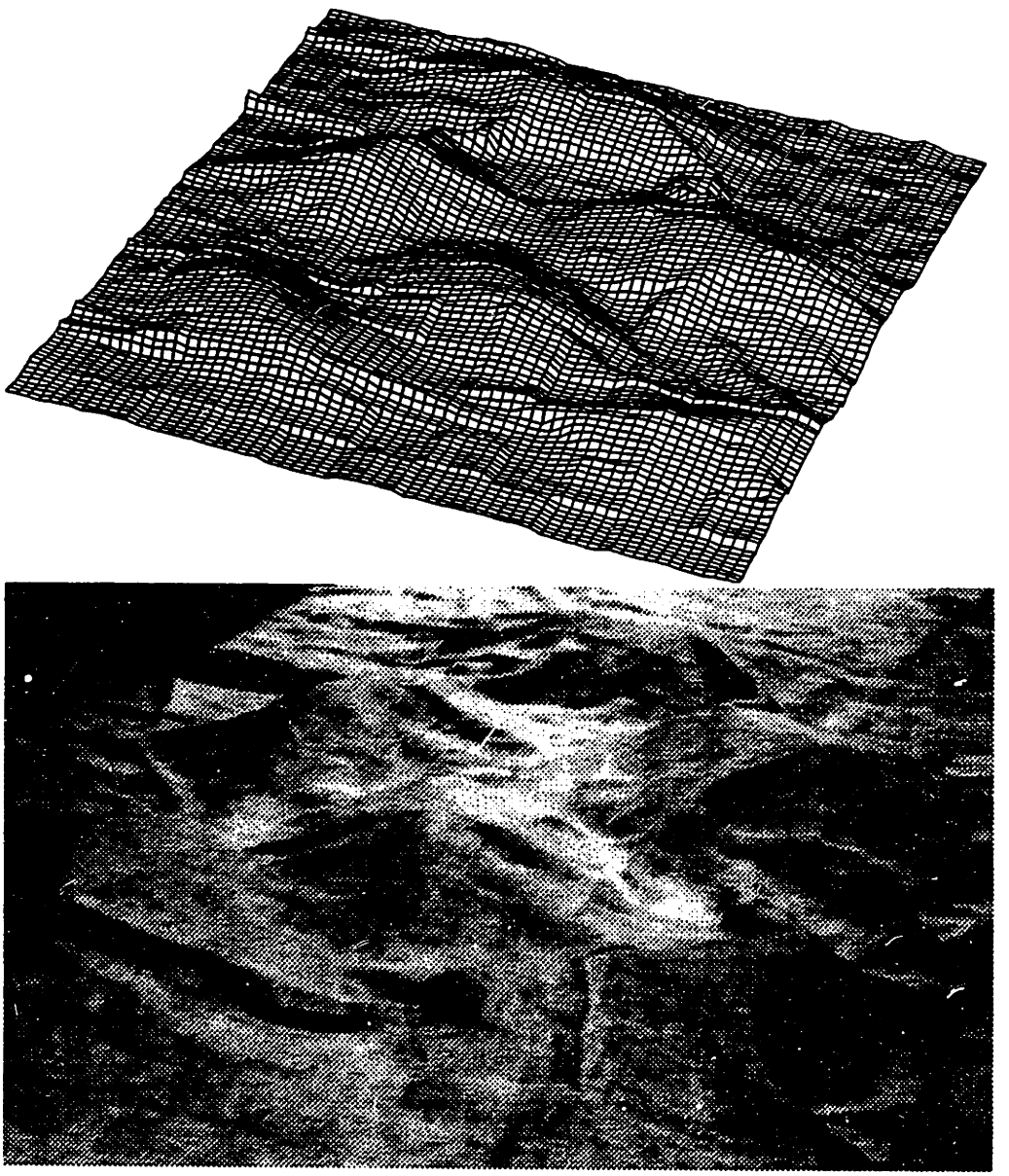

Figure 5-1: Reconstructed Terrain vs. Actual Terrain
The top image is a terrain reconstruction of the terrain pictured in the bottom image. The terrain construction is approximately 5 feet by 10 feet, and was created by piecing together the results of 143 different image pair calculations of 6" by 8" regions, taken from a height 24" above the ground while frame-averaging by 2 and without downsampling

surface for reasonable terrain mapping. Table 3.1 shows this distance to be 16" while downsampling by 3, and Table C.2 shows this distance to be 24" while downsampling by 1. This need for close distances, coupled with the inability to change the camera lens focus during a run, means that the terrain can not have too much height variation, or else it sometimes gets out of focus. Focus blur is equivalent to low-pass filtering, and often removes the burlap texture frequency which is responsible for much of the success of the range estimates. As seen in Table 1.1 the workspace is 35" high. However, focus blur limits the routine such that any terrain can only be sculpted to use about one third of the workspace height. Even with this restriction, the success of the algorithm is dependent on the surface and the lighting. The parameters have been picked to work with the burlap cloth (as a simulated sandy terrain) and with the lab lighting as currently exists. but there is no guarantee that the same parameters would work if real sand were used or if the ceiling lighting configuration were changed. If such changes did occur, the method is robust enough and the aforementioned parameters flexible enough that parameter changes alone should be enough to handle the new situation (by using different amounts of downsampling, different camera height and step size, external light sources, etc.). Finally, the hour needed to gather all of the terrain information is rather long, though still acceptable. Future research could be focused at improving the robustness of the system and at speeding up the overall time needed to run the whole terrain-mapping utility.

In general, this thesis developed a successful routine to perform terrain mapping of a large surface through optical flow techniques. Chapter 2 developed the basic least-squared algorithm to be used, as well as presenting the general issues. Chapter 3 explored the issue of the frequency component of images and presented some limits to these frequency components. It also developed a methodology by which the errors caused by high-frequency components can be reduced. Chapter 4 explored the most prevalent issue of the real-world non-ideal sensor used in testing this thesis – radial distortion. In that chapter, the optical flow equations were redeveloped to account for

this distortion. Finally, this chapter has explored the use of the optical flow terrain mapping routine to map a specific lab environment.

Appendix A

Framework Overview

The Simulation Framework is a tool designed to aid in the development and execution of vehicle simulations. After a brief overview, this appendix will outline the following aspects of the Framework:

- The Database Preprocessor
- User Interface Features
- Framework Features

A.1 Source

The original version of this Simulation Framework description was written as an appendix to a thesis proposal prepared by Tim Kalvaitis on August 9, 1993.

A.2 Overview

The Framework is designed to aid in the development of simulation code written in C. It consists of a preprocessor that generates run time information and an execution environment that makes use of the generated information. The preprocessor is used to create a hierarchical organization of simulation variables and relate them to underlying C structures that are used in the vehicle model code. The execution environment gives access to these variables and provides the user with the ability to manipulate, log, plot, and drive their values.

A command interface is used to control the simulation at run time. Commands can be issued through any of the following mechanisms:

- File based command scripts (input files)
- Terminal based console

- X windows based console

In addition to commands, a Motif based graphical user interface has been implemented to display, monitor, and modify variables. An on screen dynamic plotting facility has also been implemented to monitor logged variables with PostScript output of plots available via command. A Form facility has been developed for creating Motif user interface elements and attaching them to simulation variables.

Other run time features include:

- Data Logging
- Plots
- Unlogging – Driving simulation variables from previously save data
- Profiling – Driving simulation variables from a tabular time history
- Runtime Interpreter – A C-like language for manipulating simulation variables during computation
- Snapshot – A tool for creating a table of current simulation variables values
- Process Control – Real time control, processor allocation, process priority specification
- GL Graphics support – Update control, animation, etc.
- Distributed Shared Memory – Memory-sharing between processes, frameworks, and machines
- I/O Interface – Generic setup for hardware-in-the-loop simulation

Simulation Framework features are controlled by simulation variables themselves and are fully configurable from the command line, input files, and browse windows.

The following will provide more detail of the Framework features.

A.3 Database Preprocessor

The database preprocessor is responsible for creating the runtime information used by the execution environment. All simulation variables are organized in a hierarchy similar to a UNIX file system. The hierarchy definition is created through the use of specification (spec) files. A spec file contains the definition of directory types and variable names. A directory specification is similar to a C structure declaration with a few exceptions: all directory variables have an initialization value, units, and a description. The directory specification is usually followed by a directory name. This name is then used to create a global C structure. The type of the variable

is the C equivalent to the directory specification. Three files are generated by the preprocessor. The first is the reference (ref) file. The ref file acts as a header file for C source files. It contains the structure declarations generated by each directory specification, and external references to any simulation directories defined in the spec file. The definition of these directories is contained in the definition (def) file. The def file initializes the directories with the initializers specified in the spec file. The database (db) file contains all information necessary to manipulate the variable at run time including the name, type, units, description, and address of each variable.

Model code is created by including the ref file as a header and referencing the directory names as C structure variables and the simulation variables as members of the C structure. Variables are referenced at run time by using the same name and a colon. For example, the directory called '`sim`' contains the variable '`t`'. In C the variable is referenced as `sim.t` and at run time as '`sim:t`'. The value for both is obtained at the same location in memory.

The database preprocessor was designed to allow access to C variables at run time without imposing an efficiency penalty. There is never a need to call a function to access a model variable so efficiency is maintained. In addition, the run time information directly references the location in memory of the C structure for each variable. This allows direct manipulation of any model variable by the user at run time.

Related database features include:

- Run time allocation of directories and simulation variables
- Memory mapping of directories to arbitrary addresses

A.4 User Interface

There are many ways to communicate with the Framework. The basis of communication is through commands. The console process is responsible for interpreting the commands and recording the result. All commands entered into the simulation are recorded in a log file. A command may be issued in any of the following ways:

- File based command scripts (input files)
- Terminal based console
- X windows based console

All commands may be placed in an input file for execution. This allows entirely scripted simulation runs to occur. Command aliases can also be created, to work like aliases in a shell environment. Most commands allow for UNIX-like pattern matching when specifying simulation variables thereby easing command specification. In addition to file based input, a terminal style console may be used to enter commands

directly, or a Motif, X windows console may be used. Console features include command history and command line editing. The X console additionally includes:

- Cut and paste capability
- A scrolling record of the command log
- A configurable status bar for display of simulation time and other key pieces of information
- A configurable push button bar for frequently used commands (also tied to function keys)

Several browse windows may be used as a mechanism for traversing the directory hierarchy. The browse window provides graphical access to each simulation variable. Features include:

- Direct viewing and manipulation of all simulation variables
- Menu selection of C enumeration types
- Automatic unit conversion after a change of units
- Emacs-like incremental search for variable names
- Interactive control of browse window display parameters

Similarly, there are several specific browsers for examining arrays, directory-arrays, savesets, variable-views, and a generic memory browser for examining memory in other formats. These windows all provide the same features as above, but allow access to multi-dimensional arrays, to arrays of directories, to logged data, to non-hierarchical organizations of the directory structure, or to any location in the program's memory space.

Several plot windows are available for the display of simulation plots. Each window accesses a plot list and contains both variable versus time and variable versus variable plots. Features include:

- Strip chart-like scrolling during simulation computation.
- Automatic axis limits determination (based on data)
- Aspect ratio maintenance
- Multiple variables per plot
- Symbols on data points

- User controlled pan and zoom
- Data point tracking by the mouse
- PostScript hard copy via keystroke

The number of plots is limited only by memory constraints. All plots may be sent for hardcopy regardless of whether the plot windows are used for display.

A Time monitor window is used to display time information. Features include:

- Animated analog or digital clock
- Display of simulation time and real time
- Menu driven interface for time control

Forms may be used for run time, user definable windows. A specification language has been created to allow the user to specify all Motif/Xt user interface elements and their resources. Custom elements have been created to provide access to simulation commands and variables. Current interface elements available to the user include:

- Updated labels and text fields connected to simulation variables
- Push buttons connected to simulation commands
- Slider bars connected to simulation variables
- All Motif widgets

Given the proliferation of windows in the Framework interface, a subwindow manager has been created to keep track of them. Features include:

- Opening and Closing of all Framework windows
- Iconification of individual windows
- Iconification of all windows to one icon at once
- Push button window representation

Current work for the extension of the user interface includes the incorporation of a generic mechanism for creating dials and meters connected to simulation variables using GL animation. This will be based on work done on the T38 cockpit simulation.

A.5 Framework Features

There are many features of the Framework in addition to the user interface features. Some have been mentioned in the context of the User Interface and will be expanded on here. Addition of features of the Framework include the following:

- Data Logging
- Plots
- Unlogging
- Profiling
- Runtime Interpreter
- Snapshot
- Process Control
- GL Graphics support
- Distributed Shared Memory
- I/O Interface

Data logging allows the user to save the value of any variable in the simulation. Logged data are organized on log lists. With the logging facility one can:

- Track all simulation variables over time
- Set logging rates per log list
- Specify a time interval on which to log
- Specify a condition for logging using the run time interpreter (see below)
- Save logged data to a file
- Read previously logged data from a file

In addition to run time logging features, several separate programs are available to convert logged data files into other file formats as well as converting other formats to the Framework format.

The plotting facility is built on the logging facility. A simulation variable may be plotted versus time or another simulation variable. Multiple curves may be put on each plot. Several plots are grouped in a plot list. There is no imposed limit on the number of plots that may reside on a plot list. In addition to plotting logged data as a

simulation progresses, past data can be displayed on the same plot as currently logged data for reference. This is made possible through the use of the logging facilities ability to read previously logged data. Hard copy plotting is achieved through the generation of a PostScript file. One need never display a plot on screen to take advantage of the plotting facility. An entire job may be run from a file in a background process with plots issued at the end of the run.

Unlogging refers to the ability to take logged data and drive simulation variables. While unlogging, no model computation takes place, but all displays are updated. This allows the replay of graphical representations in a way that is not compute bound. In addition, real time rates control the speed at which unlogging takes place allowing smooth animation using previously computed data.

Profiling is similar to unlogging in that simulation variables are driven from previously known data. Profiling, however, allows the specification of drive data in tabular format. The user is able to specify time intervals and values for individual simulation variables as well as conditions to be met before taking on a particular value. This is useful for creating input functions for the simulation models.

The Runtime Interpreter is a C-like language for manipulating simulation variables during the compute process. Before every simulation tick is performed, a list of Runtime Interpreter programs is executed. Each program, specified from an input file, contains semi-compiled C code to perform every simulation tick. Currently supported constructs include:

- If, do, while, and for statements
- All C unary and binary expressions
- A library of built in math functions (similar to the standard C math library)
- Integers, float and double types for constants, simulation variables, and built in functions

With the Runtime Interpreter, code may be quickly checked without the need to compile in C. In addition to calls every simulation tick, the Runtime Interpreter is used for expression variables and the calc command. The calc command provides a quick means to carry out calculations at the command line. Expression variables are simulation variables containing Runtime Interpreter code that is evaluated to obtain a value. This allows a variable to be set equal to some combination of other variables and consequently saved and plotted.

The Snapshot facility uses a specification file to create a table of current simulation variable values. The user can specify a page format, run the simulation to a certain point, and snapshot the result to a file. This facility is also used to generate strings for display elsewhere in the Framework. For example, the run number on a plot is generated with a snap line that evaluates a simulation variable via snapshot and places the result in a string for display.

Process control refers to the Framework's ability to divide the computation and user interface into multiple processes. Each process may be assigned to a particular processor and given a priority. This allows maximum hardware utilization. Often, the compute process will be assigned to a separate processor from the graphics in order to maintain speed. Graphics updates occur only on an as needed basis and when there is enough processor power. When assigned to a processor other than the compute processor, the graphics do not interfere with the models.

GL graphics support is available in the Framework through the use of a mechanism to control animation as well as through process control. In addition to GL drawing, user input is integrated into the Framework through both the GL input and X windows input mechanisms. Support for SGI's dial box, button box and Spaceball are also provided.

The distributed shared memory facility allows multiple frameworks running on the same or different machines to transparently share their data. Built on the database feature of memory mapping directory structures to specific locations in the computer, this feature allows the same physical memory location to be included in the directory structure for multiple simulations. As a result, either sim can edit or display the memory as with any other variable. In the case of over-the-network communication, each machine keeps a local copy of all memory, but network-wide locks are used to maintain consistency over all machines.

The same memory mapping techniques are used to provide a generic hardware I/O interface to the framework. Currently capable of supporting any VME bus hardware, the interface allows reference to the hardware as if it were simply another variable in the directory structure, allowing for all inputs to be examined, plotted, saved, and edited.

Please note that the distributed shared memory and I/O interface features of the framework were both implemented as part of a thesis in Electrical Engineering and Computer Science at MIT (Tim Kalvaitis, BS and MS May '94 [Kal94]). More information on those features is available in the thesis itself.

Appendix B

Discarded Radial Distortion Methods

Radial distortion is introduced and explained in Chapter 4. However, as described in that chapter, there are several possible implementations and approximations which can be used to remove this distortion. Some of the possible implementations are outlined below.

Throughout this appendix, x, y, and r refer to undistorted image coordinates, and i, j, and ρ refer to the distorted coordinates. The distorted coordinates are those seen in the actual image, whereas the undistorted coordinates are those that would have been seen using a perfect lens. Equation 4.7 always holds, and states that

$$(E_i\frac{\delta i}{\delta x} + E_j\frac{\delta j}{\delta x})u + (E_i\frac{\delta i}{\delta y} + E_j\frac{\delta j}{\delta y})v + E_t = 0$$

It is important to remember that the goal is to find expressions for the four coordinate transformation derivatives ($\frac{\delta i}{\delta x}$, $\frac{\delta j}{\delta x}$, $\frac{\delta i}{\delta y}$, and $\frac{\delta j}{\delta y}$). However, any method used must be minimizible in both the optical flow components u and v as well as any radial distortion coefficient α because it is needed both to calculate α and also to calculate the optical flow using the predetermined α.

As outlined in Chapter 4, there are two basic methods for expressing the radial distortion. The first is by representing ρ as a function of r, as in Equation 4.2. The other is by representing r and a function or ρ, as in Equation 4.1. These two methods have been referred to below as methods 1 and 2 respectively. Variants on each method have been given letters based on the order they were coded, so that the first attempt at a method 1 radial distortion adjustment scheme is named method 1A. In every case, I chose to use the same approximation for $\frac{r}{\rho}$ when calculating α and when calculating u and v. Although we may be able to use a better approximation to the ratio after we've solved for α, it seems beneficial to try to use exactly the same approximations in both parts of the problem, so that the variable α is being used in exactly the equations for which it was solved. As it turns out, the method for calculating u and v from α is rather forgiving, so that most of the techniques perform

identically in this regard. It is the result of solving for α which is more affected by which method is used. The next few sections describe all of these schemes, and the final section gives a little comparison between them as a means of choosing the best one.

B.1 Radial Distortion Schemes – Method 1

Method 1 schemes are all based on the equation

$$\rho = r - \alpha r^3 + ...$$

with $\alpha > 0$. The necessary derivatives for this method are

$$\begin{array}{rclrcl} \frac{\delta i}{\delta x} &=& 1-\alpha(3x^2+y^2) & \frac{\delta i}{\delta y} &=& -\alpha 2xy \\ \frac{\delta j}{\delta x} &=& -\alpha 2xy & \frac{\delta j}{\delta y} &=& 1-\alpha(x^2+3y^2) \end{array}$$

These derivatives are simple, but refer to the ideal (and therefore unmeasurable) variables x and y instead of measurable variables i and j. In order to convert x and y to i and j, we need to use the ratio $\left(\frac{r}{\rho}\right)^2$ as shown below. Unfortunately, this ratio is again a function of the ideal variables x and y, and so needs to be approximated.

$$\begin{array}{rclrcl} \frac{\delta i}{\delta x} &=& 1-\left(\frac{r}{\rho}\right)^2\alpha(3i^2+j^2) & \frac{\delta i}{\delta y} &=& -\left(\frac{r}{\rho}\right)^2\alpha 2ij \\ \frac{\delta j}{\delta x} &=& -\left(\frac{r}{\rho}\right)^2\alpha 2ij & \frac{\delta j}{\delta y} &=& 1-\left(\frac{r}{\rho}\right)^2\alpha(i^2+3j^2) \end{array}$$

Method 1A used the approximation $\left(\frac{r}{\rho}\right)^2 = 1$ and solves the minimizations normally.

Method 1B used the approximation $\left(\frac{r}{\rho}\right)^2 = (1+\alpha\rho^2)^2$ which comes directly from the method 2 equation. The minimization can be solved normally. This is actually the method finally selected, and so is described in more detail in the actual text.

A reasonable way to obtain an expression for r in terms of ρ is by adding and subtracting powers of r from some overall sum. To start, we use the equation for ρ in terms of r, and then we add the equation for ρ^3 in terms of r, to remove the larger of the undesired terms of r. Eventually, you are left with a sum of powers of ρ which approximately equals r.

$$\begin{array}{rcl} \rho &=& r-\alpha r^3 \\ +\alpha\rho^3 &=& +\alpha r^3 - 3\alpha^2 r^5 + 3\alpha^3 r^7 - \alpha^4 r^9 \\ +3\alpha^2\rho^5 &=& +3\alpha^2 r^5 - 15\alpha^3 r^7 + 30\alpha^4 r^9 - 30\alpha^5 r^11 + 15\alpha^6 r^13 - 3\alpha^7 r^15 \end{array}$$

Continuation of this process gives that

$$r = \rho + \alpha\rho^3 + 3\alpha^2\rho^5 + 12\alpha^3\rho^7 + 55\alpha^4\rho^9 + 258\alpha^5\rho^11 + ...$$

Methods 1B2 and 1B3 were introduced, which simply used further terms in the approximation for $\frac{r}{\rho}$, but these new methods produced no noticeable improvements, and were essentially the same as the original method 1B, and so were discarded.

Method 1C used the approximation $\left(\frac{r}{\rho}\right)^2 = \frac{1}{1-2\alpha\rho^2-3\alpha^2\rho^4}$ which comes from rearranging the original equation $\rho = r - \alpha r^3$ into $\frac{r}{\rho} = \frac{1}{1-\alpha r^2}$ and then squaring to get $\left(\frac{r}{\rho}\right)^2 = \frac{1}{1-2\alpha r^2+\alpha r^4}$. This is a function of r, not ρ, and so is not yet usable. Method 1C comes from using the expansion for r calculated under method 1B above, and plugging it into this equation and keeping only the first few terms. However, in order to perform the minimization, we must "cheat" by multiplying through by the denominator before performing the minimization sums. This is quite an extreme cheat, as it turns out, because the multiplication happens for all the terms at one location of i and j, but then a different denominator is multiplied through the next set of terms for the next i and j.

Methods 1C2 and 1C3 were introduced. Method 1C2 simply used further terms in the expansions for r, giving the equation $\left(\frac{r}{\rho}\right)^2 = \frac{1}{1-2\alpha\rho^2-3\alpha^2\rho^4-10\alpha^3\rho^6-50\alpha^4\rho^8}$. Method 1C3 substituted a truncated approximation for r in the ratio *before* squaring, giving the equation $\left(\frac{r}{\rho}\right)^2 = \frac{1}{1-2\alpha\rho^2-3\alpha^2\rho^4+4\alpha^3\rho^6+4\alpha^4\rho^8}$.

These new approximations for $\frac{r}{\rho}$ produced no noticeable improvements, and since methods 1C2 and 1C3 were essentially the same as the original method 1C, they were discarded.

Method 1D used the approximation $\left(\frac{r}{\rho}\right)^2 = \frac{1+\alpha\rho^2+3\alpha^2\rho^4}{1-\alpha\rho^2-2\alpha^2\rho^4-7\alpha^3\rho^6}$ which comes from combining methods 1B and 1C. The ratio $\left(\frac{r}{\rho}\right)^2$ is considered as $\frac{r}{\rho} \times \frac{r}{\rho}$, with the first ratio expanded using method 1B and the second ratio expanded using method 1C. Because there are now terms discarded from both the numerator and the denominator, the overall approximation seems to be better. However, you must again cheat by multiplying through by the denominator before summing.

Methods 1D2 and 1D3 were introduced, which used more terms in the expansions. Method 1D2 expanded the numerator and denominator to the same power of α, resulting in the equation $\left(\frac{r}{\rho}\right)^2 = \frac{1+\alpha\rho^2+3\alpha^2\rho^4+12\alpha^3\rho^6}{1-\alpha\rho^2-2\alpha^2\rho^4-7\alpha^3\rho^6}$ Method 1D3 expanded the numerator and denominator using the same expansion for r in each case, giving the equation $\left(\frac{r}{\rho}\right)^2 = \frac{1+\alpha\rho^2+3\alpha^2\rho^4+12\alpha^3\rho^6}{1-\alpha\rho^2-2\alpha^2\rho^4-7\alpha^3\rho^6-30\alpha^4\rho^8}$

These new approximations for $\frac{r}{\rho}$ again produced no noticeable improvements, and since methods 1D2 and 1D3 were essentially the same as the original method 1D, they were discarded.

B.2 Radial Distortion Schemes – Method 2

Method 2 schemes are all based on the equation

$$r = \rho + \alpha\rho^3 + \ldots$$

with $\alpha > 0$. The necessary derivatives for this method are

$$\begin{array}{rclcrcl} \frac{\delta i}{\delta x} & = & R \times (1 + \alpha(i^2 + 3j^2)) & & \frac{\delta i}{\delta y} & = & R \times (\ -\alpha 2ij) \\ \frac{\delta j}{\delta x} & = & R \times (\ -\alpha 2ij) & & \frac{\delta j}{\delta y} & = & R \times (1 + \alpha(3i^2 + j^2)) \\ R & = & (1 + 4\alpha\rho^2 + 3\alpha^2\rho^4)^{-1} = \frac{1}{(1+\alpha\rho^2)(1+3\alpha\rho^2)} & & & & \end{array}$$

As with methods 1C and 1D, we now have a function of α as a denominator, which can not be easily solved for using least-squared minimization. Cheating by multiplying through by the denominator at the start is so bad that it produces a negative guess for α, which we know is not correct (a negative guess would mean pincushion distortion, but the image is clearly visible as barrel-distortion). This method was clearly unacceptable.

One solution is to write our method 2 equation as ρ in terms of r using the same method we used for method 1C above. By adding and subtracting higher terms of ρ, you eventually get the equation

$$\rho = r - \alpha r^3 + 3\alpha^2 r^5 - 12\alpha^3 r^7 + 55\alpha^4 r^9 - 258\alpha^5 r^{11} + ...$$

In order to simplify this equation, we can approximate the subsequent terms to be a geometric series, where each term is $G\alpha r^2$ times the previous term. G is a constant which can be picked as needed, but matlab minimization of the error between the exact equation and the approximation showed that $G = -3.48$ was the closest value to use. This gives the equation

$$\rho = r - \alpha r^3 + 3\alpha^2 r^5 \left(\frac{1}{1 - G\alpha r^2}\right)$$

which can then be differentiated with respect to α. Unfortunately, using the infinite sum formula again introduces an α in the denominator, so that it can not be minimized normally.

By rewriting the equation in Method 1 format, we have also reintroduced the method 1 problem that we now have an equation in terms of r, the ideal radius, instead of ρ, the actual radius. This is easy to fix, however, since the ratio $\frac{r}{\rho}$ can be determined from the method 2 equation to be exactly $1 + \alpha\rho^2$, and so does not require any of the approximations which were required for method 1.

Method 2A used the approximation $\frac{1+\alpha\rho^2}{1-G\alpha r^2} = 1$ so that we didn't have any α in the denominator, and because the term on top seemed to be of a similar magnitude and order to that on the bottom.

Method 2B used the exact expression for $\frac{1}{1-G\alpha r^2}$. However, in order to solve the minimization, an iterative scheme was used, where the previous calculation for α was plugged into the α in the denominator. This turned the denominator into a scalar quantity which could be handled in the minimization. This iteration scheme was very stable, usually converging by a couple of powers each iteration (so that if a certain iteration was accurate to 4 significant figures, the next iteration was accurate to 6

significant figures).

Because of the success of the iterative method, I went back to the original method 2 derivatives, (r as a function of ρ) and tried iterating that directly.

Method 2C used the exact method 2 derivation without any approximations, iterating over α by considering the α in the denominator to be a constant before taking the derivative. Not only was it rather unstable, but it usually ended up with a negative guess for α, which we know is not correct.

Method 2D again used the exact method 2 derivation, but calculated the derivative analytically before fixing any denominator α variables as constants and iterating. The result was better than that for method 2C, but was still unstable for several of the image pairs tested.

Method 2E again used the exact method 2 derivation with the derivative analytically calculated before fixing any denominator α and iterating. However, there was a term of the form $\frac{1}{(1+\alpha\rho^2)(1+3\alpha\rho^2)}$ which was rewritten to put more dependence on α in the numerator by expressing it as $1 - \frac{4\alpha\rho^2+3\alpha\rho^4}{(1+\alpha\rho^2)(1+3\alpha\rho^2)}$. This made the iterative method much more stable again.

In all cases, by iterating over α by replacing denominator values of α with the previously calculated value, we condemn ourselves to calculate values of α which are too small. This is because the denominator, which is of the form $(1+\alpha\rho^2)(1+3\alpha\rho^2)$, is smaller for smaller α. A smaller denominator would yield a bigger error term when divided out, but by fixing the denominator at a specific value during the minimization, this effect is removed.

Method 2E2 was introduced to improve the minimization step and remove the favoritism shown to small α values. To do so, I introduced a term

$$\frac{(1+\alpha_{iter}\rho_{avg}^2)(1+3\alpha_{iter}\rho_{avg}^2)}{(1+\alpha\rho_{avg}^2)(1+3\alpha\rho_{avg}^2)}$$

into the minimization, after summing but before calculating the minimum. In this equation, ρ_{avg} is an average value for ρ in the area that was summed and α_{iter} is the previous guess for α which is being used. This way, we reintroduce the appropriate function of α in the denominator for the minimization. If the calculated α were identical to the previous iteration value (α_{iter}), then multiplying by this term would be equivalent to multiplying by 1, and therefore would not affect things. However, until that point, it returns a bit of the original response. This function is not ideal, because of the necessity of using ρ_{avg} instead of ρ. In fact, it made the iterative method converge at a faster rate, but had little difference on the final result. Because of this, it was basically identical to Method 2E (but more complicated), and so was discarded.

Method	Calculated α	Minimum Distance	Maximum Distance	Center Distance
2D	1.05882e-7	1.67	2.45	2.35
2B	1.98804e-7	2.02	2.56	2.36
1B	2.15810e-7	2.09	2.63	2.36
1A	2.21703e-7	2.09	2.62	2.36
2A	2.29665e-7	2.12	2.66	2.36
2E	2.34569e-7	2.09	2.62	2.36
1D	2.44648e-7	2.20	2.74	2.37
1C	2.62329e-7	2.27	2.81	2.37

Table B.1: Comparing Different Radial Distortion Methods
Calculating α using different radial distortion methods on the same image pair of approximately flat terrain, then using converting to distances. Typically, the minimum was at the bottom left, while the maximum was at the top right, which suggests that the camera was not perpendicular to the surface when taking these base pictures. With this image pair, method 2C gave an unstable negative guess for α, and so was discarded.

B.3 Radial Distortion Schemes – Comparisons

A representative image pair of a flat surface was taken and analyzed using all of the different methods. Table B.1 shows the resultant α values. In addition, the table shows the resulting minimum, maximum, and center distances observed after processing the flat image pair, using those α values to calculate for the optical flow.

A second pair of images of a sculpted terrain was also grabbed, to verify that the methods all worked on both smooth and sculpted surfaces. Because the process of calculating α requires that you know the optical flow at all points in the image, and we decided to achieve this by calculating α for a flat surface, we could not calculate α using this new image pair. We could, however, examine the ranges and make sure that they are consistent. All methods responded as expected to the sculpted terrain, and the radial distortion correction worked about the same as for the flat terrain pictures listed in Table B.1.

When comparing the magnitude of the calculated α, method 2D is far below all others, with method 2B being next weakest, methods 1B, 1A, 2A, 2E, and 1D calculated medium strength α values, and method 1C calculated the strongest α value. Visibly, any of the methods other than method 2D seem reasonable. Methods 2B, 1B, and 2E look slightly better than the others. (Note that even though method 2E estimates an α 10 percent larger than method 1B, the resultant range estimates are essentially identical, this difference in α being counteracted by the difference between the approximations used.) These also seem to be the preferred methods theoretically, since methods 1C, 1D, and 2F seem excessively messy, and methods 1A, 1C, 1D, 2A, 2C, and 2D all seem rather strained, since you have to make some

rather weird approximations. 2B is reasonable and not too messy, but involves an arbitrary constant G which needs to be set. That leaves methods 1B, 2B, and 2E as both acceptable in theory and practice. Of these, method 2B involves the arbitrary constant, which visibly doesn't seem to improve the situation much. In addition, picking values of this constant between -8 and 4 seem to produce less than a 10 percent change in the resultant α calculated (and examination of the terms which G is used to approximate shows that G must be between -3 and -5), so using method 2B seems to be extraneous. Furthermore, since method 1B is almost identical mathematically to method 2B with the constant G removed, method 2B can be discarded.

Of the remaining two methods, method 1B and method 2E, both seemed to produce almost identical results on all of the image pairs selected. Since method 1B involved slightly simpler math, it was selected as the method of choice, even though method 2 is mathematically more correct for barrel distortion (as shown in Figure 4-2).

The complete derivation for method 1B, as well as explanations on why the radial distortion method is needed and how to use it, can be found in Chapter 4.

Appendix C

Specific Terrain-Mapping System Choices

Chapters 2, 3, and 4 have outlined our choice of a theoretical imaging system, and have attempted to supply reasons for this choice. However, there were many implementation choices which needed to be made in the process of completing this thesis which have not yet been addressed. Some of these choices, such as our camera position above the ground, the amount to downsample and frame-average images, and the amount of camera shift between images, were referred to in Chapter 5. Others, such as the decision to do all imaging from directly over the terrain looking straight down and the determination of the principal distance of the camera, have not been referenced in the body of this thesis. In fact, there are many choices, such as the decision not to do more complicated image filtering (such as median or outlying-point filtering) to reduce the camera noise and the decision not to try to correct the brightness-falloff at the image corners, which were only made because empirical tests seemed to suggest that they were not needed. This appendix details some of the more difficult or important decisions and calculations, along with the justifications behind each of them.

C.1 Camera Orientation Decision

At the very beginning of the project, I decided to point the camera straight down towards the ground and leave it pointed in that direction throughout the imaging process. One reason for this decision was that the basic optical flow calculations are much simpler under pure translational motion, suggesting that changing the angle of view would only complicate the problem. Another consideration was that the final transformation from a distance map in the image coordinate system into a terrain reconstruction in the world coordinate system is typically a rather complex transformation, but is actually trivial when looking straight down from above. Not only does this transformation become trivial when looking from above, but errors in distance

calculations are confined to affect only a single terrain reconstruction value, whereas distance errors in the transformation while the camera is not looking straight down could cause errors in calculating a large number of the terrain reconstruction values. Another related issue is that of occluding objects, where an object near to the camera can block our view of objects behind it. This is an issue from any angle, but when looking from above, this effect would prevent us from properly recognizing any overhangs or caves in the terrain surface, neither of which is expected or desired, and in fact neither of which we can store in our two-dimensional terrain-reconstruction array, where each (x, y) location has a single height associated with it.

Given the above issues, it seemed simpler to take all pictures from above, looking down. This decision actually simplified several other aspects of the imaging system when coupled with our decision to solve the optical flow problem with a least-squared solution, as described in Chapter 2. In our least-squared solution, we made the assumption that each region corresponds to an area of constant distance. Because we are imaging from above, these regions of constant distance also correspond to regions of constant height, and so we were able to pick our 1" terrain reconstruction squares as our image regions. As an added benefit, these assumptions guarantee that these terrain reconstruction squares are square regions in the image. By choosing the camera heading angle to be a multiple of 90 degrees, the edges of these square regions line up with the axes of the image, which makes it possible to specify the regions in the imaging algorithm through only two corner points instead of by describing the entire periphery. If we were not looking straight down from a heading angle which is a multiple of 90 degrees, we would not easily be able to match 1" terrain squares to regions in the image, and specifying the extents of these regions would be much more difficult.

C.2 Camera Focal Length Calculation

In order to calculate distance from optical flow, Section 2.1.2 used a method of similar triangles, introducing a variable f which represented the distance between the camera lens and the image plane, or principal distance. As used in the actual code, this variable not only signifies the principal distance, but it also includes the scaling between pixels and inches. Although these values are available in the camera specifications, they can also be calculated empirically. By calculating them empirically, we avoid having to worry about whether the actual apparatus deviates from the documented specifications (through manufacturing errors or some other issue), but the theoretical value has also been calculated as a check.

In order to calculate the distance f, we placed a ruler on the ground and imaged it from several distances. Then, by counting the number of pixels in one inch of the ruler from the different heights, we can do a least-squares minimization to determine the best value for f. While doing this minimization, we can also calculate the best value for the height of the robot gantry origin above the terrain floor, Z_{origin}. The

Z value (inches)	Horizontal Pixels	Vertical Pixels
15	147	149
10	93	95
5	69	70
0	54	54

Table C.1: Pixels per Inch of Terrain vs. Camera Z Value
These values were obtained by taking images of a ruler from the specified Z values. The horizontal pixels counts were obtained while the ruler was horizontal, using the $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ edge-detection filter and counting pixels between inch marks in the x direction. The vertical pixels counts were obtained while the ruler was vertical, using the $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ edge-detection filter and counting pixels between inch marks in the y direction.

gantry origin is Z_{origin} inches above the floor, with the Z axis of the gantry pointing down towards the floor, as described in Section 1.3.1. The distance between the camera and the floor (needed for the transformation between distance map and terrain reconstruction), is therefore $(Z_{origin} - Z)$.

Two different sets of images were acquired in order to calculate the value of f in the x and y directions individually, to verify that the image pixels are actually square. In each image set, an appropriate edge-detection filter was used to bring out tick marks on the ruler and to suppress all other image information. The results are recorded in Table C.1.

By using the idea of similar triangles, we see that

$$\frac{f}{\Delta_{pixels}} = \frac{Z_{origin} - Z}{1\ in}$$

From this, we can solve the least-squared minimization for

$$\sum((Z_{origin} - Z)\Delta_{pixels} - f)^2$$

simultaneous for f and Z_{origin} using all of the data points in Table C.1, and get that

$$f = 1287.72\ pixels$$

and

$$Z_{origin} = 23.69\ in$$

The theoretical value can be calculated using the facts that the lens has a focal length of 12 mm, that the CCD array width is 0.64512 cm, and that there are 640 pixels across the CCD array. This tells us that the pixel size is $\frac{0.64512\ cm}{640\ pixel} =$

0.001008 $cm/pixel$. The focal length of the camera in pixels is therefore

$$\frac{1.2\ cm}{0.001008\ cm/pixel} = 1190.48\ pixels$$

Since the value of f empirically calculated is the principal distance instead of the focal length, and the relation between the two is that the principal distance is $(1 + M)$ times the focal length (where M is the lateral magnification of the lens, which is relatively small for our lens), the theoretical value is reasonably close to the empirical value $f = 1287.72\ pixels$, and so lends credence to the empirical results.

Unfortunately, Table C.1 shows that the pixels aren't exactly square, but are in fact a few percent different in size. I decided that this few percent was small enough to ignore, and so minimized both sets of data together. However, accounting for those different pixel sizes might affect the outcome of the optical flow to distance calculations. Another thing to beware is that the calculation for Z_{origin} neglects to account for the thickness of the ruler itself or that of the burlap cloth. The ground is therefore a little further away than the calculated distance, so I decided to use a value, $Z_{origin} = 24.0\ in$. Another thing to note is that these pixel measurements were counted directly in the image, which is known to suffer from radial distortion. The pixel counts in the table should no doubt be a few pixels higher in all measurements. However, I again decided to ignore this source of error, since it seemed insignificant compared to other errors in the system.

C.3 Camera Height, Downsample, and Frame-Averaging Decisions

The camera height at which to grab our image pairs, along with the amount of downsampling to use, are perhaps the most critical decisions to make. These two choices greatly affect the frequency content of the images, and as shown in Chapter 3, the frequency content of our images can have a large effect on the overall success of the optical flow calculations on that image. Table 3.1 shows the detected frequency content of the burlap texturing from different heights when downsampling by 3. Table C.2 shows the same information without downsampling. Downsampling by 3 has started to remove the burlap texturing frequency, and larger downsampling values almost completely remove it. This is devastating to the ranging estimates, because the burlap texture is the only guaranteed contrast in the image which allows us to successfully calculate ranges. This means that no downsampling, downsampling by 2, and downsampling by 3 are the only real options. No downsampling and downsampling by 3 are explored in detail, with the assumption that downsampling by 2 would fall somewhere between the other two in all aspects.

As discussed in Section 3.1, the highest frequency ideally allowed in the image is $\omega = 1.3389$. Examination of Tables 3.1 and C.2 shows that this means we should

Camera Height	Burlap Frequency	Burlap Pixels
16"	0.902	6.966
18"	1.026	6.121
20"	1.151	5.459
22"	1.244	5.050
24"	1.338	4.698
26"	1.462	4.298
28"	1.555	4.040
30"	1.649	3.811

Table C.2: Burlap Frequency and Period vs. Camera Height Without Downsampling At each of the specified heights, the burlap frequency is the observed frequency (with π as the maximum) of the burlap texture grid without downsampling, and the burlap pixels is the corresponding number of pixels in one period of this grid. The heights from 10" to 14" are not shown in this table because the frequency components were too low to be distinguished from the low-frequency offset and lighting effects in my frequency analysis. These frequencies are all about 1.5 times the frequencies found at the same height when downsampling by 3 (as shown in Table 3.1.) The frequency shown here is the correct burlap frequency, while the one detected when downsampling by 3 is essentially the first harmonic, detected because the lower "fundamental" frequency was filtered out by a combination of focus blur and the non-ideal antialiasing filter.

Camera Distance	Screen Width	Screen Height
10"	4.91"	3.68"
12"	5.89"	4.42"
14"	6.87"	5.15"
16"	7.85"	5.89"
18"	8.83"	6.63"
20"	9.82"	7.36"
22"	10.80"	8.10"
24"	11.78"	8.83"
26"	12.76"	9.57"
28"	13.74"	10.31"
30"	14.72"	11.04"

Table C.3: Screen Size vs. Camera Height
At each of the specified distances, there are the maximum possible number of inches of terrain which can be seen in the 632x474 pixel image taken by our camera. The edges with frame-grabber noise have already been discarded, but parts of the darkened corners are still in the image.

limit ourselves to a height of 16" when downsampling by 3 and a height of 24" when not downsampling. Downsampling by 3 has the benefit that there is less image data to process, but taking pictures from a higher location allows for imaging a larger area at a single time. Table C.3 shows the maximum number of inches of width and height that can be seen in the image at various camera heights.

At this point, it is important to remember that even though we are picking the height from which the camera should take images, we are expecting there to be some variation in the terrain being mapped. If we have 8" of variation in our terrain, and we're imaging from 18" above the terrain floor, we can expect to detect distances from 10" to 18", and all of our analysis actually depends on the distance detected, not the commanded distance above the terrain floor. This fact has several important ramifications. First, the camera will not always be in-focus. Second, we cannot choose an ideal distance, but must instead select an appropriate range of distances, and use the largest distance in that range as the commanded camera height.

If we run at a downsample value of 3, our frequency analysis says we want the camera to be 16" away, and in that case, if we had 6" of variation in the terrain, we must be prepared for the possibility that the whole image is only 10" away. At this distance, we would be able to calculate the terrain reconstruction for areas of at most 5" by 4". In reality, we can only calculate the terrain reconstruction for 4.91" by 3.68", but we need full inch-squares. Usually that would mean rounding down to 4" by 3". However, it is reasonable to attempt a distance estimate of the partially cut-off inch-squares for the large efficiency improvement gained by imaging almost

twice as much terrain per image. It is important to remember that this will only be an issue at the peaks of the terrain, where the distance is 10". Even so, 6" is a very restrictive amount of variation, and ranging the terrain in 5" by 4" squares would require approximately 490 images, which would take about two and a half hours to complete. This is unacceptable.

By stretching our frequency limitation and running at 20", we can allow ourselves 8" of variation, and calculate 6" by 4" of terrain each image, requiring 414 images, which takes just over 2 hours. Alternately, we could allow ourselves 6" of variation and calculate 7" by 5" of terrain each image, requiring 280 images, which is only an hour and a half. By running at 20" instead of 16", we increase the noisiness of some of the calculations. However this error will only affect measurements of terrain lower than 4" (meaning that it is between 16" and 20" away from the camera) – it will not affect the whole terrain reconstruction.

If we don't downsample but we do pay attention to our frequency limits, we can run at 24". At this height, we could allow ourselves 10" of variation in 7" by 5" images or 8" of variation in 8" by 6" images, requiring only 207 images, which takes just over an hour to run.

Obviously, there are many possible configurations which could be chosen using Tables 3.1, C.2, and C.3 as a guide, and remembering that the full terrain is 70" by 140", and that the program needs approximately 17 seconds to grab and process each image pair and move on to the next location. I chose (somewhat arbitrarily) to run at a height of 24" without downsampling, imaging 8" by 6" each step. This allows for an acceptable terrain variation of 8".

Because we are not downsampling, we do not get any noise-reduction benefit from downsampling. Because of this, we can only get noise reduction by filtering and frame-averaging. As discovered in Chapter 3, we want to filter out the high frequency components of our image anyway. So, I chose to filter the image with a filter of cutoff frequency $\omega = 1.3389$. The least-squared nature of our optical flow method, along with this lowpass filter, will remove much of the stochastic CCD camera noise. As a result, I decided only to frame-average by 2 (it adds approximately 1 second to the 17 second grab and process time for every extra frame averaged in).

C.4 Camera Shift Decision

The amount of camera shift to use between images can easily be chosen after deciding on the commanded camera height and the desired optical flow shift in the image pair. However, picking the desired optical flow shift is rather complex. Chapter 3 explores the issue of the frequency content of images and optical flow shift. In order to keep the time derivatives from being too noisy, we want to keep the optical flow pixel shift smaller than 1. However, the chapter also shows that the distance estimate will be more precise when calculated from larger pixel shifts. We must remember, however, that under no condition can we ever let the pixel shift be larger than half

of the burlap texture frequency. If we were to let that value be exceeded, the time derivative calculation will become aliased, and could cause major problems. We can see that this is possible if we realize that, since each little thread of burlap is typically indistinguishable from the adjacent one, a shift in one direction of more than half of the burlap texture frequency would look just like a smaller shift in the other direction. The optical flow algorithm is more likely to find the smaller pixel shifts, and so will miscalculate the pixel shift, and therefore miscalculate the distance. From this, we see that we have a soft maximum of one pixel worth of shift, and a hard maximum of half of the burlap texture frequency (which is dependent on the height). However, when using the shift-biasing methods developed in Section 3.4.1, these limits are measured relative to the initial bias used between the two images.

Because of the direct relationship between camera shift and pixel shift, we can use these limits on the desired pixel shifts to help us decide on what camera shift to use. Table C.4 shows the pixel shifts which would result from using the specified camera shifts at the specified heights. Because of the use of a single image pair to determine all heights, the important thing to consider is the difference between pixel shifts as a result of the different camera heights using a given camera shift. Again, it is important to remember that although the commanded camera height is fixed for an image pair, the observed camera height can be anywhere between zero and the commanded camera height, depending on the topography of the terrain in the image. As an example of how to consider the table, let us assume that we were going to command the camera to run at 24" while taking undownsampled images, and expecting to see 10" of variation in the image. This means that we are expecting to notice camera heights of 14" to 24". For now, we will ignore the effect of shift-biasing. We may choose to consider 20" to be a central height where we would want the pixel shift to be close to 1 pixel. From this, we can use the table to choose a camera shift of 0.016", which would give us a pixel shift of 1.429 pixels at 14" and 0.833 pixels at 24". If we wished to push the pixel shift up to its hard limit, we can look in Table C.2 and notice that at 20", the burlap texture has a period of 5.459 pixels. This means that we can allow a pixel shift of 2.73 pixels, so we can multiply the camera shift by 2.73, thereby using a camera shift of 0.043". Because both the pixel shift and the burlap texture period are directly proportional to the height, satisfying the half-period requirement at 20" means that it will automatically be satisfied at the other frequencies as well, where we can expect to see pixel shifts of $1.429 \times 2.73 = 3.90$ pixels and $0.833 \times 2.73 = 2.27$ pixels.

These pixel shifts, between 2.27 pixels and 3.90 pixels, are way above our soft limit of one pixel, which means that they would produce rather noisy derivative calculations, and therefore would result in rather noisy distance estimates. If it weren't bad enough that we had noisy derivative calculations, the camera shift of 0.043" used to produce these derivatives is still rather small, meaning that if we could increase it, we should.

Fortunately, shift-biasing allows us to increase the camera shift and escape our

Camera Height	Camera Shift (inches)										
	0.008	0.009	0.011	0.012	0.014	0.016	0.017	0.019	0.020	0.022	0.023
10"	1.000	1.200	1.400	1.600	1.800	2.000	2.200	2.400	2.600	2.800	3.000
12"	0.833	1.000	1.167	1.333	1.500	1.667	1.833	2.000	2.167	2.333	2.500
14"	0.714	0.857	1.000	1.143	1.286	1.429	1.571	1.714	1.857	2.000	2.143
16"	0.625	0.750	0.875	1.000	1.125	1.250	1.375	1.500	1.625	1.750	1.875
18"	0.556	0.667	0.778	0.889	1.000	1.111	1.222	1.333	1.444	1.556	1.667
20"	0.500	0.600	0.700	0.800	0.900	1.000	1.100	1.200	1.300	1.400	1.500
22"	0.455	0.545	0.636	0.727	0.818	0.909	1.000	1.091	1.182	1.273	1.364
24"	0.417	0.500	0.583	0.667	0.750	0.833	0.917	1.000	1.083	1.167	1.250
26"	0.385	0.462	0.538	0.615	0.692	0.769	0.846	0.923	1.000	1.077	1.154
28"	0.357	0.429	0.500	0.571	0.643	0.714	0.786	0.857	0.929	1.000	1.071
30"	0.333	0.400	0.467	0.533	0.600	0.667	0.733	0.800	0.867	0.933	1.000

Table C.4: Pixel Shift vs. Camera Height and Camera Shift
The above table shows the pixel shifts (in pixels) which would occur in an image pair taken at the given height with the given camera shift without downsampling. Any camera shifts are allowed, but the listed camera shifts were chosen to produce a pixel shift of 1 pixel for one of the camera heights.

soft limit on pixel shifts. If we add shift-biasing to the previous example, we can shift bias the image pair by 3 pixels (halfway between 2.27 and 3.90, but remember that the shift bias must be an integer), so that objects at a camera height of 24" produce a pixel shift of $2.27 - 3 = -0.73$ pixels and objects at 14" produce a pixel shift of $3.90 - 3 = 0.90$ pixels. It is these pixel shifts, not the original 2.27 and 3.90, which we should compare against our soft and hard limits. From this we see that, using a camera shift of 0.043" and using the shift-biasing method, we are not only well below the hard limits of 2.27 and 3.90 pixels, but we are also below the soft limit of one pixel, which means that our time derivatives are actually rather clean. At this point, it is important to note that the pixel shift of 3 pixels actually corresponds to a height slightly smaller than 18", which is noticeably closer to 14" than to 24". The shift-biasing height will not always correspond to a point exactly half-way between the near and far extremes being imaged.

After all of the analysis in Chapter 3 showing the importance of frequency-limiting the image, along with the importance of restricting ourselves to pixel shifts of one pixel or less, it seems foolish to even consider violating this limitation and allowing higher pixel shifts. However, by iterating the region determination, derivative calculation, and range estimation steps of the terrain-mapping scheme as mentioned in Chapter 5, we can avoid the issue. As long as the range estimation returns a reasonable estimate, the next iteration will use a better bias when shift-biasing, and will therefore get a better estimate. Within very few iterations, the shift-biasing is within one pixel of the actual answer, and so the issue of noisy derivatives can be avoided. This is a

quick way to guarantees that our final optical flow calculation is a sub-pixel shift, which is often desirable [DF95]. Under no circumstances can we ignore the hard frequency limit of half of the burlap texture frequency, however, since ignoring that limit would result in an unreasonable original range estimate, leaving no guarantee that the iterative method will converge to the correct answer.

With all of this in mind, we can now revisit Table C.4 and choose the largest possible camera shift which does account for shift biasing. For comparison, we will again assume that we were going to command the camera to run at 24" while taking undownsampled images, and expecting to see 10" of variation in the image, just as with the previous example. Using Table C.2, or remembering what we saw in the previous example, we know that the largest allowable pixel shift at 14" is 3.90 pixels, and at 24" the largest allowable is 2.27 pixels. What that means to us now is that we want to choose a camera shift which will cause a pixel shift of at most $(\Delta_{bias} + 3.90)$ at 14" and $(\Delta_{bias} - 2.27)$ at 24", where Δ_{bias} is the expected pixel shift used to bias the initial derivative calculation. This means that we're looking for a camera shift which gives a difference of $3.90 + 2.27 = 6.17$ pixels between the pixels shifts at 14" and 24", with an integer pixel shift (to use for biasing) falling about two-thirds of the way between the 14" and the 24" pixel shift values.

Considering the 0.016" camera shift of the previous example, the difference between the two pixel shifts is $1.429 - 0.833 = 0.596$, which means we can use a shift around $\frac{6.17}{0.596} = 10.35$ times larger. Unfortunately, this would yield pixel shifts of $10.35 \times 1.429 = 14.79$ pixels and $10.35 \times 0.833 = 8.62$ pixels, and there are no integers within 2.27 pixels of the lower bound and within 3.90 of the upper bound. In fact, the absolute largest allowable camera shift in this case is 0.155", which causes objects at a distance of 14" to shift by exactly 13.90 while causing objects at 24" to shift by 8.10 pixels. In order for this to work, it is imperative that the image pair is shift-biased by 10 pixels, which corresponds to a height of just under 20". For safety, I decided to use a slightly smaller camera shift, namely 0.125"

References

[AS64] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions.* National Bureau of Standards Applied Mathematics series. U.S. Government Printing Office, Washington, D.C., 1964.

[DF95] C. Quentin Davis and Deenis M. Freeman. Statistics of sub-pixel motion estimates based on optical flow. Submitted to IEEE Transactions on Pattern Analysis and Machine Vision, March 1995.

[Fau93] Olivier Faugeras. *Three-Dimensional Computer Vision – A Geometric Viewpoint.* Series in Artificial Intelligence. MIT Press, Cambridge, MA, 1993 edition, 1993.

[Hor86] Berthold Klaus Paul Horn. *Robot Vision.* The MIT Press, Cambridge, MA, 1986.

[Kal94] Timothy Elmer Kalvaitis. Distributed shared memory for real time hardware in the loop simulation. Bachelor's/master's joint thesis, MIT, Department of Electrical Engineering, May 1994.

[KK65] A. Kraskna-Krausz, editor. *The Focal Encyclopedia of Photography.* Focal Press, London, 1965 edition, 1965.

[Lim90] Jae S. Lim. *Two-Dimensional Signal and Image Processing.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990.

[Shi86] Yoshiaki Shirai. *Three-Dimensional Computer Vision.* Series in Symbolic Computation. Springer-Verlag, New York, 1986.

[Ski91] Kurt D. Skifstad. *High-Speed Range Estimation Based on Intensity Gradient Analysis.* Series in Perception Engineering. Springer-Verlag, New York, 1991.

[Ste93] Gideon P. Stein. Internal camera calibration using rotation and geometric shapes. Master's thesis, MIT, Department of Electrical Engineering, 1993.

[Wol74] Paul R. Wolf. *Elements of Photogrammetry.* McGraw-Hill Inc., New York, 1974.