Range Estimation Accuracy Using
Depth-From-Focus Methods — Theory and Experiment
by
Paul Werner Fieguth
B.Sc., University of Waterloo (1991)
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 1992
© Massachusetts Institute of Technology 1992. All Rights Reserved.
This thesis may be reproduced and distributed for LFM partners’ internal use only.
No other person or organization may reproduce this thesis without permission of the copyright owner. Requests for permission can be addressed to the LFM Manager, Communications.

Author .................................................................
Department of Electrical Engineering and Computer Science
Aug 14, 1992
Certified by ..............................................................
David H. Staelin
Professor of Electrical Engineering
Thesis Supervisor
Accepted by ..............................................................
Campbell L. Searle
Chairman, Departmental Committee on Graduate Students
Range Estimation Accuracy Using Depth-From-Focus

Methods — Theory and Experiment

by

Paul Werner Fieguth

Submitted to the Department of Electrical Engineering and Computer Science
on Aug 14, 1992, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Abstract

The surface profile of three dimensional objects can be determined using a depth-
from-focus method. An object is viewed in patterned incandescent visible light at a
variety of standoff distances, and its three dimensional shape is inferred from focus
information. The objective of this work is to minimize the depth estimation error, and
to uncover the factors which determine this minimum limit. A principle advantage
of this depth-from-focus method is the large number of pixel depths which can be
processed in parallel – over 600 points were observed.

The system optics are analyzed in some detail. The manner in which the object
blurs as it is moved from focus is strikingly asymmetrical. This asymmetry cannot
be explained using thin lens classical or diffraction optics, although classical optics
applied to a thick lens (using ray tracing) does so successfully.

A set of results using depth-from-focus estimation is provided for a variety of
materials. Using an ideal test object, a mirror, r.m.s accuracies of 0.05 micron were
achieved; algorithms requiring less memory or computing power routinely achieved
accuracies around 0.1 micron. The other materials tested – chalk, black plastic, and
copper – all achieved accuracies of between 1 and 2 microns.

The effect of scaling the apparatus dimensions is analyzed, taking both classical
and diffraction effects into account. The apparatus used in this work has a field of
view of 0.5 mm; using the derived scaling laws, the results of this thesis may be applied
to depth-from-focus systems having very different dimensions.

A variety of calibration issues are examined. Calibration algorithms (and some
computer listings) are provided for calibrating the projected pattern, the motion of
the stage holding the test object, the system optics, and the imaging camera.

Thesis Supervisor: David H. Staelin
Title: Professor of Electrical Engineering
Range Estimation Accuracy Using Depth-From-Focus

Methods — Theory and Experiment

by

Paul Werner Fieguth

Submitted to the Department of Electrical Engineering and Computer Science
on Aug 14, 1992, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Abstract

The surface profile of three dimensional objects can be determined using a depth-from-focus method. An object is viewed in patterned incandescent visible light at a variety of standoff distances, and its three dimensional shape is inferred from focus information. The objective of this work is to minimize the depth estimation error, and to uncover the factors which determine this minimum limit. A principle advantage of this depth-from-focus method is the large number of pixel depths which can be processed in parallel — over 600 points were observed.

The system optics are analyzed in some detail. The manner in which the object blurs as it is moved from focus is strikingly asymmetrical. This asymmetry cannot be explained using thin lens classical or diffraction optics, although classical optics applied to a thick lens (using ray tracing) does so successfully.

A set of results using depth-from-focus estimation is provided for a variety of materials. Using an ideal test object, a mirror, rms accuracies of 0.05 micron were achieved; algorithms requiring less memory or computing power routinely achieved accuracies around 0.1 micron. The other materials tested — chalk, black plastic, and copper — all achieved accuracies of between 1 and 2 microns.

The effect of scaling the apparatus dimensions is analyzed, taking both classical and diffraction effects into account. The apparatus used in this work has a field of view of 0.5mm; using the derived scaling laws, the results of this thesis may be applied to depth-from-focus systems having very different dimensions.

A variety of calibration issues are examined. Calibration algorithms (and some computer listings) are provided for calibrating the projected pattern, the motion of the stage holding the test object, the system optics, and the imaging camera.

Thesis Supervisor: David H. Staedlin
Title: Professor of Electrical Engineering
Acknowledgements

First I would like to thank Professor Staelin for his guidance and support over this past year. His contributions extend well beyond the confines of this thesis - he has taught me some valuable elements of doing research. In particular, over the course of the year he made a number of suggestions on issues which I preferred to ignore, finding them too messy or awkward to be of consideration; some of these issues have since found an important place in this thesis.

Next I would like to thank my friends Mike Schwartz, Bruce Normand, Kevin Christian, John Keen, and Alain Curodeau; they may not have added much insight to the thesis, but their presence has made my time at MIT much more enjoyable.

More distant friends (geographically speaking), David Willms and Andrew Isaak in particular, contributed well over two hundred email messages to my account this past year - thanks for keeping in touch!

Finally my parents and my sister deserve special thanks. Their support for me in my various endeavors has been unconditional and is much appreciated. I apologize ahead of time for producing a thesis that you may find somewhat less than exciting.

The author wishes to thank the following for their support of the research leading to this thesis:

- Leaders for Manufacturing for providing research funds

- Natural Sciences and Engineering Research Council (NSERC) of Canada for providing a NSERC 1967 Fellowship
Dedicated To My Parents
And To My Sister Anita

Hope You Like It!
Contents

List of Figures 8

List of Tables 10

1 Introduction 11
   1.1 Research Motivation ............................................. 11
   1.2 Overview of Depth Sensing Methods .............................. 14
   1.3 Thesis Overview .................................................. 17

2 System Description 19
   2.1 System Overview .................................................. 19
   2.2 Optics Overview .................................................. 22
   2.3 Hardware Overview ............................................... 24
   2.4 Software Overview ............................................... 26

3 System Optics and Algorithm Theory 27
   3.1 Modes of Operation .............................................. 28
   3.2 Optical Theory ................................................... 28
      3.2.1 Perfectly Specular Object ................................. 30
      3.2.2 Diffuse Object Without Detail ............................ 42
      3.2.3 Object Having Surface Detail ......................... 44
   3.3 Optical System Scaling ......................................... 45
      3.3.1 Constant Projected Pattern .............................. 46
      3.3.2 Object Having Surface Detail ............................ 48
3.3.3 Scaling with Diffraction Effects .......................... 48
3.3.4 Practical Issues ............................................ 50
3.4 Depth Estimation Algorithms ................................. 51
  3.4.1 Blur Measures ........................................... 52
  3.4.2 Estimators ............................................... 55
3.5 System Performance .......................................... 59
  3.5.1 Overview of System Noise Sources ..................... 59
  3.5.2 Effects of Camera White Noise ......................... 61
  3.5.3 Effects of Sampling Size ............................... 64

4 System Calibration .............................................. 70
  4.1 Projected Pattern .......................................... 70
    4.1.1 Pattern Spot Location ................................ 71
    4.1.2 Pattern Spot Tracking .................................. 73
    4.1.3 Pattern Preparation ................................. 75
    4.1.4 Pattern Positioning ................................. 75
  4.2 Stage Motion ............................................... 76
    4.2.1 Motion Jitter ......................................... 76
    4.2.2 Periodic Aberration ................................ 78
    4.2.3 Micrometer Backlash .................................. 81
  4.3 System Optics ................................................ 82
    4.3.1 Screen Deformation ................................ 82
    4.3.2 Curvature of Focus ................................ 85
    4.3.3 Light Source ........................................ 86
  4.4 Camera Frame Grabbing ..................................... 87
    4.4.1 Interference Signals ................................ 87
    4.4.2 Camera CCD Noise ................................ 89

5 Experimental Results .......................................... 91
  5.1 Ideal Specular Surface - Blur Measure Noise .......... 91
    5.1.1 FFT Measure ........................................ 91
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.2 Difference-Squared Measure</td>
<td>93</td>
</tr>
<tr>
<td>5.1.3 Calibration Curve Noise</td>
<td>94</td>
</tr>
<tr>
<td>5.2 Ideal Specular Surface – Detailed Noise Breakdown</td>
<td>95</td>
</tr>
<tr>
<td>5.2.1 Optical Offsets</td>
<td>97</td>
</tr>
<tr>
<td>5.2.2 Camera Noise</td>
<td>99</td>
</tr>
<tr>
<td>5.2.3 Pattern Spot Differences</td>
<td>100</td>
</tr>
<tr>
<td>5.2.4 Test Object Flatness</td>
<td>104</td>
</tr>
<tr>
<td>5.2.5 Stage Motions</td>
<td>105</td>
</tr>
<tr>
<td>5.3 Ideal Specular Surface – Further Considerations</td>
<td>106</td>
</tr>
<tr>
<td>5.3.1 Sampling Size</td>
<td>106</td>
</tr>
<tr>
<td>5.3.2 Spot Pattern Improvement</td>
<td>108</td>
</tr>
<tr>
<td>5.3.3 Depth Postprocessing</td>
<td>110</td>
</tr>
<tr>
<td>5.3.4 System Scaling Examples</td>
<td>111</td>
</tr>
<tr>
<td>5.4 Other Surfaces – With Pattern</td>
<td>111</td>
</tr>
<tr>
<td>5.4.1 Chalk</td>
<td>112</td>
</tr>
<tr>
<td>5.4.2 Plastic</td>
<td>115</td>
</tr>
<tr>
<td>5.4.3 Copper</td>
<td>116</td>
</tr>
<tr>
<td>5.5 Other Surfaces – Without Pattern</td>
<td>118</td>
</tr>
<tr>
<td>5.6 Practical Experiments</td>
<td>119</td>
</tr>
<tr>
<td>5.7 Conclusions</td>
<td>125</td>
</tr>
<tr>
<td>6 Conclusions</td>
<td>126</td>
</tr>
<tr>
<td>6.1 Thesis Achievements</td>
<td>126</td>
</tr>
<tr>
<td>6.2 Future Research Possibilities</td>
<td>127</td>
</tr>
<tr>
<td>A Calibration Code Listings</td>
<td>130</td>
</tr>
<tr>
<td>B Theory &amp; Simulation Code Listings</td>
<td>162</td>
</tr>
<tr>
<td>Bibliography</td>
<td>171</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Open Loop Computer Aided Manufacturing</td>
<td>12</td>
</tr>
<tr>
<td>1-2</td>
<td>Closed Loop Computer Aided Manufacturing</td>
<td>12</td>
</tr>
<tr>
<td>1-3</td>
<td>Operating Algorithm for Closed Loop Manufacturing</td>
<td>13</td>
</tr>
<tr>
<td>2-1</td>
<td>Overview of the Depth-From-Focus Estimation System</td>
<td>20</td>
</tr>
<tr>
<td>2-2</td>
<td>Overview of the Optical System</td>
<td>23</td>
</tr>
<tr>
<td>3-1</td>
<td>Optical System for Specular Object</td>
<td>30</td>
</tr>
<tr>
<td>3-2</td>
<td>Theoretical Blur Radius $d_b/2$ vs. Object Position</td>
<td>32</td>
</tr>
<tr>
<td>3-3</td>
<td>Theoretical Image Distance $(d_i - d_{fp})$ vs. Object Position</td>
<td>33</td>
</tr>
<tr>
<td>3-4</td>
<td>Comparison of Simplistic and Experimental Blur Curves</td>
<td>34</td>
</tr>
<tr>
<td>3-5</td>
<td>Effect of CCD Image Sampling</td>
<td>35</td>
</tr>
<tr>
<td>3-6</td>
<td>Basic Lens Diffraction Effects</td>
<td>36</td>
</tr>
<tr>
<td>3-7</td>
<td>Simple Laboratory Lens Focus Test</td>
<td>38</td>
</tr>
<tr>
<td>3-8</td>
<td>Ray Tracing Program Variables</td>
<td>39</td>
</tr>
<tr>
<td>3-9</td>
<td>Ray Tracing Program Results</td>
<td>40</td>
</tr>
<tr>
<td>3-10</td>
<td>Subset of Image Patterns from Ray Tracing</td>
<td>41</td>
</tr>
<tr>
<td>3-11</td>
<td>Optical System for Diffuse Object</td>
<td>42</td>
</tr>
<tr>
<td>3-12</td>
<td>Simple Demonstration of Light Point Spread</td>
<td>43</td>
</tr>
<tr>
<td>3-13</td>
<td>Required Variables for Scaling Considerations</td>
<td>46</td>
</tr>
<tr>
<td>3-14</td>
<td>Outline of Depth Estimation Process</td>
<td>51</td>
</tr>
<tr>
<td>3-15</td>
<td>Idealized Blur Curve for Simulation Program</td>
<td>65</td>
</tr>
<tr>
<td>3-16</td>
<td>Simulation Results for Sampling Test</td>
<td>68</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Discrete Simulation Results ........................................... 65
4.1 Micrometer Backlash Measurements .............................. 81
4.2 Curvature of Focus Regression Parameters (units of \(\mu m/pixel^2\)) ... 86
4.3 CCD Camera Noise Calibration Results .......................... 90
5.1 FFT Blur Measure Noise Comparison ............................... 92
5.2 Difference-Squared Blur Measure Noise Comparison .......... 94
5.3 List of Experiments and Applicable Noise Sources ............. 95
5.4 Test of CCD Pixel Noise Effects ..................................... 100
5.5 Error due to Differences in Pattern Spots ....................... 101
5.6 Experimental Results Over Time (RMS Roughness in \(\mu m\)) .... 106
5.7 Experimental Depth RMS Error (\(\mu m\)) vs. Sampling Step \(\Delta\) .... 107
5.8 Simulated Depth RMS Error (\(\mu m\)) vs. Sampling Step \(\Delta\) .... 107
5.9 Depth RMS Errors (in \(\mu m\)) for Improved Pattern .......... 110
5.10 Accuracy Examples for Scaled Systems ......................... 111
5.11 Experimental Results using Chalk as Test Object .............. 114
5.12 Penny Tests with Pattern, Errors in \(\mu m\) ...................... 118
5.13 Test Results on Copper, No Pattern (Errors in \(\mu m\)) ........ 119
Chapter 1

Introduction

This chapter outlines the motivation driving the research, discusses the general application of the research, and provides an outline for this thesis.

1.1 Research Motivation

The competitiveness of the manufacturing industry has increased dramatically in the last few decades. A manufacturer wishing to increase or just maintain a market share must be capable of rapid production of high quality goods. A simplistic view of production divides it into two stages:

- The manufacture or forming process - converting a design and raw materials into some good.

- Quality control - assessing a good’s adherence to tolerance specifications.

The advent of the digital computer brings with it the possibility of Computer Aided Manufacturing (CAM). Figure 1-1 shows a typical “open loop” CAM station. The computer has direct control over a machine (for example a lathe, milling machine etc.) which produces some item. Under the assumption that all equipment is properly maintained and precisely calibrated, the manufactured goods may be assumed to be of some consistently high quality. There is, however, no provision for the computer
Figure 1-1: Open Loop Computer Aided Manufacturing

Figure 1-2: Closed Loop Computer Aided Manufacturing
to observe the manufactured item and compare it with the intended specifications (hence “open loop”).

The ability to observe the manufactured item is made possible in a “closed loop” system (see Figure 1-2). A possible manufacturing algorithm is shown in Figure 1-3: the item is rapidly scanned by some three dimensional shape acquisition system, the actual dimensions are computed, and these dimensions then finally compared to the initial design goals. Such a system enables a variety of benefits to be realized:

- In general, a closed loop system having an inferior feedforward path (here the forming process) can match the performance of an open loop system with a superior feedforward component[38]. This implies the ability to manufacture high quality items with less accurate, and hence lower cost, forming machines than before.

- Automated quality control can yield a lower item failure rate. Furthermore, the drudgery of human inspection of parts may no longer be required.

- By producing fewer inferior items, and by possibly correcting faults in items that fail to meet specifications, the amount of waste which is created may be reduced.
1.2 Overview of Depth Sensing Methods

The following discussion of depth sensing methods\(^1\) is intentionally brief, and should serve only to familiarize the reader with the variety of methods available. These methods are described in greater detail in [1], and in various publications (for example [27]). Readers interested in further details may wish to consult one of [19, 28, 29, 30].

The general methods available for depth sensing may be grouped into two broad categories: contact and noncontact measurement.

Contact Measurement

The coordinate measuring machine (CMM) has been the standard laboratory and industrial tool of choice for some time [27]. The machine is capable of accuracies to a few microns; however it is capable of measuring only one point at a time (so mapping an entire object takes a long time). Furthermore, it requires that a stylus make contact with the object under study; this restricts the class of objects to be analyzed to those than can tolerate the required stylus pressure, and furthermore that will not damage the stylus. Thin films, very soft materials, and high temperature objects fail these criteria. The surface of the object must also be reachable by the stylus at all points; deep inward rifts or spikes pointing away from the object may prohibit stylus contact with a significant portion of the surface.

Noncontact Measurement

A variety of noncontact depth sensing methods are actively being investigated – triangulation, Moiré or Optical Interference, and Focus Sensing are among the more prevalent methods.

\(^1\)The use of the word “depth” in “depth sensing” may be ambiguous in certain applications. Strictly speaking we are discussing surface profiling; the determination of the unknown surface shape of an object. The use of the word “depth” is used for conciseness and convenience; if one imagines the object under study as mounted on a horizontal table and viewed from above, then depth takes on the correct intuitive meaning.
Triangulation may take two general forms:

1. A single camera is used.
   - A laser illuminates the object from a position different than the camera’s point of observation. A shift in depth on the object is observed as a shift in the observed position of the laser spot by the camera.
   - Some pattern (a grid, lines, spots etc.) is projected onto the object by a light source located away from the camera. All points of interest on the object in question must lie within the focal range of the projection.

In either case, the relative position of the source of illumination relative to the camera must be accurately known.

Alternatively, a single camera views a scene from a variety of vantage points. The position of the camera may be measured independently or inferred from calibration markers within the scene. The images are used to reconstruct the relative positions of markers placed about the field of view (i.e., the points of interest must be set \textit{a priori} and cannot be changed during the experiment). The technique is known as photogrammetry\cite{22, 23}.

2. Two cameras located at different positions are used. The differences between the images from the cameras are due to an offset in depth (this is the same principle used by the human brain). If the object being viewed contains sufficient detail, structured illumination (a pattern) is not required. Although an entire scene can, in principle, be analyzed as a whole (as in the human brain), this is a difficult problem, and the object is frequently scanned via laser instead. In any case, the relative positions of the two cameras must be known accurately; the position of the light source (if any) need not be known.

Accuracies around the micron level have been reported \cite{20}. Triangulation via laser scanning is slow however, and whole scene triangulation remains an active problem.

In Moiré sensing, depth is determined by counting interference fringes\cite{12, 18}. Diffraction gratings are placed both at the source of illumination and at the cam-
era, generating interference fringes spaced at constant depth intervals on the object. Counting the interference fringes between any two points determines the depth offset. The Moiré method permits depth estimation for arbitrary points on the surface with good spatial resolution and high speed. The primary limitation, however, is that the object be sufficiently smooth that each fringe be separately resolvable - there can be no discontinuities in depth. Although lower limits of 10 to 100 microns have been quoted for this method[25], recent results indicate accuracies down to 3 microns, still at high speed and spatial resolution[26].

In focus sensing[13, 14, 15, 16, 17, 21, 24, 31], the approach used in this thesis, the degree to which the object surface is in focus reveals the depth at that point. In the confocal scanning optical microscope (CSOM), illumination and detection beams use a common objective (with a beam splitter). When the point on the object being observed is in focus, the detected spot is bright; when it is out of focus, the spot is spread out (and hence dim). A single observation is insufficient to determine depth - typically the object under test is observed at a variety of standoff distances. The CSOM has the disadvantage that it measures only one spot at a time.

Focus sensing can also be applied to an entire area simultaneously. A camera views the entire area and assesses the degree of focus in each subarea. For this method to function, the object must either have significant surface details that reveal focus, or some pattern must be projected onto it.

The depth estimation method described in this thesis applies focus sensing to a whole area. Past efforts using this method [2, 3] differ from the work of this thesis in two significant ways:

- Past efforts used simple objects that had diffuse, rather than specular, surface characteristics. The object either had a simple pattern (stripes or dots) on it, or such a pattern was projected onto it. In either case, the pattern of interest on the object was nearly ideal and unambiguous.

- Past efforts observed objects on a human scale (ie., inches). The pattern on the object and its distance from the camera can be observed with the human eye.
This thesis deals with objects on a micron scale:

1. The use of classical optics begins to break down; diffraction effects become an issue.

2. Objects no longer act as ideal screens - microstructure and light diffusion effects confuse matters.

3. The system cannot be observed by eye; the experimenter must rely on the camera as the sole source of information.

As a result, the theory (which is quite tractable in [3]) becomes much more involved; even our intuition of the problem is frequently challenged.

In general, an effective three-dimensional scanning system must be capable of both high accuracy and high speed. In the problem at hand, the speed at which an object is observed and processed can always be increased:

- Use faster, custom built, or parallel computer hardware.
- Use faster, high torque, stepper motors to move the object.
- Use high speed CCD cameras operating beyond 60 frames per second.

On the other hand, it is not at all obvious how indefinite improvements in accuracy may be achieved. For this reason, this thesis will concentrate on optimizing accuracy regardless of the incurred processing time. It is expected that practical applications of this work may then opt for reduced accuracy in order to attain realistic speed.

### 1.3 Thesis Overview

This thesis continues the work started by Howard Stuart [4] and John Delisle [1]. It is hoped that this thesis can bring to a conclusion our knowledge of depth-from-focus sensing using the apparatus developed by John Delisle.

\[\text{That is, given unlimited funding.}\]
Chapter 2 gives an overview of the physical apparatus used to perform depth estimation. The chapter discusses the equipment used to maneuver the object, the hardware used to control the positioning motors and illumination, and the computer software that controls the hardware and imaging camera.

Chapter 3 discusses the theory pertinent to this experiment: developing optical theory and modeling the optical system, discussing and justifying algorithms to estimate depth from camera images, and considering the changes in attainable depth accuracy as the system is scaled.

Chapter 4 discusses a variety of calibration issues. Some issues apply generally to depth-from-focus methods; other issues are more particular to our apparatus, however they are included to illustrate the types of problems frequently encountered.

Chapter 5 gives some experimental results, applying the algorithms of Chapter 3 to a variety of materials. The experimental limits to performance are determined, and these are compared to the theoretical expectations of Chapter 3.

Chapter 6 lists the achievements of the thesis and suggests possible further work using our apparatus and for depth-from-focus sensing in general.
Chapter 2

System Description

This chapter briefly describes the optics, hardware, and software of the experiment used in this thesis. The intent of this chapter is to provide a sufficient understanding of the experiment to the extent that the reader will need it for the remainder of the thesis. Details for constructing such a system may be found in [1].

2.1 System Overview

The basic blocks making up the system used in this thesis are sketched in Figure 2-1. The basic operation of the system is described in the following paragraph; the next three sections provide additional details.

The basic principle of operation is that a visible, incoherent light source projects incoherent light through a pattern, which is imaged by a microscope objective onto the object being studied. The object is observed by a CCD camera (plus an attached lens) through a beam splitter and the same microscope objective. The range (or depth) of the object at some point may be inferred from focus information (ie, all points that are simultaneously in focus on the object are at the same range). The object is mounted on a four axis stage capable of movement in the X (horizontal), Y (vertical), Z (range), and rotational directions. The operation of the camera, light source, and positioning stage are all under computer control.

Position sensors are placed on each stage axis to prevent micrometer railing, to
Motors: One stepper motor for each axis (X, Y, Z, and rotation)

Position Sensors:
- Limit Switches: Two switches (+ and - limits) per linear axis (X, Y, and Z)
- Photo Switches: 1. Rotation stage at zero degrees

Figure 2-1: Overview of the Depth-From-Focus Estimation System
permit determination of absolute stage positions, and to prevent the object from hitting the microscope objective.

A display monitor is provided to permit the experimenter to see the camera output without requiring computer intervention.

The basic steps involved in estimating range over a section of an object are as follows:

1. Position the object in the X and Y directions so that the area of interest (specified by the user) lies in the field of view of the camera.

2. Adjust the range (Z axis) to bring the object into approximate focus.

3. Divide the camera field into sections (N by N pixel regions). One range estimate is performed for each section, not one estimate per pixel.

4. Step back far enough in range (-Z direction) so that all parts of the object are known to lie behind the focal plane.

5. Grab one frame of video information from the camera and store in computer memory.

6. For each section (from step 3) apply a blur measure - an operator that takes an N by N pixel image as an argument, and returns a vector or a scalar which is a measure of the degree of focus.

7. Step some amount forward in range (+Z direction).

8. Go to step 5 until all parts of the object are known to lie in front of the focal plane.

9. Apply an estimator to the outputs of the blur measure operator. This yields an estimate of range for each image section.

10. Output the estimates in some user defined format.

11. If the region of the object to be studied exceeds the field of view of the camera, move to a new portion of the object and go to step 4.
2.2 Optics Overview

The optical components of the apparatus are shown in Figure 2-2. Refer to Figure 2-1 to see each component in the context of the whole system. Each item in the figure is described below:

**Light Source:** Two light sources were used:

1. A 100W, 110V AC diffuse bulb. Distance from light to pattern: 30cm (focusing lens not used with this light source).
   Manufacturer: GTE Sylvania


**Focusing Lens:** 5cm diameter convex lens, focal length 30cm. This lens was only used with the 15V light source; the diffuse bulb shone directly onto the pattern.

**Pattern:** An array of spots captured on 35mm slide film. About 29 (horizontal) by 24 (vertical) spots are visible within the camera’s field of view. The spot diameter on the slide is about 40 microns, and the distance between spot centers is about 110 microns.

**Objective:** A 10X standard microscope objective with NA = 0.25, 1cm focal length, 5mm lens diameter.
   Manufacturer: EDSCOR

**Camera Lens:** Long range microscope, 1.5 inch diameter lens.
   Manufacturer: Infinity Corp., Model MRM-1

**CCD Camera:** A NTSC monochrome camera, 512(H) by 480(V) pixels on a 0.5inch CCD, F1.4 lens with automatic gain control.
   Manufacturer: Panasonic, Model WV-BL200
Figure 2-2: Overview of the Optical System
2.3 Hardware Overview

With the exception of the custom-built digital interface card, all of the hardware items described below are standard and commercially available. The design of the custom card is documented fully in [1].

**Computer:** A standard IBM-PC compatible 20MHz 386 system.

   Manufacturer: ZEOS, Model Zeos 386-20

**Analog Interface:** This hardware is mounted in the computer backplane. This card has one digital to analog converter which is used to control the brightness of the light source. The converter has eight bits of accuracy, resulting in 256 possible brightness levels. The board also provides a general purpose digital input/output register to which two inputs are connected:

- A photosensor on the rotation stage which indicates that the stage is precisely positioned at zero degrees (used for automatic calibration).
- A photosensor which is mounted just in front of the microscope objective. When the light path to the photosensor is blocked, it warns that the object being tested is about to hit the objective.

   Manufacturer: Data Translation, Model DT 2808

**Frame Grabber:** This hardware\(^1\) is mounted in the computer backplane. This card is a NTSC video frame grabber. The card sports a variety of features, most of which are not used. There is one video input (from the camera), and multiple video outputs (one of which is connected to the camera display monitor). The card can operate either in continuous frame mode (for displaying real-time images on the monitor to be viewed by the experimenter) or in single frame mode (when sampling frames for the computer). Frame sampling occurs in real time.

\(^1\)The use of this hardware is not recommended. The hardware maps the camera frame buffer to IBM PC extended memory, and various bizarre behaviors have been observed in trying to access this memory.
Digital Interface: This hardware\(^2\) is located in the computer backplane. The development of the hardware is detailed in [1]; it basically provides a set of eight digital input and eight digital output lines. There are four sets of output lines (one set for each stepper motor on the stage’s four axes: X, Y, Z, and rotation); each set of lines is made up of a direction indicator (forward / backward) and pulse control. Motor acceleration, deceleration, and slew speed are under computer control via this pulse line. The actual control of the stepper motor windings is provided by standard hardware that comes packaged with the motors. A further six input lines are connected to limit switches that detect railing of any of the stage’s linear axes (X, Y, Z). Motion should not be attempted past the point set by these switches. These inputs are used both for safety (to prevent damage to the stage or the motors) and for calibration (since there is no other feedback to the computer giving the absolute position of each axis). Each single step of the motor is equivalent to 2.5 micrometers of motion.

Stepper Motors: High torque stepper motors capable of slewing at up to 10000 steps per second, 200 steps per rotation, 0.05 degree positioning accuracy.

Manufacturer: Stepper Motors: Superior Electronic, Model M062-FC03
Motor Controllers: Superior Electronic, Model 230TH

Micrometers: These are attached via a spring-loaded coupling to the stepper motors. A full turn of the micrometer results in 500 microns of linear motion (hence 2.5 microns of motion per stepper motor step). The micrometers are accurate to 0.0001 inch over one inch of travel.

Manufacturer: Starrett Manufacturing, Model 63M

\(^2\)This hardware is also not recommended - commands to the stepper motor drivers, which pass through this interface, are occasionally confused. The hardware is sufficiently simple, so future experimenters should redesign it, or purchase a standard off-the-shelf interface unit.
**Stage Bearings:** 2 inch long slide bearings; two are used on the X-axis, one on the Y-axis.
Manufacturer: Instrument Industries

### 2.4 Software Overview

The software developed for this thesis consists of a set of library routines, which are used by a variety of small application programs (calibration routines, example test programs etc.). Any group wishing to implement the concepts of this thesis in some practical setting (for automatic data acquisition in a manufacturing process, for example) will have its own particular software requirements which cannot be anticipated at this time; for this reason the author has not developed a fully featured user interface, since it is most probable that it would fail to meet the needs of any specific application. A primitive user interface was developed in [1]; although the software is rather limited in its capabilities, a group planning to write their own software may wish to consider that interface as a starting point.

All of the computer code is written in the “C” language. In general the code is intended to be computer and compiler independent, however some hardware interfacing functions are particular to the IBM PC environment, and some of the higher level interface functions rely on particular features of the Borland C compiler.

A complete listing of the hardware interface routines and a sample user interface are provided in [1]. The appendices in this thesis give a complete listing of all code referenced throughout the thesis, including system calibration programs (see Appendix A), theoretical calculation and simulation programs (see Appendix B), and various support functions (see Appendix ??).
Chapter 3

System Optics and Algorithm Theory

This chapter begins by discussing the optical theory required to understand the basic properties of the optical subset of the apparatus for the basic modes of system operation (listed below). The effect of scaling the whole apparatus is analyzed, and equations relating depth accuracy to system parameters are derived. The depth estimation algorithms are presented and their use is justified. Finally we assess certain performance limits of the depth estimation algorithms.

The accuracy of estimating depth from focus information will clearly depend on the characteristics of the measurement apparatus and the object being studied. Rather than attempting to catalogue accuracy for an impossibly wide range of objects, this thesis will concentrate on a few representative cases, the results of which may form benchmark accuracies to be met by future experiments. Three categories of operation are as follows:

- Ideally specular object having no surface detail (a mirror). This object is the most tractable analytically and also permits excellent depth estimation performance.

- Diffuse object having no surface detail (chalk, plastic).

- An object providing some surface detail (some metal).
The category appropriate for a particular object depends strongly on the level of magnification. Our apparatus operates on the order of microns, at which scale metals and paper both possess considerable detail. On the order of inches, however, paper is a nearly ideal diffuse surface and polished metals are specular.

3.1 Modes of Operation

Figure 2-1 shows the presence of a pattern that may be projected onto the object under study. The presence of the pattern is optional, resulting in two principal modes of system operation:

1. A pattern is used. This pattern is normally a regular two dimensional arrangement of discs; however squares, a line grid, or random shapes could in principle be used. A projected pattern is desired if the object being observed has insufficient surface detail to allow focus information to be extracted. Highly specular objects (eg, mirrors) and visually plain objects (eg, paper, plastic, smooth metals etc) fall into this category.

2. No pattern is used. In order for depth to be estimated from focus information, the object must have considerable visual detail scattered across its entire surface. For any surface element for which a depth estimate is desired, the estimate accuracy is a smooth monotonic function of the spatial cutoff frequency of the surface element’s image.

Clearly many objects will fail to fit perfectly into one of the above categories. Borderline cases (ie, objects having only some detail) should normally employ a projected pattern to ensure sufficient image detail.

3.2 Optical Theory

The state of modern optical theory permits an optical system, such as that used for this thesis, to be analyzed in anywhere between one and one thousand pages,
depending on the desired complexity. Past attempts [1, 3] attacked the problem of modeling the optical system with vigor, applied fairly advanced optical theory to the problem, found it to be intractable, and settled for a very simplistic classical model instead.

This thesis will take a different approach: it goes beyond the most simple classical models. However it stops short of the completely general, intractable equations which add little to intuitive understanding.

The purpose of developing the optical theory in this thesis is not to make detailed quantitative predictions, but rather to predict the general forms of results, justifying our intuitive understanding of the issues governing depth-from-focus experiments. As will be shown, a simplistic classical model is incapable of even demonstrating the correct qualitative results.

In general, an optical system may be characterized by its point spread function (PSF) \( h_d(x, y) \), which is the image formed in some image plane by a point source. For a general irradiance function (ie, the distribution of light sources), the PSF is a function of distance:

\[
i(x, y) = \int \int h(x - \alpha, y - \beta, d(\alpha, \beta)) u(\alpha, \beta) \, d\alpha \, d\beta
\]

where \( i(x, y) \) represents the created image, \( u(x, y) \) represents the irradiance function, \( h(x, y, d()) \) represents the generalized PSF, and \( d(x, y) \) represents the distance to \( u(x, y) \). If \( d(x, y) \) is nearly constant\(^1\) then

\[
i(x, y) = \int \int h(x - \alpha, y - \beta, d(\alpha, \beta)) u(\alpha, \beta) \, d\alpha \, d\beta
\]

\[
= \int \int h(x - \alpha, y - \beta, d) u(\alpha, \beta) \, d\alpha \, d\beta
\]

\[
= \int \int h_d(x - \alpha, y - \beta) u(\alpha, \beta) \, d\alpha \, d\beta
\]

\[
= h_d(x, y) \ast u(x, y)
\]

\(^1\)This assumption is true if the irradiance distribution is arranged in a plane normal to and proximate to the optical axis. These are very nearly the conditions present in this experiment.
Figure 3-1: Optical System for Specular Object

which is just a simple convolution. The optical transfer function (OTF) is the representation of the PSF in the Fourier transform domain:

\[
H_d(\omega_x, \omega_y) = \mathcal{F}\{h_d(x, y)\} \quad \text{(3.6)}
\]

\[
I(\omega_x, \omega_y) = \mathcal{F}\{i(x, y)\} = H_d(\omega_x, \omega_y)\mathcal{F}\{u(x, y)\} \quad \text{(3.7)}
\]

In this thesis we shall be making frequent references to 'blur', which is not well defined quantitatively, although humans have a strong intuitive appreciation for it. We shall adopt the following convention for this thesis:

In the spatial domain, the blur value is inversely proportional to the width of the main lobe of the PSF. In the frequency domain, the blur value is proportional to the cutoff frequency of the OTF. So as an image is unfocussed, its blur value decreases. Although both forms may be used, we will generally prefer using the frequency domain to characterize blur.

3.2.1 Perfectly Specular Object

We first consider the case of a perfectly specular object (we used a mirror). This case is developed in detail, since the theory is rather tractable and a number of predictions can be made.

The pertinent optical elements for this case are shown in Figure 3-1. In actual
fact there is only one objective lens, through which each light ray passes twice. For convenience, the optical path (as shown in the figure) is unfolded and the objective lens appears twice; the dashed line between the lenses represents the position of the mirror.

**Simple Classical Approach**

The analysis is straightforward. We start with the standard lensmaker equation[7]

\[
\frac{1}{o} + \frac{1}{i} = \frac{1}{f}
\]  

(3.8)

where \(o\), \(i\), and \(f\), are the object distance, image distance, and lens focal length respectively. With reference to the specific symbols used in Figure 3-1,

\[
d_{lf} = \left( \frac{1}{f} - \frac{1}{d_{sl}} \right)^{-1}
\]  

(3.9)

\[
d_{ft} = 2d_{im} - d_{lf}
\]  

(3.10)

\[
d_{it} = \left( \frac{1}{f} - \frac{1}{d_{ft}} \right)^{-1}
\]  

(3.11)

\(d_{l}, d_{sl}, d_{im},\) and \(d_{lp}\) are parameters which are fixed by the apparatus or under computer control.

There are two cases to consider:

1. \(d_{lf} > d_{im}\)

   The mirror lies in front of the focal position (this is the case shown in Figure 3-1). The size of the light cone is constrained by the diameter of the left lens. By similar triangles,

   \[
d_{k} = d_{l} \left( \frac{d_{ft}}{d_{lf}} \right) \left( \frac{d_{ki} - d_{lp}}{d_{it}} \right)
\]  

(3.12)

2. \(d_{lf} < d_{im}\)

   The mirror lies beyond the focal position. The size of the light cone is con-
Figure 3-2: Theoretical Blur Radius $d_b/2$ vs. Object Position

strained by the diameter of the right lens. By similar triangles,

$$d_b = d_t \left( \frac{d_{tp} - d_{ti}}{d_{ti}} \right)$$  \hspace{1cm} (3.13)

The above equations were simulated for a sample system having the following parameter values: $f = 0.007m$, $d_t = 0.005m$, $d_{il} = 0.15m$, $d_{tp} = 0.15m$. The resulting blur radius and image positions are shown in Figures 3-2 and 3-3 respectively. The blur radius $d_b/2$ is symmetric with respect to the object positioning about focus. The position of the image along the optical axis is asymmetric. Quantitative results should not be inferred from this simulation, since the effects of the camera and microscope assembly are not included. The predictions of this simulation are as follows: The blur value of an imaged point source is a strongly peaked function which is symmetric about the position of best focus.

Figure 3-4 compares a sketch of the above predictions with an experimentally derived result. There are two reasons why the experimental curve may be less peaked.
Figure 3-3: Theoretical Image Distance \((d_i - d_o)\) vs. Object Position

and flattens out on top:

1. Our above discussion does not consider the effects of spatial sampling by the CCD camera. A non-zero blur radius \(d_b\) much smaller than a CCD pixel radius will appear to have the same blur as \(d_b = 0\).

2. The above discussion, based on classical optics, does not include lens diffraction effects.

Spatial Sampling Effects

The intent of this section is to justify the remark from the previous paragraph, that CCD image sampling can round the peak of the blur measure curve.

A simple simulation was written that increasingly blurs a disc. For each level of blur the image is quantized into pixels, and a blur measure (see Section 3.4.1) is applied. The results of this simulation are shown in Figure 3-5.
Figure 3-4: Comparison of Simplistic and Experimental Blur Curves
Figure 3-5: Effect of CCD Image Sampling

As can be seen from the figure, the slope of the curve becomes steeper as the blur radius increases from zero – the blur value curve is decidedly flattened at its peak.

**Diffraction Effects**

There are two sources of diffraction in this experiment: the projected pattern (which, as a regular spacing of small holes, acts as a diffraction grating), and the microscope objective. If the mirror is positioned such that the pattern is in focus in the image plane, then the diffraction due to the pattern clearly has no effect. Diffraction from the pattern would be observed\(^2\) only if we were to image the light source itself (the light source may, for all intents and purposes, be considered to lie at infinity). The remaining source of diffraction, which is present in our experiments, is that due to the lens.

\(^2\)An experiment was set up and the diffraction effects were observed exactly as expected.
Figure 3-6: Basic Lens Diffraction Effects

Taking diffraction effects into account for lens systems is typically an involved undertaking. Hopkins[8] has derived some closed form expressions for optical transfer functions under various assumptions. Alternatively, one can start with the basic Fresnel (near field) form[9, 10]

\[
i(x_i, y_i) = \frac{1}{j \lambda z_i} \exp \left[ jk(z_i + \frac{x_i^2 + y_i^2}{2z_i}) \right] \mathcal{F}\{u(x, y)\}
\]  

(3.14)

where \( f_x = x_i/\lambda z_i \) and \( f_y = y_i/\lambda z_i \), or the Fraunhofer (far field) form

\[
i(x_i, y_i) = \exp \left[ \frac{jk(x_i^2 + y_i^2)}{2z_i} \right] \mathcal{F}\{u(x_0, y_0) f_z(x_0, y_0)\}
\]  

(3.15)

\[
f_z = \frac{1}{j \lambda z_i} \exp \left[ jk \left( z_i + \frac{x_i^2 + y_i^2}{2z_i} \right) \right]
\]  

(3.16)

The appropriate equation was applied to a circular aperture with a phase shift (representing the lens), and the resulting expression was simulated (see [1, p. 42]).

Ignoring the classical effects of the lens, the basic far field electromagnetic energy
distribution for diffraction through a plain circular aperture is [9]\(^3\)

\[ i(l) = \frac{1}{j \lambda z} \exp \left[ jk \left( z + \frac{l^2}{2z} \right) \right] \left( \frac{J_1(2\pi l d/2\lambda z)}{ld/2\lambda z} \right) \]  

(3.17)

as shown in Figure 3-6. Ignoring the phase and amplitude terms, this leads to Rayleigh’s criterion[7] that the first minimum in the observed intensity occurs at

\[ l = \frac{1.22\lambda z}{d} \]  

(3.18)

The simulation in [1] assumes a lens diameter \(d = 0.005m\), image distance \(z = 0.018m\), and wavelength \(\lambda = 750\, nm\). Our simple form (Equation 3.18) predicts a minimum at \(l = 32.5\, \mu m\), in excellent agreement with the simulated results[1, p. 45] at a fraction of the effort.

The magnification provided by the microscope objective is about a factor of 20, so on the scale of the object being viewed, the first zero occurs at \(32.5\, \mu m / 20 = 1.6\, \mu m\). From calibration tests performed on the camera (Chapter 4), each CCD pixel spans approximately 0.9\, \mu m along the object, so the first diffraction minimum should occur at \(1.6/0.9 = 1.8\) pixels. Since the diffraction PSF somewhat exceeds one pixel it is concluded that diffraction effects, rather than spatial sampling, is the slightly more significant factor in producing a rounded focus curve (Figure 3-4).

**Thick Lens Effects**

The rounded top of the experimental curve in Figure 3-4 has been justified. However the above work predicts a symmetrical curve (as do [1, 2, 3]) whereas the experimental results are strikingly asymmetrical (Figure 3-4).

A simple test was carried out (see Figure 3-7) using an incandescent light and a single lens. The lens is sufficiently large that classical optics should be sufficient to explain all observations; the system exhibits strong defocus asymmetries.

---

\(^3\)Because the image is circularly symmetric, \(i()\) is written in terms of a radial parameter only, ie, \(i(l) = i(x, y)|_{l = \sqrt{x^2 + y^2}}\)
Figure 3-7: Simple Laboratory Lens Focus Test

A simple ray tracing program (RayTrace.C, see Appendix B) was written to model the system of Figure 3-7, but in two dimensions only for simplicity. Lenses are modeled as two intersecting circles (after [6]) having user definable radii. Figure 3-8 shows the pertinent program variables.

Figure 3-9 shows the results for a complete simulation. The vertical axis measures light intensity, the horizontal axis is \( l \), that is, the position along the image screen. The depth axis shows the position of the image screen: on the front side of the plot the screen is located near to the lens, on the back side of the plot the screen is distant from the lens. Some readers may find the presentation in Figure 3-10 somewhat clearer; this figure displays a subset of eight plots taken from the entire ensemble shown in Figure 3-9 (the asymmetry of interest is the difference between Figure 3-10a and Figure 3-10b).

A few observations and words of caution are in order:

- The predicted form of light intensity as a function of screen position qualitatively matches our simple test (Figure 3-7) extremely well.

- The qualitative form of the asymmetry is maintained for any convex lens, regardless of the index of refraction or lens radii \( r_1 \) and \( r_2 \).

- The ray trace simulation is only two dimensional, so quantitative results (such

\(^4\text{ie, Light rays move in a plane (two dimensions); lenses are modeled as having thickness and height, but zero width.}\)
Figure 3-8: Ray Tracing Program Variables
Figure 3-9: Ray Tracing Program Results

as the relative heights of peaks in Figure 3-10) will not be correct.

- The simulation takes a continuous problem and discretizes it. The discretization introduces aliasing and averaging. The discretization is responsible for the asymmetry of curves in Figure 3-10a, and for the many unconnected peaks of Figure 3-9.

This concludes our study of the optical system for a highly specular object. There are several points that should be highlighted:

- The asymmetry of the focus curve is a striking feature which has a simple explanation. Future experimenters are advised to employ some simple ray tracing (or an equivalent experimental laboratory setup) in selecting an appropriate objective lens.

- The curve asymmetry cannot be deduced from thin lens diffraction methods as pursued by other authors.
Figure 3-10: Subset of Image Patterns from Ray Tracing
Illumination:

![Diagram of illumination](image)

Reflection:

![Diagram of reflection](image)

Figure 3-11: Optical System for Diffuse Object

- The flatness of the top of the curve has contributions from both spatial discretization and lens diffraction. In our case, diffraction dominates the effect.

### 3.2.2 Diffuse Object Without Detail

In this section we consider the case in which the object under study has diffuse reflectance properties, that is, the object has a stronger tendency to scatter light than to reflect it (e.g., chalk, paper, most rough materials).

A general outline of the optics involved is shown in Figure 3-11. The required
a) Surface Reflection, Small Dispersion

b) Internal Reflection, Large Dispersion

c) Multiple Reflections, Large Dispersion

Figure 3-12: Simple Demonstration of Light Point Spread

calculations to determine values for \( d_{b1} \) and \( d_{b2} \) are similar to before, and are repeated here only briefly.

For a lens having focal length \( f \),

\[
d_{lf} = \left( \frac{1}{f} - \frac{1}{d_{at}} \right)^{-1}
\]

(3.19)

\[
d_{b1} = d_{l} \left( \frac{|d_{lf} - d_{at}|}{d_{lf}} \right)
\]

(3.20)

\[
d_{ti} = \left( \frac{1}{f} - \frac{1}{d_{ao}} \right)^{-1}
\]

(3.21)

\[
d_{b2} = d_{l} \left( \frac{|d_{ti} - d_{ao}|}{d_{ti}} \right)
\]

(3.22)

A detailed discussion of the form of the observed camera image as a function of object depth would need to take into account the optical properties of the object’s surface. Two principal properties would need to be assessed:

- The intensity of scattered light as a function of the object viewing angle (eg, face-on or normal view, oblique view, etc).

- The surface point spread function (that is, the observed spatial light distribution for a point illumination).
The former issue is well documented[32, 33, 34], particularly since it plays an active part in computer graphics and modeling. The latter issue, determining the surface point spread function, is less well documented, perhaps since it is apparent only at higher magnifications (such as ours). The point spread function is caused by multiple reflections on the surface or even within the object (see Figure 3-12). Standard simple approaches (eg, [34]) model the surface as a collection of mirror facets oriented at various angles to the normal.

The PSF is a strong function of the material being observed, so a thorough treatment of the subject is well beyond the scope of this thesis. A simple PSF model is used in Section 5.4.1 when using chalk as a test object.

3.2.3 Object Having Surface Detail

The final object category is the class of objects having sufficient surface detail such that a projected pattern is not required (ie, the amount and density of spatial information available is comparable to that of the projected pattern). The class of possible objects is far too large to allow significant detail of discussion.

The optical setup corresponds to the bottom half of Figure 3-11, which shows the path of light from the object to the image plane (the camera). Similarly the equation describing the blur radius is given by Equation 3.22.

The manner in which the object's surface is modeled depends upon the magnification and the illumination:

- If a coherent light source is used (not the case for this thesis), the speckle produced may supply plenty of detail for focus purposes, however the speckle pattern may vary as the object is moved in depth, possibly complicating matters.

- At high magnifications (in particular for metals) the object surface detail may be due to surface roughness causing glint. This case will be explored in attempting to use various metals as test objects later in the thesis.
3.3 Optical System Scaling

This thesis deals with a particular depth estimation apparatus, built with components appropriate for a particular level of magnification (in this case, objects are measured on the order of microns). In order to extend the usefulness of this work, it is interesting to consider how the depth-from-focus resolution would change as the scale at which the unknown object is viewed changes.

There are a few limitations to this discussion which should be made clear up front:

- It is assumed that the nature of the object being observed (i.e., its reflectance properties) does not change as the system is scaled. This assumption would be valid for an object such as a mirror. A sheet of metal, on the other hand, although very much like a mirror on a scale of inches, exhibits very different properties at the micron level. The following results will thus typically yield pessimistic estimates for accuracy as the system is made larger, since the reflectance properties for most materials become cleaner.

- The only parameters being scaled are the dimensions explicitly shown in Figure 3-13. It is assumed, for example, that the dimensions of the CCD array within the camera remain fixed.

- This section concerns the manner in which the limit to optimal estimation performance changes with scale. If, on the other hand, a practical system is designed which improves object scanning speed by sampling more coarsely in depth (see Section 3.5.3), then a wider depth of focus can in certain cases improve accuracy (which runs counter to the predictions of this section).

We will first take a simple classical approach, followed by a modification of the results to take lens diffraction into account.

Figure 3-13 shows the basic elements and parameters to be considered in the following discussion. Since the camera is to observe the pattern focused on the object,
clearly \( d_p = d_i \).\(^5\)\(^6\)

### 3.3.1 Constant Projected Pattern

For constant equipment tolerances and object properties (see assumptions listed above):

\[
\text{Error} \propto \frac{\text{Size of Pattern Spot}}{\text{Rate of Change of Blur Radius over Object Range}} \tag{3.23}
\]

Since \( d_p = d_i \) and the pattern spot size is held constant,

\[
\text{Error} \propto (\text{Rate of Change of Blur Radius over Object Range})^{-1} \tag{3.24}
\]

Consider small deviations from the in-focus position. Let

\[
\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f} \tag{3.25}
\]

where \( f \) is the focal length of the lens. Then

\[
\text{At } d_o + \delta : \quad d_{i+} = \left(\frac{1}{f} - \frac{1}{d_o + \delta}\right)^{-1} \quad d_{b+} = d_i \frac{d_i - d_{i+}}{d_{i+}} \tag{3.26}
\]

\(^5\)In the case of a perfect mirror this requirement does not hold, since a focus can always be established. General objects being measured are not perfect mirrors however.

\(^6\)Because the camera may include additional optics (as does our system, see Figure 2-1) the physical distance to the camera on the optical table may not equal the distance to the pattern. Our discussion ignores camera optics and uses an idealized image plane instead.
At $d_o - \delta$:

$$d_i = \left( \frac{1}{f} - \frac{1}{d_o - \delta} \right)^{-1} \quad d_o = d_i \frac{d_o - d_i}{d_i}$$ (3.27)

For small values of $\delta$ we use a standard first order approximation:

$$\frac{1}{a + \delta} \simeq \frac{1}{a} - \frac{\delta}{a^2}$$ (3.28)

$$d_i+ \simeq d_i - \delta \frac{d_i}{d_o}$$ (3.29)

$$d_i- \simeq d_i + \delta \frac{d_i}{d_o}$$ (3.30)

So then with the calculated values for the blur radius (Equation 3.26):

$$\frac{1}{\text{Error}} \propto \frac{d_o + d_i}{2\delta}$$ (3.31)

$$= \frac{d_i(d_i - d_i)/d_i + d_i(d_i - d_i)/d_i}{2\delta}$$ (3.32)

$$\lesssim \frac{d_i}{2\delta} \left( \frac{d_i}{d_i - \delta \frac{d_i}{d_o}} - 1 + \frac{d_i}{1 + \delta \frac{d_i}{d_o}} \right)$$ (3.33)

$$= \frac{d_i}{2\delta} \left( \frac{1}{1 - \delta \frac{d_i}{d_o}} - 1 + \frac{1}{1 + \delta \frac{d_i}{d_o}} \right)$$ (3.34)

$$\lesssim \frac{d_i}{2\delta} \frac{d_i}{d_o}$$ (3.35)

$$\frac{1}{\text{Error}} \propto d_i \frac{d_i}{d_o}$$ (3.36)

Note that the parameter $f$ is present implicitly in this result, since $d_i, d_o$, and $f$ are not independent (see Equation 3.25).

For a fixed camera size, the field of view is also determined implicitly in this result, since

$$\text{Field of View} \propto \frac{d_o}{d_i}$$ (3.37)

Thus, as would be expected, reducing the error and enlarging the field of view are competing objectives.
3.3.2 Object Having Surface Detail

The above argument applies to the surface detail case as well, except that the perceived size of detail on the object changes as $d_o$ and $d_i$ vary. In particular, for some feature on the object, its perceived size (as seen by the camera) is

$$\text{Object Size } \propto \frac{d_i}{d_o} \quad (3.38)$$

So based on Equations 3.23 and 3.36 we have

$$\text{Error } \propto \frac{d_o}{d_i} \quad (3.39)$$

This makes sense intuitively: if a pattern is used, the pattern spot size on the object is proportional to $d_o/d_p = d_o/d_i$; since the object feature size is assumed constant, the constant of proportionality $d_i/d_o$ is lost.

3.3.3 Scaling with Diffraction Effects

Equation 3.36 suggests that the error can be reduced indefinitely by increasing $d_i$. Aside from the practical experimental limits to such an attempt, there are also limits imposed by diffraction effects.

For the continuous two-dimensional Fourier transform[42],

$$f(x, y) = \begin{cases} 
1 & x^2 + y^2 < r^2 \\
0 & \text{otherwise}
\end{cases} \quad \Leftrightarrow \quad F(\omega_1, \omega_2) = 4\pi \frac{J_1(r \sqrt{\omega_1^2 + \omega_2^2})}{\sqrt{\omega_1^2 + \omega_2^2}} \quad (3.40)$$

where $F()$ has a cutoff frequency\(^7\) of $\sqrt{\omega_1^2 + \omega_2^2} = 2.215/r$ and a first zero at $4/r$. By duality, a Bessel spatial distribution having a first zero at radial distance $r_1$ has a frequency cutoff at $\omega = 4/r_1$.

Three frequency domain distributions need to be considered:

\(^7\)The cutoff frequency is taken here to be the frequency at which the frequency response drops to one half of the DC level.
1. $F_S$ - due to the projected spot (radius $r_s$).

2. $F_B$ - the OTF, assuming simple classical optics\(^8\) (radius $r_b$).

3. $F_L$ - the OTF due to lens diffraction (first zero at radius $r_l$).

The distribution observed by the camera is the convolution

$$F_{\text{Total}} = F_S \cdot F_B \cdot F_L$$  \hspace{1cm} (3.41)

Let $\omega_f$ be the cutoff frequency of the in-focus distribution $F_S \cdot F_L$, so

$$\omega_f = \min \left( \frac{2.215}{r_s}, \frac{4}{r_l} \right)$$  \hspace{1cm} (3.42)

Let the limits of the range-of-focus be the points at which blurring causes the cutoff to drop by a factor of two, i.e.,

$$\frac{\omega_f}{2} = \frac{2.215}{r_b}$$  \hspace{1cm} (3.43)

If the spot radius on the pattern itself is $r_o$, then

$$r_s = r_o \frac{d_i}{d_p}$$  \hspace{1cm} (3.44)

Rayleigh’s criterion\(^7\) gives the value for $r_l$:

$$r_l = \frac{1.22 \lambda d_i}{d_l}$$  \hspace{1cm} (3.45)

We assume, as before, that $d_p = d_i$, hence $r_o = r_s$. The apparatus is diffraction limited when increasing $d_i$ has no effect on $\omega_f$:

$$\frac{2.215 d_p}{r_o d_i} = \frac{4 d_l}{1.22 \lambda d_i}$$  \hspace{1cm} (3.46)

$$d_{\text{imax}} = \frac{4 d_r r_o}{2.5 \lambda}$$  \hspace{1cm} (3.47)

---

\(^8\)This is a poor model, as discussed earlier in optical theory, however it suffices for the approximate calculations presented here.
Considering the approximations made in this development, one should constrain \( d_i \) to one half the limiting value as a safety margin. For typical parameter values used in our apparatus (see Figure 2-2), we find the limiting value to be \( d_{\text{max}} \approx 0.6m \) where the actual value is \( d_i \approx 0.2m \), suggesting that our apparatus is approaching diffraction limited accuracy.

To conclude, the actual range-of-focus is calculated. From Equation 3.36 we know that

\[
\text{Blur Diameter} = 2r_b = \frac{d_i d_i \delta}{d_s} \quad (3.48)
\]

\[
\omega_f = 4 \cdot 2.215 \frac{d_i^2}{d_i d_s} \quad (3.49)
\]

\[
\delta = \frac{8.86d_i^2}{d_i d_i \omega_f} \quad (3.50)
\]

For the values of our apparatus, this evaluates to \( \delta \approx 10\mu m \).

### 3.3.4 Practical Issues

The manner in which error varies with system parameters, as determined in the above sections, ignores many practical issues that may constrain performance well before the limits of the previous sections are met.

- It was assumed that the depth-from-focus error is governed by the depth of focus and optical constraints; mechanical equipment tolerances do not enter the picture. This may be a poor assumption for the extremes in scale: a very large system loses rigidity, a very small system may have difficulty meeting tolerances for stage positioning.

- There are practical constraints to lens diameters and focal lengths. Large lenses (large \( d_i \)) become prohibitively expensive and may sag under their own weight, reducing performance. Practical lower limits on lens focal lengths are determined by available indices of refraction and limits to lens curvature.

- If an attempt is made to improve accuracy by selecting a very large \( d_i \) one might wish to "fold" the optical path to avoid having the camera an unrealistic
distance from the rest of the apparatus (for example folding the optical path into multiple reflections between two mirrors). This will not affect the results of the previous sections, since the diffraction pattern of the objective lens is just magnified in the same way as the rest of the image.\footnote{Optical folding could increase estimation error if lenses or mirrors were used having a comparable diameter to the objective lens, since these would add additional diffraction contributions.}

- In many cases, the ultimate limit to performance may be due to a minimum required field of view. In this case, Equation 3.36 becomes

\[
\text{Error} \propto \frac{\text{Field of View} \cdot d_c}{d_t} \quad (3.51)
\]

so increasing \( d_t \) has no effect. The field of view may always be increased by using a CCD array having more pixels.

### 3.4 Depth Estimation Algorithms

There are two sub-algorithms that need to be determined in order to convert a sequence of camera images to a depth estimate (see Figure 3-14):

![Figure 3-14: Outline of Depth Estimation Process](image)
1. Conversion of a single \( N \) by \( N \) pixel subimage from the CCD camera to a scalar, known as a blur value (the operator that generates the blur value is called a blur measure). The scalar value should increase monotonically as the image is brought into focus.

2. Conversion of a set of blur values to a single depth estimate.

### 3.4.1 Blur Measures

A blur measure is an operator, accepting a section of an image as input, and producing a real scalar\(^{10}\) as output. A great many such operators can be imagined; for our purposes any acceptable operator must, at a minimum, fulfill the following requirements:

- The scalar result of the operator must be a monotonic function of the degree to which the object is in focus. A completely unfocussed image yields a blur measure value of zero.

- Since the position of the spot projected by the pattern may drift slightly with time or depth, the operator must be invariant over small shifts of the image.

- Since the illumination of the object may not be uniform, the operator must be illumination level invariant.

From earlier work in this chapter (Section 3.2.1) we know that as any scene is increasingly blurred, it is effectively convolved with a low pass filter of decreasing cutoff frequency. Candidate blur measures are operators that calculate some function of frequency domain coefficients.

The position invariance criterion eliminates common image processing algorithms such as the DCT\(^{[40, 42]}\) or LOT\(^{[43, 44]}\). On the other hand, both of these transforms deal more effectively with brightness discontinuities at the image edges than does the FFT. Future researchers may wish to address this issue further.

\(^{10}\)It is conceivable that a vector could be produced instead (eg, a set of FFT coefficients). This would require a different depth estimator than the one presented in this thesis.
Assuming that the main feature of interest (ie, the projected spot if a pattern is
used) stays within the image region being considered, the FFT is shift invariant\(^{11}\) and
image edge discontinuities will not be a concern. The following sections discuss the
two blur measures which have been selected for use.

**Image Normalization**

Let the illumination intensity on the object be \(I(x,y)\) and the surface reflectivity
\(R(x,y)\). Then, assuming an infinitesimal point spread function, the observed light
intensity is \(I(x,y) \cdot R(x,y)\). If the intensity varies slowly (as it should for any rea-
sonable light source), then over small sections of the object, the intensity \(I()\) may be
assumed constant.

The effect of varying illumination is removed by normalizing each section of the
camera independently. If the received camera image is \(d(x,y)\) and the normalized
image is \(\hat{d}(x,y)\), then for any particular image segment,

\[
d(x,y) = I(x,y) \cdot R(x,y) = I \cdot R(x,y) \tag{3.52}
\]

\[
\hat{d}(x,y) = \frac{d(x,y) \sum_{x}^{N} \sum_{y}^{N} 1}{\sum_{x}^{N} \sum_{y}^{N} d(x,y)} \tag{3.53}
\]

\[
= \frac{N^2 R(x,y)}{\sum_{x}^{N} \sum_{y}^{N} R(x,y)} \tag{3.54}
\]

which is independent of the illumination. Such a normalization routine may be found
at the end of program Cal\_Curve.C (Appendix A).

**Difference–Squared Measure**

A simple, computationally light, algorithm is the difference-squared measure. Given
a section of camera data

\[
d(x,y) \quad 0 \leq x < N, \quad 0 \leq y < N \tag{3.55}
\]

\(^{11}\)The FFT is shift invariant to within a complex phase factor. Our blur measures are always real,
so only the magnitudes of FFT coefficients will be considered.
the difference-squared blur measure is defined as

$$\text{DiffSq} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-2} \left( (d(i, j) - d(i, j + 1))^2 + (d(j, i) - d(j + 1, i))^2 \right)$$  \hspace{0.5cm} (3.56)$$

That is, the blur measure equals the sum of squared differences of all vertically and horizontally adjacent pixel pairs.

It is not immediately clear that this operator is a member of the family of weighted FFT coefficients, however a quick argument can demonstrate this. Let

$$d(i, j) - d(i, j + 1) = h_y(i, j) * d(i, j)$$  \hspace{0.5cm} (3.57)$$
$$d(j, i) - d(j + 1, i) = h_x(j, i) * d(j, i)$$  \hspace{0.5cm} (3.58)$$
$$H_x(k_x, k_y) = \mathcal{F}\{h_x(i, j)\} \quad \quad H_y(k_x, k_y) = \mathcal{F}\{h_y(i, j)\}$$  \hspace{0.5cm} (3.59)$$

where * refers to the linear two dimensional convolution operator[42]. Then from Equation 3.56 and using Parseval’s relation for two dimensional signals,

$$\text{DiffSq} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-2} \left( (d(i, j) - d(i, j + 1))^2 + (d(j, i) - d(j + 1, i))^2 \right)$$  \hspace{0.5cm} (3.60)$$
$$= \sum_{i=0}^{N-1} \sum_{j=0}^{N-2} \left( (h_y(i, j) * d(i, j))^2 + (h_x(j, i) * d(j, i))^2 \right)$$  \hspace{0.5cm} (3.61)$$
$$\approx \frac{1}{N^2} \sum_{k_x=0}^{N-1} \sum_{k_y=0}^{N-1} |\mathcal{F}\{d(i, j)\}|^2 \cdot (|H_x(k_x, k_y)|^2 + |H_y(k_x, k_y)|^2)$$  \hspace{0.5cm} (3.62)$$

which is clearly a sum of weighted squared FFT coefficients. The form of these weights is easily calculated:

$$h_x(i, j) = \delta(i, j) - \delta(i + 1, j)$$  \hspace{0.5cm} (3.63)$$
$$H_x(k_x, k_y) = \left(1 - e^{j2\pi k_x/N}\right)$$  \hspace{0.5cm} (3.64)$$

A similar calculation holds for $H_y$. The weights thus increase with frequency (with a weight of zero at DC, and a weight of eight\textsuperscript{12} at the highest frequency component).

\textsuperscript{12}$k_x = N/2, k_y = N/2, H_x(k_x, k_y) = H_y(k_x, k_y) = 2$. So $H_x()^2 + H_y()^2 = 8$. 

54
FFT Blur Measure

Although the difference-squared measure just discussed belongs to the family of operators derived from weighted FFT coefficients, it is calculated directly in the spatial domain. Alternatively, albeit at considerable computational cost, we can choose to apply the blur measure explicitly in the frequency domain by calculating the FFT and selecting various coefficients and subjecting them to certain weights.

It is difficult to say a priori what set of weights forms an optimal blur measure. Indeed, the choice of weights may depend somewhat on the camera resolution, the type of projected pattern (if any), and the material being observed. A quick series of tests showed the four lowest frequency FFT coefficients\(^{13}\) to be fairly “clean”\(^ {14}\); these four coefficients form the basis of a blur measure to be tested:

$$FFTMeas = |D(1,0)|^2 + |D(0,1)|^2 + |D(1,1)|^2 + |D(1,-1)|^2$$

(3.65)

where \(D() = \mathcal{F}\{d()\}\), \(d()\) represents the camera image. Certainly additional research is required to determine a less ad-hoc coefficient selection.

3.4.2 Estimators

An estimator, in the general sense, is an algorithm that determines the value of some parameter given a set of data points\(^ {39}\). In this case, our estimator is given a set of blur measure values (sampled at equal increments in depth as the object is moved along the Z axis), possibly also a reference curve, and is required to determine the depth of the surface on the object corresponding to these values.

Let \(b(n)\) represent the blur measure value as a function of depth \(n \in \mathcal{I}\); \(b(n)\) is a discrete function. From our requirements for a blur measure (defined in the previous

\(^ {13}\)ie, \((k_x, k_y) = \{(1,0), (0,1), (1,1), (1,-1)\}\)

\(^ {14}\)Clean in the sense that the coefficient values were a smooth function of focus and consistent across different pattern spots. Higher frequency coefficients were less correlated between spots and yielded ragged functions of focus.
section), we know that

\[
\lim_{n \to -\infty} b(n) = \lim_{n \to \infty} b(n) = 0 \tag{3.66}
\]

so \(b(n)\) has finite energy.

Let the continuous function \(c(d)\) represent the ideal blur measure (determined from an ideal system devoid of noise). The estimation problem is as follows. Given \(y(n)\) and \(b(n)\), estimate the value of \(\delta\):

\[
y(n) = c(\Delta n) \tag{3.67}
\]

\[
b(n) = \alpha c(\Delta n - \delta) + w(n) \tag{3.68}
\]

where \(y(n)\) represents the discrete calibration curve actually available to us, \(\Delta\) measures the separation in depth between successive samples of \(b(n)\) or \(y(n)\), \(\alpha\) is an amplitude scaling factor, \(w(n)\) is a noise function,\(^{15}\) and \(\delta\) represents the offset in depth between \(y(n)\) and \(b(n)\). \(\delta\) is the parameter that we wish to estimate.

This is a nonlinear estimation problem, in that \(\delta\) does not enter the problem as a simple multiplicative constant. Rather \(\delta\) is the offset or delay between two curves. Certain approaches exist which linearize such estimation problems (for example [35]), however these inevitably incur additional error in the estimate. Given the premise that we are aiming to optimize depth accuracy rather than estimation speed we will utilize relatively slow nonlinear estimators in this thesis. The development of appropriate linearized estimators may be a focus of future research.

**Correlation Estimator**

If the value of \(\alpha\) is unknown, a correlator and peak detector form a robust estimator:

\[
\hat{\delta} = \arg\max_{\delta \in \mathcal{I}} \left\{ \sum_{i=-\infty}^{\infty} y(i) \cdot b(i - \delta) \right\} \quad \hat{\delta} \in \mathcal{I} \tag{3.69}
\]

\(^{15}\)Assumed to be white Gaussian noise for now. This noise is considered in further detail in Chapter 5.
The algorithm presented in this equation limits the accuracy of our estimate \( \hat{\delta} \) by constraining it to have integer values. In continuous time, the correlator estimator exhibits no such constraints; we may be inclined to employ linear interpolation on \( y(n) \) or \( b(n) \) to improve performance:

\[
\hat{\delta} = \text{arg}_\delta \max \left\{ \sum_{i=-\infty}^{\infty} y(i) \cdot \left[ (\beta b(i - \lfloor \delta \rfloor)) + (1 - \beta)b(i - \lfloor \delta \rfloor - 1)) \right] \right\} \quad \delta \in \mathbb{R} \tag{3.70}
\]

where \( \beta = \delta - \lfloor \delta \rfloor \). Let

\[
f(d) = \sum_{i=-\infty}^{\infty} y(i) \cdot b(i - d) \tag{3.71}
\]

Then Equation 3.70 becomes

\[
\hat{\delta} = \text{arg}_\delta \max \left\{ \beta f(\lfloor \delta \rfloor)) + (1 - \beta)f(\lfloor \delta \rfloor + 1) \right\} \leq \max(f(\lfloor \delta \rfloor), f(\lfloor \delta \rfloor + 1)) \tag{3.72}
\]

That is, using linear interpolation, the correlation is never maximized for non-integer values of \( \delta \); so the use of linear interpolation is fruitless.

**ML Estimation**

An alternative estimation scheme uses a maximum likelihood approach.\(^{16}\) The maximum likelihood estimation equation is sensitive (not robust) with respect to unknown scaling factors such as \( \alpha \); in the following, the value of \( \alpha \) is presumed known. Define \( g(d) \) as follows:

\[
\hat{\delta} = \text{arg}_\delta \min g(\delta) = \text{arg}_\delta \min \sum_{i=-\infty}^{\infty} \left( \alpha y(i) - b(i - \delta) \right)^2 \tag{3.73}
\]

\[
= \text{arg}_\delta \min \sum_{i=-\infty}^{\infty} \left( \alpha^2 y^2(i) + b^2(i) \right) - 2\alpha f(\delta) \tag{3.74}
\]

where \( f() \) is defined as in Equation 3.71. Clearly in the continuous case, where the sampling interval \( \Delta \) goes to zero and sums become integrals, then Equation 3.74 and

---

\(^{16}\)This is not to say that the estimator is maximum likelihood (ML), but rather the equation used is a standard ML form applied to discrete functions in Gaussian white noise.
Equation 3.69 differ only by a sign and a constant offset.

As before, the form of Equation 3.74 allows integer estimates only, so we attempt to improve these estimates using linear interpolation. Let

\[
h(d) = \sum_{i=-\infty}^{\infty} (\alpha y(i) - \beta b(i - d - 1) - (1 - \beta) b(i - d))^2, \quad 0 \leq \beta \leq 1 \quad (3.75)
\]

where our depth estimate is obtained by minimizing \(h(d)\) over both \(d\) and \(\beta\). Then

\[
h(d) = \sum_{i=-\infty}^{\infty} (\beta (\alpha y(i) - b(i - d - 1)) + (1 - \beta) (\alpha y(i) - b(i - d)))^2 \quad (3.76)
\]

\[
= \beta^2 g(d+1) + (1 - \beta)^2 g(d) + \sum_{i=-\infty}^{\infty} 2\beta(1 - \beta) (\alpha y(i) - b(i - d - 1)) (\alpha y(i) - b(i - d)) \quad (3.77)
\]

\[
= \sum_{i=-\infty}^{\infty} \alpha^2 y^2(i) + (1 - 2\beta(1 - \beta)) \sum_{i=-\infty}^{\infty} b^2(i) + 2\beta(1 - \beta) \sum_{i=-\infty}^{\infty} b(i - d)b(i - d - 1) - 2\beta \alpha f(d+1) - 2(1 - \beta) \alpha f(d) \quad (3.78)
\]

If subsample resolution is possible, then one of

\[
h(d) - g(d) < 0 \quad \text{or} \quad h(d) - g(d+1) < 0 \quad (3.79)
\]

must be possible. For example

\[
h(d) - g(d) = -2\beta(1 - \beta) \sum_{i=-\infty}^{\infty} \{b^2(i) - b(i)b(i - 1)\} - 2\beta \alpha (f(d+1) - f(d)) < 0 \quad (3.80)
\]

Since \(\sum \{b^2(i) - b(i)b(i - 1)\} \geq 0\), and \(f(d+1) - f(d)\) may be positive, negative, or zero, clearly the inequality in Equation 3.79 can hold. In particular, when \(f(d+1) = f(d)\), then a non-integer value for \(\beta\) will satisfy the minimization of \(h(d)\). Such a solution for \(\beta\) implies a noninteger estimate \(\hat{\delta}\). 

58
Hybrid Estimator

A simple correlator estimator is robust in that it is insensitive to linear scaling factors, however the precision of estimation, even using linear interpolation, is limited to the sampling step. The maximum likelihood type estimator, although intolerant of scaling factors, can provide subsample resolution. Thus the following hybrid estimator is recommended:

1. Use correlation (without interpolation) to approximate $\hat{\delta}$ and to establish an amplitude estimate $\hat{\alpha}$. A wide range of $\hat{\delta}$ values is searched.

2. Use maximum likelihood, with linear interpolation, within a constrained region about the estimate $\hat{\delta}$ already determined. Divide $b()$ by $\hat{\alpha}$ to scale it approximately to the level of $y()$.

This estimation algorithm was implemented and has met with considerable success. Some modifications of the above may be appropriate in certain applications:

- Under highly controlled conditions, the value of $\alpha$ may be known a priori. In this case the correlation step might be omitted.

- If the desired depth precision is on the order of the depth sampling size, then the maximum likelihood step might be omitted.

3.5 System Performance

This section begins to investigate the sources of noise in the system, and the effect they may have on the estimation process. In this section we shall concentrate on developing the appropriate theory for two sources of error. Other less tractable noise terms, which require empirical data, are addressed in Chapter 5.

3.5.1 Overview of System Noise Sources

The following items comprise the primary sources of noise in the apparatus used for this thesis (refer to Chapter 4 for additional discussion):
• **CCD Camera Noise:** the noise contributions from the camera are divided into three categories:

1. Random noise: each CCD element accumulates charge as it is exposed to radiation. The amount of charge present is a random process (Poisson) frequently modeled as white noise.

2. Pixel offset noise: the flux of incident radiation for a particular pixel is proportional to its surface area. Slight variations in area between pixels yield deterministic differences in the expectation value returned by the pixels. In principle a correction map could be determined to remove this effect.\(^\text{17}\)

3. Aliasing Noise: the scene being viewed is spatially continuous, and is discretized by the CCD array. This discretization causes aliasing and adds noise (eg, the difference–squared blur measure returns different values for an image edge centered on a row of pixels as opposed to lying between two rows of pixels).

• **Variation Between Pattern Elements:** if the projected pattern is not completely uniform, then the blur measure values taken at different points on the object may vary, even though the object has uniform reflectance properties. Differences in spot sizes (resulting only in differing intensities of illumination) are tolerated by the estimator, however differences in spot shapes may yield fundamentally different blur curves.

• **Shape of Test Flat:** the test mirror used in experiments is assumed to be perfectly flat. Although perfect flatness is a reasonable assumption in most cases, certain high precision experiments may challenge this assumption.

\(^{17}\)Such a correction was not made here. It is a cumbersome correction, requiring large disk files and computer memory. In addition, the calibration of pixel area requires illumination by a light source whose illumination varies very little (spatially); the author is not convinced that the light source used in this apparatus is not sufficiently uniform for such a calibration.
• **Stepper Motor and Micrometer Linearity:** We consider three categories of micrometer errors:

1. Periodic variations: the micrometer advances in a nonlinear periodic fashion\(^{18}\) (the period being one complete turn of the micrometer). Large variations may be detected and the micrometer replaced; small variations must be tolerated.

2. Jitter: the advance of the micrometer is not completely smooth (among other things, caused by small particles located in the screw).

3. Backlash: when the micrometer turning direction is reversed, the first bit of turning may have no effect because of play in the micrometer screw. The code written for this thesis includes provisions to remove micrometer backlash.

• **Miscellaneous Stage Motions:** the stage is not perfectly rigid and may shift or vibrate over time. The stage may change size or shape due to thermal stresses.

### 3.5.2 Effects of Camera White Noise

In the following image pixels are assumed to be independent, Gaussian random variables each having variance \(\sigma^2\). Image pixel noise is quantified in Section 4.4.2.

**Blur Measure Noise**

The following derivation establishes the level of noise present in a FFT based blur measure as a function of image variance \(\sigma^2\).

Let \(v(t)^{19}\) be the discrete zero mean\(^{20}\) input signal, where \(\text{var}(v(t_o)) = \sigma^2\). Let

---

\(^{18}\)This effect is particularly severe on one of our micrometers, see Section 4.2.

\(^{19}\)We use one dimensional signals for simplicity. The argument is unchanged in two dimensions.

\(^{20}\)Any real image clearly has non-zero mean pixel values. A non-zero mean just adds offset terms throughout the following argument.
\( \phi_i(t) \) represent the FFT analysis functions:

\[
a_i = \sum_t \phi_i(t)v(t)
\]

\[
\sum_t \phi_i(t)\phi_i^*(t) = \delta_{ij}
\]

ie, \( a_i \) are the Fourier coefficients. Then

\[
E[a_i a_j^*] = E \left[ \left( \sum_t \phi_i(t)v(t) \right) \left( \sum_t \phi_j(t)v(t) \right)^* \right]
\]

\[
= E \left[ \sum_t \sum_s \phi_i(t)\phi_j^*(s)v(t)v(s) \right] = \sigma^2 \delta_{ij}
\]

\[
E[|a_i|^2] = \sigma^2
\]

and clearly \( Re(a_i) \) and \( Im(a_i) \) are Gaussian random variables, whereas \( |a_i| \) is Rayleigh distributed[36]; also the Fourier coefficients \( a_i \) are independent.

For some input image \( i(x, y) \), let the blur measure be defined by

\[
\text{Blur Value} = \sum_{x=1}^{N} \sum_{y=1}^{N} \{i(x, y) * h(x, y)\}
\]

\[
= \frac{1}{N^2} \sum_{x=1}^{N} \sum_{y=1}^{N} |I(k_x, k_y)|^2 |H(k_x, k_y)|^2
\]

\[
= \frac{1}{N^2} \sum_{x=1}^{N} \sum_{y=1}^{N} |I(k_x, k_y)|^2 \alpha(k_x, k_y) \ \alpha(k_x, k_y) \in \mathcal{R}
\]

Then, assuming that the camera noise is independent of the image (Section 4.4.2),

\[
i(x, y) = \text{Ideal Signal}(x, y) + \text{Noise}(x, y)
\]

\[
= f(x, y) + w(x, y)
\]

\[
\mathcal{F}\{i()\} = \mathcal{F}\{f(x, y)\} + \mathcal{F}\{w(x, y)\}
\]

\[
E \left[ |I(k_x, k_y)|^2 \right] = |\mathcal{F}\{f(x, y)\}|^2 + |\mathcal{F}\{w(x, y)\}|^2
\]

So then the expected blur measure value

\[
E \left[ \text{Blur Value} \right] = E \left[ \sum_{x=1}^{N} \sum_{y=1}^{N} \alpha(k_x, k_y) |I(k_x, k_y)|^2 \right]
\]
\[ E [\text{Blur Value, No Noise}] + \sum_{N}^{\sum_{N}} \alpha(k_x, k_y) \sigma^2 \]  

(3.94)

deo, for this class of blur measures the expected value increases in the presence of noise.\(^{21}\) The variance of the blur measure is of greater interest however (the algebra is simple but long; several steps are omitted):

\[
\text{var(Blur Value)} = E \left[ \left( \sum_{N}^{\sum_{N}} \alpha(k_x, k_y) I(k_x, k_y) I^*(k_x, k_y) \right)^2 \right] 
- \left\{ E \left[ \sum_{N}^{\sum_{N}} \alpha(k_x, k_y) I(k_x, k_y) I^*(k_x, k_y) \right] \right\}^2
\]

(3.95)

\[
\sum_{N}^{\sum_{N}} \alpha(k_x, k_y)^2 \left( 4 |I(k_x, k_y)|^2 \sigma^2 + 2\sigma^4 \right)
\]

(3.96)

So, for non-negligible values for \(I()\), the variance of the blur measure should be roughly proportional to the blur measure value, so the signal to noise ratio improves for larger blur measure values. The above result will be tested against experiment in Section 5.1.

**White Noise Impact on Estimator**

Because our estimator operates on a discretized rather than continuous depth scale, it is difficult to determine analytically the variance of the estimate due to additive white noise. Instead, we will approximate the variance using standard results from continuous time, and compare these to a discrete time simulation.

In continuous time, the unknown position of a \(\sin c\) function may be estimated using a matched filter\([39]\); for some parameter \(x\) to be estimated, the Cramer-Rao bound (the variance of the optimal unbiased estimator) is

\[
E [(x - \hat{x})^2] = \frac{3q}{(2\pi W)^2 E}
\]

(3.97)

---

\(^{21}\)The noise values in the frequency domain are treated as Gaussian, although their distribution is actually Rayleigh. However even for modest values of \(N\) the central limit theorem \([37]\) justifies the Gaussian approximation.
Where: observed signal noise variance \( V = 2wq \)
first sinc minimum near peak \( M = 1/2W \)
height of sinc peak \( P = \sqrt{2W E} \)

So in the above units,
\[
E [(x - \hat{x})^2] = \frac{3VM^2}{\pi^2 P^2} \tag{3.98}
\]

The use of a sinc function is motivated by its analytical tractability and its rough similarity to experimental blur curves (Figure 3-4).

The discrete simulation (very similar to program Sim_Sampling.C in Appendix B) approximated the asymmetric blur measure curve (see Figure 3-15) as follows:
\[
f_-(d) = A \exp \left( -\ln(2) \cdot \frac{d^2}{C^2} \right) \tag{3.99}
\]
\[
f_+(d) = A \exp \left( -\ln(2) \cdot \frac{d^2}{B^2} \right) \tag{3.100}
\]

The curve was sampled at a constant spacing with random offsets, and processed by the hybrid correlator-maximum likelihood estimator. The continuous and discrete time results are compared in Table 3.1. The RMS error values for the simulation are comparable to the continuous time CRB (except for small values of \( \sigma^{22} \)), validating the intuitive expectation that the CRB predictions be applicable in this scenario.

### 3.5.3 Effects of Sampling Size

A practical implementation of depth estimation may opt to increase speed by sampling in steps coarser than those permitted by the mechanical apparatus. By spacing our data points farther apart in depth, we are subsampling the blur measure curve. It may be useful to develop some intuition regarding performance changes as the data point spacing is varied.

The problem is set up as follows:

\[^{22}\text{The lower limit in the RMS value of the estimate from a DT estimator is due to the step} \Delta \text{ between successive possible estimates. For no noise and a uniform probability distribution for the parameter being estimated, RMS Error} = \Delta/\sqrt{12}. \text{In our simulations,} \Delta = 0.125, \text{predicting a lower RMS error limit of about} 0.036.\]
Figure 3-15: Idealized Blur Curve for Simulation Program

<table>
<thead>
<tr>
<th>Noise $\sigma$</th>
<th>Estimate RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Test 1</td>
</tr>
<tr>
<td>0.00</td>
<td>0.000</td>
</tr>
<tr>
<td>0.05</td>
<td>0.021</td>
</tr>
<tr>
<td>0.10</td>
<td>0.042</td>
</tr>
<tr>
<td>0.15</td>
<td>0.063</td>
</tr>
<tr>
<td>0.20</td>
<td>0.084</td>
</tr>
<tr>
<td>0.30</td>
<td>0.129</td>
</tr>
<tr>
<td>0.50</td>
<td>0.214</td>
</tr>
</tbody>
</table>

Test 1: CRB for continuous time sinc function $A=13$, $B=7$, $C=7$
Test 2: Simulation for asymmetric curve $A=13$, $B=4$, $C=7$
Test 3: Simulation for symmetric curve $A=13$, $B=7$, $C=7$
Test 4: Simulation using sinc function $A=13$, $B=7$, $C=7$

Expectations:
Test 1 and Test 4 should yield similar results.
Test 2 has a narrower peak than Test 3, so Test 2 should have smaller errors.

Table 3.1: Discrete Simulation Results
• $c(d)$ represents the ideal, continuous, noiseless comparison curve.

• The observed curve is $d(k) = c(k\Delta + \delta) + n(k)$.

• The noise $n(k)$ is Gaussian, variance $\sigma^2$, and independent of $d(k)$.

• $\Delta$ is the spacing along $d$ between successive samples.

• $\delta$ is the offset between the curves, which is the parameter to be estimated.

• $d(k)$ is defined for $N$ successive values of $k$; ie, we have $N$ samples.

• The signal energy of $c(t)$ is finite and is contained within the $N$ samples.

We form the sufficient statistic $m()$ and calculate its statistics:

$$m(q) = \sum_{i}^{N} (c(q + i\Delta) - d(i))^2 \quad q \in \mathcal{R} \quad (3.101)$$

$$E[m(\delta)] = E \left[ \sum_{i}^{N} (c(\delta + i\Delta) - d(i))^2 \right]$$

$$= E \left[ \sum_{i}^{N} n^2(i) \right] = N\sigma^2 \quad (3.102)$$

$$\text{var}(m(\delta)) = \text{var} \left( \sum_{i}^{N} n^2(i) \right) = 2N\sigma^4 \quad (3.103)$$

$$E[\hat{m}(\delta)] = E \left[ \sum_{i}^{N} 2(c(i\Delta + \delta) - d(i)) \hat{c}(i\Delta + \delta) \right]$$

$$= E \left[ \sum_{i}^{N} 2n(i) \hat{c}(i\Delta + \delta) \right] = 0 \quad (3.104)$$

$$E[\hat{\delta}(\delta)] = E \left[ \sum_{i}^{N} \left( 2\hat{c}^2(i\Delta + \delta) + 2\hat{c}(i\Delta + \delta)n(i) \right) \right]$$

$$= E \left[ \sum_{i}^{N} 2\hat{c}^2(i\Delta + \delta) \right] \quad (3.105)$$

By assumption we are sampling over the whole range in which $c(t)$ has non-zero energy, then

$$E \left[ \sum_{i}^{N} \hat{c}^2(i) \right] = \beta / \Delta \quad (3.106)$$

66
ie, the expectation value of the sum is proportional to the number of samples taken within the energy region ($\beta$ is some unknown constant).

So locally about the value of $\delta$, for low noise

$$m(d + \delta) \simeq E[m(\delta)] + d^2 E[\dot{m}(\delta)]/2$$  \hspace{1cm} (3.107)

A simple measure of estimator accuracy is the offset $\gamma$ required such that the ideal value of $m(d)$ (Equation 3.107) is one standard deviation (of its peak value) back from its ideal peak, ie,

$$\frac{\gamma^2 \beta}{2 \Delta} = \sqrt{2N\sigma^4}$$  \hspace{1cm} (3.108)

$$\gamma \propto \sigma \sqrt{\Delta \sqrt{N}}$$  \hspace{1cm} (3.109)

The above development makes three predictions:

1. The error increases linearly with the noise standard deviation.

2. The error increases as the square root of the sample step size.

3. The error is a weak function of $N$.

Furthermore, there are three limitations in applying the above derivation to our discrete time estimator:

1. For low noise (low $\sigma$) the step size of the estimator limits the RMS error of the estimate to Step Size/$\sqrt{12}$.

2. Equation 3.108 may be rewritten as $\gamma = 2\sigma \sqrt{2\Delta \sqrt{N}/\beta}$. For fine sample spacing (small $\Delta$), $N\Delta$ does not encompass all of the energy in $c(d)$ (see assumption above Equation 3.106). This causes $\beta$ to decrease, so $\gamma$ increases and estimation performance declines.

3. For large sample spacing (large $\Delta$), we may fail to sample any of the energy in $c(d)$ at all, in which case the estimator fails badly, decreasing performance.
Figure 3-16: Simulation Results for Sampling Test
Figure 3-16 shows the results of simulating our hybrid estimator (Section 3.4.2 as the sampling size is varied. The simulator is very much like program Sim_Sampling.C (Appendix B).

Figure 3-16a plots RMS error versus $\Delta$; each curve represents a contour of constant noise level $\sigma$. The plot exhibits a square root dependence for intermediate values of $\Delta$, and departure from theory is obvious at each extreme. Note that the lower limit on the RMS error varies with $\Delta$ (rather than being constant), since the estimation step within the maximum likelihood estimator was varied in proportion to $\Delta$.

Figure 3-16b plots RMS error versus $\sigma$; each curve represents a contour of constant sample spacing $\Delta$. The results are generally linear as predicted.
Chapter 4

System Calibration

The purpose of this chapter is to discuss a set of more practical issues associated with depth-from-focus apparatus. Some of these issues will be particular to our implementation, others are of interest in general.

Although the chapter is less interesting from a theoretical point of view, the author hopes that others may benefit from the time spent overcoming the problems listed here. Each case includes a brief discussion and a presentation of an algorithm if used (usually with a reference to the associated computer code in the appendices).

4.1 Projected Pattern

A set of patterns was prepared; each is a 35mm slide of round spots arranged on a regular lattice. All of the experiments in this thesis used a slide showing 29 spots horizontally by 24 spots vertically within the camera’s field of view. The interspot spacing is roughly 17 pixels; the spot diameter is roughly 7 pixels (about 40 microns on the pattern).

The slide is mounted in the optical path using a simple friction fit; such a mounting permits other slides to be easily inserted, but also gives the slide a little freedom to move.
4.1.1 Pattern Spot Location

The positions in the camera CCD array representing the center of each pattern spot location will generally not be known \textit{a priori}. Since it is desirable to know the spot locations precisely these coordinates need to be measured. Unless the apparatus is particularly rigid, the pattern may drift slowly over time, so periodic repetition of the calibration may be required. Since the calibration is run infrequently, the program is not optimized for speed. The following calibration may be inapplicable if a pattern very different than ours is used.

A successful spot location algorithm should meet the following requirements:

- Few spots on the pattern are missed,
- No spot must be counted twice,
- The center of each spot must be accurately identified.

The proposed algorithm (shown in greater detail in Figure 4-1) is as follows:

1. Obtain a rough estimate of spot separation in pixels (from the user). This value is used in determining the minimum permitted separation between candidate spot centers.

2. Apply to the image \( i(x, y) \) a two dimensional linear filter having impulse response \( \exp(-\gamma(x^2 + y^2)) \) where \( \gamma \) is chosen such that \( \exp(-\gamma r^2) = \frac{1}{2} \) where \( r \) is the expected radius of a spot. Let \( j(x, y) = i(x, y) * \exp(-\gamma(x^2 + y^2)) \).

3. Maintain a list of local maxima of \( j(x, y) \). For each new calculated value of \( j() \), search the list and eliminate any lesser values in its spatial vicinity.

4. After the entire image has been processed, the remaining entries in the list represent the spot centers.

The above algorithm is implemented in program CalSpot.C (see Appendix A). Figure 4-2 shows the results of a typical execution of this program: no spot was counted twice; a few spots around the screen periphery were missed. The identified
Start
SpotSep = Obtain rough estimate of spot separation from user
Initialize located spot list
Grab one image frame from camera
Loop from first to (last+SpotSep/2) line of camera frame
   Loop from first to last column of camera frame
      Call SpotTest
Loop from (last+SpotSep/2) to last line of camera frame
   Loop from first to last column of camera frame
      Call SpotTest
Sort spot list in order of ascending rows and columns
Output spot list to disk file
Done Exit Program

SpotTest
Apply two dimensional exponential filter centered at selected line and column
FiltVal = Filter result
KeepSpot = True
Loop over spot list
   If (list entry is within SpotSep/2 pixels of current location)
      If (list entry > FiltVal) then KeepSpot = False
      If (list entry < FiltVal) then remove current entry from list
      Go to next entry in list
   If (KeepSpot = True) then add current location and FiltVal to spot list
Done Return

Figure 4-1: Spot Location Calibration Algorithm
location of each spot is marked with a small “+” symbol in the figure. The algorithm has given consistently accurate results.

4.1.2 Pattern Spot Tracking

As the object being studied is moved forward and back in depth, if the optical elements of the system are not lined up along a common axis (among other problems) the observed spot positions may drift laterally with depth. In our case, the drift was sufficient to cause concern,\(^1\) so the position of the spots is recalibrated at each depth

\(^1\)That is, there was sufficient drift so that some spots hit the edge of the 16x16 pixel area allocated to them.
position. The problem posed here is simpler than spot location for several reasons:

- The spot location problem needed to determine the number of spots and reliably avoid duplication. Here the number is known, only the positions are updated.
- The approximate position of each spot is known, so a simple local search around the estimated position is sufficient.

Since the estimated spot positions are updated at each depth level, the program must be fairly fast.

Any one (or more) of a number of criteria might be used to find the spot center:

- Locate the brightest pixel within the given region:

\[(x_{center}, y_{center}) = \arg_{(x,y)} \max i(x,y)\]  \hfill (4.1)

- Locate the row (column) having the greatest average brightness.

\[x_{center} = \arg_x \max \sum_y i(x,y)\]  \hfill (4.2)

\[y_{center} = \arg_y \max \sum_x i(x,y)\]  \hfill (4.3)

- Locate the row (column) having the greatest brightness ratio with respect to nearby rows (columns); this searches for a local brightness peak:

\[x_{center} = \arg_x \max \frac{\sum_y i(x,y)}{\sum_y (i(x-\delta,y) + i(x+\delta,y))}\]  \hfill (4.4)

and similarly for \(y_{center}\). \(\delta\) is chosen to be on the order of one half of the spot radius.

For each of these criteria, if the spot was located (by CalSpot.C) at \((x_o, y_o)\), then the range for \(x\) and \(y\) is given by

\[x_o - \Delta \leq x \leq x_o + \Delta\]  \[y_o - \Delta \leq y \leq y_o + \Delta\]  \hfill (4.5)
where $\Delta$ is chosen empirically based on the degree of spot motion.\footnote{In any event, certainly Spot Radius $\leq \Delta \leq$ Spot Separation.}

The selection algorithm ultimately chosen (which demonstrated the most robust performance) was a hybrid of the last two methods, combining Equations 4.2 and 4.4:

$$x_{\text{center}} = \arg_x \max \frac{\left( \sum_{y} i(x, y) \right)^{\frac{1}{2}}}{\sum_{y} (i(x - \delta, y) + i(x + \delta, y))}$$

(4.6)

and similarly for $y_{\text{center}}$.

The source code for this algorithm is given under Find_Spot.C in Appendix ??.

This is not a stand-alone program, rather it is intended for use as a subroutine by other programs (for an example of use, see Cal_Curve.C in Appendix A).

### 4.1.3 Pattern Preparation

The patterns used in this thesis were prepared by an earlier student\cite{1}. A large version of the pattern was printed on paper using a laser printer, and captured on 35mm slide film. Although simple and quick, this method yields only moderately acceptable results. The graininess of commercial slide film is appreciable at our resolutions ($i 10\mu m$); significant differences in shapes and sizes between different spots have been observed.

It is recommended that other methods be investigated. If photographic film is to be used, care should be taken to obtain particularly fine grain film.

### 4.1.4 Pattern Positioning

For any object less than perfectly specular (ie, anything rougher than a mirror), the pattern and camera must share a coincident focal plane on the object. If these planes are offset in depth then the camera will never observe the pattern in sharp focus, effectively lowering the cutoff frequency of the in-focus spot (see Section 3.3) and degrading depth estimation accuracy. Fine adjustment of the focal planes may be made in one or both of the following ways:
• Place the pattern on a linear motion stage that permits precise adjustments.

• Use a focus control on the camera or its lens if available.\(^3\)

If the pattern stage positioning is under computer control, then the positioning may be automated. Since this was not the case for this thesis, the appropriate program was not written; instead the focus control was adjusted by hand.\(^4\) The basic algorithm, shown in Figure 4-3, is the same whether calibration is performed by hand or via computer.

### 4.2 Stage Motion

The object stage is a four axis system permitting motion along three orthogonal axes (X, Y, and Z) and rotation. Each axis is driven by a stepper motor connected to a micrometer. There are three basic types of errors that are encountered.

#### 4.2.1 Motion Jitter

Motion jitter appears as small random deviations from smooth stage travel. Such small offsets may be caused by slight mispositioning of the stepper motor\(^5\) and by dust or grit in the micrometer thread.\(^6\) Roughness in the stage bearings may contribute as well.

In any well built system, the amount of jitter should be considerably below the camera resolution, and so cannot be measured directly. If the jitter must be quantified, it should be measured independently using, for example, a laser range finder.

---

3 Focus control on the microscope lens attached to the camera was used in this thesis.

4 To some this may appear to be a risky prospect: adjusting focal plane positions by hand in a system striving for submicron accuracy. In actual fact, because the depth of focus is reasonably large (many microns), a small offset between the pattern and camera focal planes does not sacrifice depth estimation performance.

5 The stepper motor positioning is accurate to 0.05 degrees.

6 The micrometers are rated to 0.0001 inch deviation from linear over one inch of travel. A "jitter" specification was unavailable from the manufacturer.
Focal Plane Adjustment via Camera Focus
Start  Secure pattern in place
       Install object having significant surface detail
       Main light source shines through pattern onto object
       Auxiliary light source available for direct illumination without pattern
Repeat
       Turn main light source on, auxiliary light source off
       Loop over finely spaced object position along Z-axis
       Calculate degree of focus or blur measure (difference squared measure)
       Position object to depth for which blur value is maximized
       Turn main light source off, auxiliary light source on
       Loop over finely spaced focus ring positions, object does not move
       Calculate degree of focus as before
       Set focus ring to position maximizing blur value
Until optimal focus ring position is unchanged between iterations
Done  Exit Program

Focal Plane Adjustment via Pattern Position
Start  Secure pattern in place
       Install object having significant surface detail
       Main light source shines through pattern onto object
       Auxiliary light source available for direct illumination without pattern
Repeat
       Turn main light source off, auxiliary light source on
       Loop over finely spaced object position along Z-axis
       Calculate degree of focus or blur measure (difference squared measure)
       Position object to depth for which blur value is maximized
       Turn main light source on, auxiliary light source off
       Loop over finely spaced pattern positions, object does not move
       Calculate degree of focus as before
       Set pattern to position maximizing blur value
Until optimal pattern position is unchanged between iterations
Done  Exit Program

Figure 4-3: Camera and Pattern Focal Plane Calibration Algorithm
4.2.2 Periodic Aberration

A periodic aberration is some deviation from perfectly linear motion, typically periodic over one complete turn of the stepper motor / micrometer system.

The weak link in the motion control system is the point of contact between the rotating face at the end of the micrometer and the stationary ball bearing. If the micrometer face is not completely flat, or the face not perpendicular to the axis of micrometer rotation a periodic nonlinear motion may be induced.

The problem may be identified using program Cal_Screen.C (see Section 4.3.1 below); however Cal_Screen.C only calibrates over the field of view of the camera, whereas (for us) a complete micrometer turn is somewhat larger. Since this kind of test is performed very infrequently, a tedious (but reliable) manual calibration approach is used rather than a fully automated program.

For the X and Y axes, motion may be observed directly on the screen. Algorithm 1 in Figure 4-5 describes the procedure. For the Z axis, the linearity of the micrometer must be inferred from a constant rate of travel of the line of focus as a tilted object is moved towards the camera (see Figure 4-4) via Algorithm 2 in Figure 4-5.

Each of the three linear axes were tested (the rotational axis is never used by us and was not tested). The Y and Z axes performed admirably; the X axis demonstrates a significant periodicity as shown in Figure 4-6.
Algorithm 1
Start Place Some Object on Stage and Focus
    Do
        Grab Image Frame
        Identify Some Feature on Object (Spot, Crack, etc)
        Move Object Some Preset Distance (Here set to 12.5 micrometers)
        Grab Image Frame
        Measure Offset in Pixels of Identified Feature Between Two Image Frames
    Until Micrometer Turned Two Complete Rotations
    Save Pixel Offsets to Disk File
    Analyze Offsets, Look for Periodicity
Done Exit Program

Algorithm 2
Start Place Flat Object on Stage and Focus
    Object is Assumed Tilted to X-Axis and Parallel to Y-Axis
    Grab Series of Frames and Perform Depth Estimation
    Average Estimates Along Y-Axis Giving Average Depth vs. X Position
    Save Averaged Depth Estimates to Disk File
    Analyze Offsets, Look for Periodicity
Done Exit Program

Figure 4-5: Micrometer Linearity Assessment Algorithm
Figure 4-6: Residual Periodic Motion of X-Axis Micrometer
<table>
<thead>
<tr>
<th>Axis</th>
<th>Backlash</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>$\sim 5.0\mu m$</td>
<td>Significant Periodic Motion</td>
</tr>
<tr>
<td>Y</td>
<td>$\sim 13\mu m$</td>
<td>Large Backlash, Small Periodicity</td>
</tr>
<tr>
<td>Z</td>
<td>$&lt; 2.5\mu m$</td>
<td>Small Backlash, Small Periodicity</td>
</tr>
</tbody>
</table>

Table 4.1: Micrometer Backlash Measurements

4.2.3 Micrometer Backlash

Because the micrometer thread and screw do not make a perfect fit there is a small amount of play (freedom for the micrometer threads to move with no change in screw position).

Once the micrometer has been moved several steps in one direction, the threads and the screw make proper contact and motion is clean. When the direction of motion is reversed, the first few steps of the stepper motor may yield limited or no motion from the micrometer. This effect is known as backlash. Its magnitude is reduced by spring loading the micrometer (spring loading is used on our stage for each axis), however backlash effects still manifest themselves.

The backlash consideration is not new; it was studied and quantified by Delisle[1]. The amount of backlash actually observed in a given situation varies considerably, depending on what sort of motions have just been made by the micrometer. The amount of backlash reported in [1] is shown in Table 4.1.

All tests in this thesis remove backlash before making measurements by moving the micrometer past the desired position and returning. For example:

To move to position $Z=-30$ and proceed in the positive direction without backlash:

1. Move to position $Z=-40$
2. Insert small time delay
4.3 System Optics

4.3.1 Screen Deformation

Although a wide range of image deformations is possible, broadly speaking a camera system may be classified as pincushion, pillowcase, or not deformed (see Figure 4-7).

The form of the deformation is of interest only when depth estimates, associated with a particular region of the camera image, are to be translated to the corresponding point on the object. Although such data translation is of minimal interest within this thesis, such a calibration program was developed (see program Cal_Screen.C in Appendix A). The program algorithm is shown in Figure 4-8. Since the algorithm assumes that constant steps of the stepper motor result in constant shifts of the object, the micrometers must exhibit only minimal periodic variation.

A typical output plot of Cal_Screen.C is shown in Figure 4-9. Each ‘x’ in the figure is separated from its neighbors by an equal turn of the micrometer (here 5 motor steps, nominally 12.5 microns); the change in the spacing between the left and right sides of the figure is due to the periodic nonlinearity in the X-Axis micrometer, it is not a distortion in the optics or the CCD camera. The arrangement of ‘x’ symbols is quite rectangular even in the corner of the image, suggesting that the amount of image distortion due to the camera is small. The partial column of symbols in the upper right of the figure indicates that the camera is slightly rotated from the ideal (i.e., the x-axis along the CCD array is not quite parallel to the x-axis of the stage).
Start
Place Object on Frame
Focus Camera onto Object
Grab Frame and Display on Computer Monitor
Have User Identify Some Small Surface Feature to be Tracked
Repeat
  Move Small Amounts Along X and Y Axes
  Grab Frame and Locate New Position of Surface Feature
  Update Estimates of Pixel Width and Height in Microns
Until Pixel Width and Height Estimates Accurate
Move Surface Feature to Upper Left Corner of Screen
Loop over Y Axis Positions (User Defined Step)
  Move Object Downwards
  Remove Backlash
  Loop over X Axis Positions (User Defined Step)
    Move Object to the Right
    Grab Frame and Locate Surface Feature
    Save Identified Location
Restore Initial Object Position
Plot Map of Identified Screen Coordinates and Save to Disk
Done
Exit Program

Figure 4-8: Camera Image Deformation Assessment Algorithm
Figure 4-9: Output of Screen Shape Calibration Program
4.3.2 Curvature of Focus

The focal plane of the camera and microscope objective system is not flat. As a result, when observing a flat object, not all parts of the object will be in focus at any given time. Curvature of focus correction is desired to improve depth estimation accuracy.

As a first order approximation, the focal plane of the objective is a spherical shell of constant radius\cite{11} (with the sphere centered on the midpoint of the lens); see Figure 4-10. The approximation is best near the optical axis.

For our system, $r_{lo} \simeq 7\,mm$. The field of view of the object is about $400\,\mu m$ wide, so $d_{scrn} \simeq 200\,\mu m$. Then

$$d_{ofs} = r_{lo}(1 - \cos^{-1}\left(\frac{d_{scrn}}{r_{lo}}\right))$$  \hspace{1cm} (4.7)

which evaluates to about $3\,\mu m$.

The correction for this effect may be made in one of two ways:

1. Using a pattern, carefully evaluate the depth for each pattern spot when observing a flat mirror. Calculate the second order regression parameters\footnote{ie, The regression coefficients of $x$, $y$, $xy$, $x^2$, and $y^2$.}. The two linear coefficients are discarded, since the test object is not guaranteed to be mounted perfectly perpendicularly to the optical axis. The three higher order coefficients characterize the curvature and are saved for future use.
Table 4.2: Curvature of Focus Regression Parameters (units of \( \mu m/\text{pixel}^2 \))

<table>
<thead>
<tr>
<th>Lens Angle</th>
<th>( x^2 )</th>
<th>( y^2 )</th>
<th>( xy )</th>
</tr>
</thead>
<tbody>
<tr>
<td>45(^\circ)</td>
<td>(21.8 \cdot 10^{-6})</td>
<td>(21.3 \cdot 10^{-6})</td>
<td>(-16.9 \cdot 10^{-8})</td>
</tr>
<tr>
<td>90(^\circ)</td>
<td>(25.8 \cdot 10^{-6})</td>
<td>(17.5 \cdot 10^{-6})</td>
<td>(-18.5 \cdot 10^{-6})</td>
</tr>
<tr>
<td>135(^\circ)</td>
<td>(30.8 \cdot 10^{-6})</td>
<td>(14.4 \cdot 10^{-6})</td>
<td>(-22.8 \cdot 10^{-6})</td>
</tr>
<tr>
<td>180(^\circ)</td>
<td>(24.0 \cdot 10^{-6})</td>
<td>(6.4 \cdot 10^{-6})</td>
<td>(-17.0 \cdot 10^{-6})</td>
</tr>
</tbody>
</table>

2. Using a pattern, carefully evaluate the depth for each pattern spot when observing a flat mirror. Calculate the first order regression parameters and remove the first order component from the depth estimates. Save the depth estimate for each spot; these values are used later to calibrate each spot individually.

Both of the above calibration forms are supported by program Cal_Curve.C\(^9\) located in Appendix A.

Most test results in this thesis calibrate each spot individually for accuracy. However to test the appropriateness of the assumed spherical curvature of focus (Equation 4.7) a number of tests to calibrate the second order regression parameters were performed. A subset of the results is shown in Table 4.2. The regression results differ for the X and Y axes, suggesting that the objective lens is mounted slightly off the camera’s optical axis.\(^{10}\) Using typical regression values from this table predicts an offset in the focal plane at the edge of the camera screen of between 2\( \mu m \) and 4\( \mu m \) which is approximately that expected for our system.

### 4.3.3 Light Source

For most of the work of this thesis the light source\(^{11}\) is not a major concern. Generally, if the region of interest on an object is sufficiently well lit, then the light source is considered to be adequately fulfilling its role and is forgotten. This dismissal is inappropriate in two cases:

\(^9\)In fact this program serves a number of other purposes, including determining empirical calibration curves and assessing the variation in blur measure curves due to spot differences.

\(^{10}\)Simple trigonometric calculations show that the regression parameters should each vary sinusiodally as the lens is rotated. Some of this variation is apparent in Table 4.2.

\(^{11}\)We used incoherent, incandescent light. See Section 2.2 for details.
• We wish to infer information from the observed brightness at each point on the object (the surface angle, roughness, or reflectivity for example). In this case, the brightness of the light source must be observed and stored over the entire image. This type of calibration was not of interest in this thesis and was not performed.

• In some instances the shape of the light source becomes important. In particular, in our experiment a small (effectively point sized) light source accentuated the variations in the spots on the pattern (see Section 4.1.3); a spread light source\textsuperscript{12} tended to make the observed spots more even. Although this improvement may not be true in general, it suggests that a variety of light sources should be tried.

4.4 Camera Frame Grabbing

The CCD camera used in this experiment is a Panasonic WV-BL200 having a resolution of 512H by 480V pixels. It is connected to a Data Translation DT2851 real-time frame grabber card.

4.4.1 Interference Signals

The DT2851 analog to digital converter suffers an acute interference problem. The frequency responses of sampled images (see Figure 4-11) all contain a spike at 3.58 MHz, the standard color reference frequency. The peak-to-peak amplitude of the interference in the spatial domain is just in excess of one CCD quantization level\textsuperscript{13}. The source of the interference is unknown: the color frequency is present on the computer backplane, in the nearby color video controller, and in the monitor next to the computer. The interference is deterministic (constant amplitude and phase from one frame to the next), so digital post-filtering is an option; we chose to enable the

\textsuperscript{12}A "soft" (ie, translucent shell) 100W light bulb performed admirably.

\textsuperscript{13}One CCD quantization level is the separation between two successive pixel values.
Figure 4-11: Color Frequency Interference Example Plot
chrominance filter on the DT2851 board instead.

The implications of such an interference are as follows:

- The interference is limited to a few specific high frequency FFT coefficients; low frequency coefficients are unaffected. The FFT blur measure is a function of low frequency coefficients only, so it is insensitive to the interference.

- The difference-squared blur measure is strongly affected by the interference since high frequency components are weighted heavily. The reduction in signal-to-noise ratio is devastating.

- The removal of the interference signal removes the perceived correlation between pixel noise values (as cited in [1]) and clearly reduces the pixel noise level (see next section).

4.4.2 Camera CCD Noise

Throughout this thesis the noise associated with each pixel was stated to be white and independent of the noise on other pixels.

Program Cal_Pixel.C in Appendix A observes a set of image frames and computes noise statistics. The simplest approach to measuring pixel noise levels is to assume uniform illumination of the whole screen, and to calculate statistics based on a single image. Since it is questionable whether our light source provides such even illumination\textsuperscript{14} (see Section 4.3.3) an improved algorithm, presented in Figure 4-12, makes no such assumption.

The summarized results of the camera tests are shown in Table 4.3; the white noise and independence assumptions for pixel noise are justified. The use of an automatic gain control (AGC) tends to increase the pixel noise variance, however it also spreads pixel values and thus reduces the error due to quantization noise. The use of the AGC is recommended.

\textsuperscript{14}To be blunt, it is obvious that the light source does not provide such evenness.
Start  
Clear Statistics

Repeat
  Prompt User to Position Light Source in Desired Manner
  Loop Over Image Lines
    Loop Over Multiple Frames
      Grab Image Frame from Camera
      Extract and Save Desired Line
    Determine Mean and Variance of each Pixel on Line
    For each Pixel, add Variance to Statistic Corresponding to Pixel Mean
  Until User Has Performed a Sufficient Variety of Tests
Output Averaged Variance as a Function of Pixel Mean

Done  Exit Program

Figure 4-12: Camera Noise Measurement Algorithm

<table>
<thead>
<tr>
<th>Automatic Gain</th>
<th>Experiment</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Dark view</td>
<td>All pixels at zero intensity, zero variance</td>
</tr>
<tr>
<td>On</td>
<td>Dark view</td>
<td>Pixel values range from zero to five, typical pixel variance around 2.0</td>
</tr>
<tr>
<td>Off</td>
<td>Mixed view</td>
<td>$0.65 &lt; \sigma &lt; 1.0$, Variance is essentially independent of pixel value</td>
</tr>
<tr>
<td>On</td>
<td>Mixed view</td>
<td>$0.95 &lt; \sigma &lt; 1.6$. Approximate dependence is $\sigma \simeq 1.0 + \text{PixVal}/500$. The dependence is weak and, as a simple approximation, is ignored.</td>
</tr>
</tbody>
</table>

Table 4.3: CCD Camera Noise Calibration Results
Chapter 5

Experimental Results

This chapter discusses the experimental results obtained by applying the blur measures and estimator of Chapter 3 and the calibrations of Chapter 4 to experimental data from a variety of materials.

5.1 Ideal Specular Surface - Blur Measure Noise

In Section 3.5.2 equations were developed to predict the effect of camera pixel noise on the noise in the blur measure. Having measured the camera pixel noise in Section 4.4.2 we are in a position to try to predict the blur measure noise and compare with experiment. The following two subsections address the FFT and difference-squared blur measures respectively.

5.1.1 FFT Measure

From Section 3.4.1 the FFT blur measure was given as

$$\text{FFTMeas} = |D(1,0)|^2 + |D(0,1)|^2 + |D(1,1)|^2 + |D(1, -1)|^2$$  \hspace{1cm} (5.1)
<table>
<thead>
<tr>
<th>Blur Value</th>
<th>Theoretical Noise $\sigma$</th>
<th>Experimental Noise $\sigma$</th>
<th>Signal / Exper. Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.26</td>
<td>0.70</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>1.16</td>
<td>1.17</td>
<td>4.3</td>
</tr>
<tr>
<td>10</td>
<td>1.64</td>
<td>1.73</td>
<td>5.8</td>
</tr>
<tr>
<td>30</td>
<td>2.85</td>
<td>2.90</td>
<td>10.3</td>
</tr>
<tr>
<td>100</td>
<td>5.20</td>
<td>5.52</td>
<td>18.1</td>
</tr>
<tr>
<td>300</td>
<td>9.01</td>
<td>10.2</td>
<td>29.4</td>
</tr>
<tr>
<td>1000</td>
<td>16.4</td>
<td>18.4</td>
<td>54.3</td>
</tr>
<tr>
<td>2000</td>
<td>23.2</td>
<td>24.0</td>
<td>83</td>
</tr>
<tr>
<td>3000</td>
<td>28.5</td>
<td>32.5</td>
<td>92</td>
</tr>
<tr>
<td>5000</td>
<td>37.8</td>
<td>44</td>
<td>114</td>
</tr>
<tr>
<td>7000</td>
<td>43.5</td>
<td>57</td>
<td>123</td>
</tr>
<tr>
<td>10000</td>
<td>52</td>
<td>67</td>
<td>149</td>
</tr>
</tbody>
</table>

Experimental data in the last two rows is weak due to limited number of data points.

Table 5.1: FFT Blur Measure Noise Comparison

where $d(x, y)$ represents the image obtained from the camera. Then, from the equation for the blur measure variance (Equation 3.96):

$$\text{var(FFTMeas)} = 4\sigma_{fft}^2 \text{FFTMeas} + 8\sigma_{fft}^4$$

$$\sigma_{fft} = 16\sigma_{\text{pixel}} = 16\sigma_{\text{ccd}} / \kappa$$

The correction factor of 16 is due to the fact that the FFT eigenfunctions are not normalized; the scaling by constant $\kappa$ is due to image normalization (see Section 3.4.1). For a typical, brightly illuminated, screen $\kappa$ averages to approximately 80.

The blur measure variance is computed in practice by grabbing a series of images from the camera (with no stage motions between grabbing the images), applying the FFT blur measure to each of these, and computing the second order statistics. Table 5.1 compares the experimental and theoretical noise levels. The results agree to the accuracy with which the camera pixel noise is known, confirming the camera noise measurements and the correctness of the derivation in Chapter 3.

Typical peak values for the FFT blur measure (ie, the blur measure value for the in-focus position) are between 8000 and 10000, giving a peak signal-to-noise ratio of about 110 to 130.
5.1.2 Difference-Squared Measure

Our discussion for the difference-squared measure proceeds similarly to the FFT case. From Section 3.4.1 the difference-squared blur measure was given as

$$\text{DiffSq} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-2} \left( (d(i, j) - d(i, j + 1))^2 + (d(j, i) - d(j + 1, i))^2 \right)$$  \hspace{1cm} (5.4)

where $d(x, y)$ is the image obtained from the camera.

Then from Section 3.5.2,

$$\text{var(DiffSq)} = \sum_{i=0}^{N} \sum_{j=0}^{N} \alpha(k_x, k_y)^2 \left( 4|D(k_x, k_y)|^2 \sigma^2 + 2\sigma^4 \right)$$  \hspace{1cm} (5.5)

$$\approx \sum_{i=0}^{N} \sum_{j=0}^{N} \alpha(k_x, k_y) \left( 4\text{DiffSq} \sigma^2 + 2\sigma^4 \alpha(k_x, k_y) \right)$$  \hspace{1cm} (5.6)

Clearly Equation 5.6 is an approximation of Equation 5.5. The approximation is met with equality when $\alpha(k_x, k_y)|D(k_x, k_y)|^2$ is constant (ie, independent of $k_x$ or $k_y$).

In actual fact, however, the camera images are quite low pass in nature, so we expect Equation 5.6 to exaggerate the noise present in the blur measure. A quick test evaluated the following ratio (Equation 5.6 / Equation 5.5) over a variety of spots and found that

$$\frac{\frac{1}{N} \sum_{i=0}^{N} \sum_{j=0}^{N} \alpha(k_x, k_y) \cdot \sum_{i=0}^{N} \sum_{j=0}^{N} \alpha(k_x, k_y) |D(k_x, k_y)|^2}{\sum_{i=0}^{N} \sum_{j=0}^{N} \alpha(k_x, k_y)^2 |D(k_x, k_y)|^2} \approx 2.8$$  \hspace{1cm} (5.7)

This correction factor is included in the theoretical noise calculation.

Table 5.2 compares the predictions of Equation 5.6 with experimental results. The results agree fairly well, although some departure from theory is expected, given the approximations made (for example Equation 5.7).

Typical peak values for the difference-squared blur measure are between 15 and 19, giving a peak signal-to-noise ratio of about 85 to 100 (slightly lower than that for the FFT).
<table>
<thead>
<tr>
<th>Blur Value</th>
<th>Theoretical Noise σ</th>
<th>Experimental Noise σ</th>
<th>Signal / Exper. Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.019</td>
<td>0.032</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.039</td>
<td>0.055</td>
<td>18.0</td>
</tr>
<tr>
<td>2</td>
<td>0.055</td>
<td>0.065</td>
<td>30.6</td>
</tr>
<tr>
<td>4</td>
<td>0.077</td>
<td>0.081</td>
<td>49.4</td>
</tr>
<tr>
<td>6</td>
<td>0.095</td>
<td>0.099</td>
<td>60.8</td>
</tr>
<tr>
<td>8</td>
<td>0.110</td>
<td>0.111</td>
<td>72.1</td>
</tr>
<tr>
<td>10</td>
<td>0.123</td>
<td>0.119</td>
<td>84.2</td>
</tr>
<tr>
<td>12</td>
<td>0.134</td>
<td>0.129</td>
<td>92.7</td>
</tr>
<tr>
<td>14</td>
<td>0.145</td>
<td>0.148</td>
<td>94.9</td>
</tr>
<tr>
<td>16</td>
<td>0.155</td>
<td>0.168</td>
<td>95.0</td>
</tr>
<tr>
<td>18</td>
<td>0.164</td>
<td>0.190</td>
<td>94.7</td>
</tr>
</tbody>
</table>

The last two rows of experimental data are statistically weak due to few data points.

Table 5.2: Difference-Squared Blur Measure Noise Comparison

5.1.3 Calibration Curve Noise

The standard implementation of the depth estimator (see Section 3.4.2) assumes that an ideal calibration curve has been determined. Typically a separate curve is determined experimentally for each surface being studied or blur measure being tested. In actual fact the calibration curve changes relatively little for different blur measures or materials, so its determination should need to be performed only once (preferably using a mirror if a variety of materials are to be observed, otherwise using the actual material of interest).

Program CalCurve.C (Appendix A) can determine a calibration curve. It does so by finding a best fit curve for an observed ensemble of blur measure curves (one curve for each spot on the pattern). Let \( c_i(d) \) represent the ideal calibration curve for spot \( i \). The pattern spots are not all uniform (see Section 4.1.3), so clearly there will be variations among the curves. The overall calibration curve \( c_{total}(d) \) is given by

\[
c_{total}(d) = \frac{1}{M} \sum_i^M c_i(d)
\]

\[
rmse(c_{total}(d)) = \sqrt{\frac{1}{M} \sum_i^M c_i^2(d) - c_{total}^2(d)}
\]

In actual fact the spot curves \( c_i(d) \) may also be offset from one another in depth. This offset is estimated and accounted for in CalCurve.C, but is omitted from this discussion for simplicity.
<table>
<thead>
<tr>
<th>Noise Sources:</th>
<th>Focus Curvature</th>
<th>Camera Noise</th>
<th>Pattern Spots</th>
<th>Object Flatness</th>
<th>Stage Motions</th>
<th>Section Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment A</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>5.2.1</td>
</tr>
<tr>
<td>Experiment B</td>
<td></td>
<td>Some</td>
<td>X</td>
<td>X</td>
<td></td>
<td>5.2.2 5.2.3</td>
</tr>
<tr>
<td>Experiment C</td>
<td></td>
<td></td>
<td>X</td>
<td>?</td>
<td></td>
<td>5.2.3 5.2.4</td>
</tr>
<tr>
<td>Experiment D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.2.4 5.2.5</td>
</tr>
<tr>
<td>Experiment E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.2.5</td>
</tr>
</tbody>
</table>

Table 5.3: List of Experiments and Applicable Noise Sources

where there are \( M \) spots seen by the camera. The rms deviation (Equation 5.9) measures the degree to which the overall calibration curve fails to accurately represent the ensemble. This deviation is also calculated by Cal_Curve.C; standard calibration curve and rms deviations are plotted for the two standard blur measures in Figure 5-1. These results are applied in Section 5.2.3.

5.2 Ideal Specular Surface – Detailed Noise Breakdown

In this section we perform a series of depth measurement experiments on an ideal specular surface – a mirror. Each subsection that follows addresses and tries to quantify the effect of one of the noise sources in the system, at which point a more advanced test is developed which is insensitive to this noise.

Table 5.3 lists the noise sources to be discussed. Each row in the table refers to one experiment, and lists the noise sources affecting that experiment. The right hand column gives the section reference where an experiment is discussed; only a subset of the noise sources will be discussed in any particular section.

Most of the experiments in this chapter operate as follows: perform two complete depth passes and save the observed blur value curves \( b_{i_1}(d) \) and \( b_{i_2}(d) \) for spot \( i \). Apply each of these to the estimator to yield depth estimates \( D_{i_1} \) and \( D_{i_2} \). The depth offset or error is given by

\[
\delta_i = D_{i_1} - D_{i_2}
\]  

(5.10)
Figure 5-1: Calibration Curves and RMS Deviation
and the depth offset standard deviation is

\[
\text{Error} = \sqrt{\frac{1}{M} \sum_{i=1}^{M} \delta_i^2 - \left( \sum_{i=1}^{M} \delta_i \right)^2}
\]  \hspace{1cm} (5.11)

There are several important points that should be kept in mind to keep the results quoted in this chapter in proper context.

- The distinction between repeatability and actual accuracy must remain clear in the reader's mind. Since the surface profile of our test objects is not known \textit{a priori}, to avoid including surface roughness in the reported error level, some tests of this chapter report repeatability error, rather than absolute measurement error (which is unknown). In any event, measurement repeatability always sets the lower bound for measurement accuracy.

- All of the accuracy values quoted in this chapter are based on the standard deviation of depth estimates; the average value of the estimates is not reported. For conciseness, throughout this chapter we refer to test results as RMS errors; this notation is legitimate only with the understanding that the bias (average) values have been set to zero.

- \( \text{var}(\delta_i) = 2\text{var}(D_i, \text{ or } z) \), so the error value in Equation 5.11 is exaggerated by a factor of \( \sqrt{2} \). The results in this chapter show the experimental value \textit{divided} by \( \sqrt{2} \), since this should represent the true level of accuracy if an ideal calibration curve were used (achievable in practice by extensive averaging).

### 5.2.1 Optical Offsets

Section 4.3.2 discussed curvature of focus for a thin lens system. Second order regression parameters were calculated which quantify the degree of curvature.

Curvature of focus theory assumes a thin lens which is centered on the optical axis; this theory may not necessarily apply in all cases (e.g., lens is too thick, offset from optical axis, rotated with respect to optical axis etc.). Furthermore, there are
other possible causes for offsets in depth: the film containing the pattern may not be flat, other optical components (camera and lens system) may have unknown optical properties etc.

Experiment A (see Table 5.3):
Perform two full passes in depth\(^2\) (Z axis), shift the object along the X axis between the depth passes. Average over 5 frames to reduce pixel noise.

**Test 1:** Apply first order regression corrections: this removes a possible slope in the depth estimates possibly due to the test mirror not being mounted precisely perpendicular to the optical axis.
RMS Depth Estimate: 1.72\(\mu m\)

**Test 2:** Apply second order regression corrections: this removes a possible slope as before, and removes a best fit paraboloid which approximates a sphere near the optical axis (the deviation in depth predicted by thin lens curvature of focus[7]).
RMS Depth Estimate: 1.13\(\mu m\)

Using second order regression in Test 2 reduces the estimation error by 0.6\(\mu m\) to 1.13\(\mu m\). In the optimal case, where the depth estimation map contains zero intrinsic estimation error, the entire remaining 1.13\(\mu m\) may be attributed to the inability of a paraboloid to match the curvature of focus. In actual fact this optimal case is not realized since there is estimation noise present. This noise is somewhat less than one micron, however, so a very real improvement may be realized in correcting each spot individually.

Conclusion: to eliminate optical offset noise, future tests will always determine a calibrated depth offset \(d_i\) for each spot \(i\), to be subtracted from the depth estimate to yield the actual depth to the object.

\(^2\)That is, proceed along the Z axis from one side of focus to the other, grabbing frames at regular depth intervals. Each image frame is passed through a blur measure and the estimator. The estimated depth values are saved for further analysis.
5.2.2 Camera Noise

Section 3.5.2 discussed the effects of pixel noise on the blur measures and the impact of blur measure noise on the final estimation process. Section 5.1 obtained experimental estimates of the level of white noise in the blur measures.

Although the signal-to-noise ratio $\sqrt{V}/P$ is better for the FFT blur measure than for difference-squared (Section 5.1) the FFT measure suffers three problems:

- The largest slope of the blur measure curve (just to the right of the focus position) is about 40% lower for the FFT measure than the difference-squared case (or, equivalently, the main lobe width $M$ is somewhat wider for the FFT measure).

- The FFT measure has a greater sensitivity to spot differences (discussed in Section 5.2.3 below).

- The FFT measure is numerically intensive and thus very time consuming to use in long tests.

A few tests showed the FFT measure to have inferior performance with respect to the difference-squared measure; therefore the FFT measure will not be used until other test objects are considered in Section 5.4.

From Section 3.5.2, for averaging $N$ frames of camera data,

$$\text{RMS Error} = \frac{\sqrt{3V}M}{\pi P \sqrt{N}}$$  \hspace{1cm} (5.12)

Using the difference-squared blur measure, based on results from Section 5.1, $P = 16$, $V = 0.17^2$, $M = 20 \mu m$,

$$\text{RMS Error} = 0.12 \mu m/\sqrt{N}$$  \hspace{1cm} (5.13)

Experiment B (see Table 5.3):
Perform sets of two full passes each in depth over the object, shift the object along the X axis between the depth passes. The number of averaged frames $N$ is held constant throughout each set and is doubled between sets.
<table>
<thead>
<tr>
<th># Frames $N$</th>
<th>Theoretical Error</th>
<th>Experimental Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.120 $\mu m$</td>
<td>0.184 $\mu m$</td>
</tr>
<tr>
<td>2</td>
<td>0.085 $\mu m$</td>
<td>0.137 $\mu m$</td>
</tr>
<tr>
<td>4</td>
<td>0.060 $\mu m$</td>
<td>0.124 $\mu m$</td>
</tr>
<tr>
<td>8</td>
<td>0.042 $\mu m$</td>
<td>0.127 $\mu m$</td>
</tr>
<tr>
<td>16</td>
<td>0.030 $\mu m$</td>
<td>0.127 $\mu m$</td>
</tr>
</tbody>
</table>

Table 5.4: Test of CCD Pixel Noise Effects

Experimental results and theory comparison are shown in Table 5.4:

- The predicted level of error based solely on the effects of pixel noise are consistently lower than experimental values, suggesting that significant additional sources of noise are present.

- For $N = 1, 2, 4$ the decrease in error is similar for theory and experiment. This suggests that the white noise component of the error is being modeled fairly accurately, and that for $N \geq 4$ other sources of noise dominate.

### 5.2.3 Pattern Spot Differences

The results of Section 5.1.3 quantify the degree to which the overall calibration curve fails to match the individual curves for each spot.

We wish to simulate the deviations from the calibration curve and to obtain a rough quantitative appreciation for the error in estimating depth that results from these:

- The deviation $c_{total}(d) - c_i(d)$ is a smoothly varying function, i.e., $c_{total}(d) - c_i(d)$ and $c_{total}(d + 1) - c_i(d + 1)$ are strongly correlated.

- Let $\Delta(d)$ be some deviation curve. Let $\hat{d}$ and $\bar{d}$ be the estimated depths using $c(d)$ and $c(d) + \Delta(d)$ as calibration curves respectively. In general,

$$E[\hat{d}] \neq E[\bar{d}]$$  \hspace{1cm} (5.14)
<table>
<thead>
<tr>
<th>Camera Noise $\sigma$</th>
<th>Deviation Amplitude $a \sim N(0,0)$</th>
<th>$a \sim N(0,1)$</th>
<th>$a \sim N(0,2)$</th>
<th>$a \sim N(0,4)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.088</td>
<td>0.156</td>
<td>0.230</td>
<td>0.290</td>
</tr>
<tr>
<td>0.05</td>
<td>0.090</td>
<td>0.158</td>
<td>0.205</td>
<td>0.260</td>
</tr>
<tr>
<td>0.10</td>
<td>0.102</td>
<td>0.168</td>
<td>0.218</td>
<td>0.273</td>
</tr>
<tr>
<td>0.15</td>
<td>0.118</td>
<td>0.208</td>
<td>0.260</td>
<td>0.330</td>
</tr>
<tr>
<td>0.20</td>
<td>0.155</td>
<td>0.220</td>
<td>0.253</td>
<td>0.300</td>
</tr>
<tr>
<td>0.25</td>
<td>0.178</td>
<td>0.235</td>
<td>0.278</td>
<td>0.340</td>
</tr>
<tr>
<td>0.30</td>
<td>0.218</td>
<td>0.263</td>
<td>0.295</td>
<td>0.343</td>
</tr>
<tr>
<td>0.35</td>
<td>0.263</td>
<td>0.300</td>
<td>0.330</td>
<td>0.368</td>
</tr>
<tr>
<td>0.40</td>
<td>0.308</td>
<td>0.348</td>
<td>0.378</td>
<td>0.415</td>
</tr>
<tr>
<td>0.45</td>
<td>0.355</td>
<td>0.378</td>
<td>0.393</td>
<td>0.430</td>
</tr>
<tr>
<td>0.50</td>
<td>0.370</td>
<td>0.408</td>
<td>0.423</td>
<td>0.448</td>
</tr>
</tbody>
</table>

Table 5.5: Error due to Differences in Pattern Spots

however this offset is corrected as part of calibrating the curvature of focus (Section 5.2.1); it is the increase in the variance

$$\text{var} (\hat{d}) - \text{var} (\bar{d})$$

(5.15)

that interests us.

Based on experimental observations, for simulation purposes the deviation is modeled as follows:

$$\Delta(\bar{d}) = \text{rms}(c_{\text{total}}(d)) \cdot a \sin(d/T - \delta)$$

(5.16)

$$a \sim N(0,1) \quad \delta \sim U(-\pi, \pi) \quad T \sim U(M, 3M)$$

(5.17)

where $N()$ represents a Gaussian distribution, and $U()$ a uniform distribution. $M$ is the width of the main lobe of $c_{\text{total}}(d)$.

Program Sim_Spot.C (see Appendix B) applies an ensemble of the above deviations to a standard ideal calibration curve (based on Figure 3-15) for a variety of white noise levels. The simulation results are shown in Table 5.5.

We make the following observations:

- The input parameters to the simulation program are very approximate, and the
form of the deviations themselves somewhat ad-hoc. Therefore the simulation results should be used to enrich our intuition and to provide a rough quantitative error level, but should by no means be considered precise.

- The simulation results in the table are based on a finite number of tests, so the results are subject to an error variance. This explains why, in some cases, a test with a greater $\sigma$ yielded a lower estimation variance than one having a smaller $\sigma$.

- The effect of deviation $\Delta(d)$ is felt most strongly for small values of $\sigma$; at large values of $\sigma$ the estimation errors are due predominantly to white noise.

- In the following experiments, multiple camera frames are averaged prior to the application of the blur measure, so the approximate value for the noise is $\sigma < 0.1$.

Experiment C (see Table 5.3):
Perform sets of two full passes each in depth over the object:
1. Full depth pass (average 5 frames), save blur measure curves $\rightarrow c_i(d)$
2. Shift object along the X axis.
3. Another full depth pass (average 5 frames), save blur measure curves $\rightarrow c_i(d)$
4. Estimate offset between $c_i(d)$ and $c_i(d)$. 
This offset is the depth estimate for spot $i$. In effect, each spot is given its own calibration curve, which eliminates the need for one generic curve to represent all spots.

The above test yielded a RMS depth estimation of 0.07$\mu m$. Figure 5-2 shows a three-dimensional plot of the surface; that the bulk of the error is contained within relatively few spots is clear from the figure.

Previous results with most CCD pixel noise removed using Experiment B (from Section 5.2.2) had a RMS depth estimation of 0.13$\mu m$. Our simulation results suggest that the spot deviations, as modeled above, contribute an error of approximately
Figure 5-2: Surface Plot using Experiment C
0.14μm;\textsuperscript{3} although this error exceeds the 0.06μm difference observed experimentally, to within the accuracy of our simulation it confirms our intuition regarding the effect of spot inconsistencies.

5.2.4 Test Object Flatness

Each of Experiments A through C above shifted the object between successive depth passes. If the mirror being used as a test object is not perfectly flat, it will contribute to the estimation noise.

Experiment D (see Table 5.3):
Perform sets of two full passes in depth over the object:
1. Full depth pass (average 5 frames), save blur measure curves $\rightarrow c_i(d)$
2. Another full depth pass (average 5 frames), save blur measure curves $\rightarrow c_i(d)$
3. Estimate offset between $c_i(d)$ and $c_i(d)$.

When compared with Experiment C (Section 5.2.3) this experiment is insensitive to mirror surface roughness, however we are using the same point on the object for both calibration and measurement which is a suspect procedure (ie, this is a repeatability test, see comments at start of Section 5.2).

Test results using Experiment D yielded a rms surface roughness of 0.023μm. This is an excellent result, however because of the reservation expressed in the previous paragraph, this will not be considered a practically attainable level of accuracy.

To be used in experiments with visible light, $\lambda \approx 500nm$, a test flat should be accurate\textsuperscript{4} to at least $\lambda/10 \approx 0.05μm$ or $\lambda/20 \approx 0.025μm$. That our results in Experiment C, which included the effects of mirror roughness, are 0.047μm greater than those from Experiment D is amazingly consistent with the anticipated mirror accuracy, and lends at least some credence to the suggestion that our apparatus may in fact be capable of accuracies approaching 0.023μm.

\textsuperscript{3}The quoted value in the table is 0.165μm. That particular entry was recalculated using a smaller maximum likelihood sampling step to more accurately represent Experiment C.

\textsuperscript{4}Fortunately the actual manufactured tolerances for the mirror are not available.
5.2.5 Stage Motions

It is difficult to say precisely what errors remain unaccounted for in Experiments C and D. A number of possibilities come to mind:

- Small amounts of residual pixel noise,
- Uncorrected CCD pixel offsets, sub-pixel movement of pattern spots,
- Micrometer jitter,
- Miscellaneous stage vibrations, shifting etc.

It is suspected that stage motion and micrometer jitter contribute a significant portion of the noise at this point. Certainly it seems unlikely that the stages, on the order of 15 cm in size, should be capable of maintaining strict rigidity on the order of 0.02 microns.

Experiment E (see Table 5.3):
1. Allow apparatus to stand idle for some time (24 hours or so).
2. Perform experiment C or D multiple times in immediate succession.

Observation of the results of Experiments C and D over time, as shown in Table 5.6, reveals additional information not discussed in the previous two sections: initial use of the apparatus incurs significantly more error than later runs. Two possible explanations come to mind:

- The error is due to micrometer jitter. When not in use, dust or oil may settle on the micrometer threads, making initial use somewhat rough. Repeated travel over the same micrometer section cleans the thread and makes the travel even.

- The error is due to stage vibration or shifting. After the stage has rested for some time (many hours), with initial use the stage may shift and settle somewhat unpredictably. After consistent motions have been performed over a period of hours, these motions may be reduced in magnitude or become deterministic (in which case they are absorbed into the calibration).
<table>
<thead>
<tr>
<th>Experiment</th>
<th>First Test</th>
<th>Last Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0.138</td>
<td>0.069</td>
</tr>
<tr>
<td>D</td>
<td>0.136</td>
<td>0.023</td>
</tr>
</tbody>
</table>

Time (hours) 0 2 4 6

Table 5.6: Experimental Results Over Time (RMS Roughness in μm)

Although a firm conclusion regarding the data in Table 5.6 cannot be made, the fact that the rms error decreases by at least a factor of two over time suggests that the manner in which the physical system is initialized is significant. In cases where high precision measurements are to be made, the experimenter may wish to have the apparatus execute several depth passes before collecting data.

5.3 Ideal Specular Surface – Further Considerations

5.3.1 Sampling Size

Section 3.5.3 introduced sampling and its effect on the variance of depth estimates. A theory was developed, and simulation results were presented to support the theory. This section presents actual tests on a mirror using a variety of sampling intervals.

The simulations in Section 3.5.3 assumed that the number of data points \( N \) was fixed (so that as the sample step \( \Delta \) increases, so does the sampled depth range \( N\Delta \)). In the following experimental tests the depth range is constant, so \( N \propto 1/\Delta \).

Table 5.7 shows the results of three tests using the difference-squared blur measure. The tests are performed using Experiment B (see Table 5.3).

Program Sim_Sampling_C (in Appendix B) generated the set of simulation results shown in Table 5.8. The simulator uses the same estimator and the same calibration curve as used in the experimental tests above.

The simulation does not take differences in pattern spots (ie, deviations from the calibration curve) into account, so it comes as no surprise that the errors reported in the simulation are lower than experimental errors. There are, however, two important
Table 5.7: Experimental Depth RMS Error (\(\mu m\)) vs. Sampling Step \(\Delta\)

<table>
<thead>
<tr>
<th>Test #</th>
<th>2.5</th>
<th>5.0</th>
<th>7.5</th>
<th>10.0</th>
<th>12.5</th>
<th>15.0</th>
<th>17.5</th>
<th>20.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.23</td>
<td>0.45</td>
<td>0.30</td>
<td>0.42</td>
<td>0.80</td>
<td>1.40</td>
<td>2.61</td>
<td>3.24</td>
</tr>
<tr>
<td>2</td>
<td>0.15</td>
<td>0.18</td>
<td>0.20</td>
<td>0.44</td>
<td>0.74</td>
<td>1.41</td>
<td>2.69</td>
<td>3.27</td>
</tr>
<tr>
<td>3</td>
<td>0.22</td>
<td>0.22</td>
<td>0.24</td>
<td>0.33</td>
<td>0.68</td>
<td>1.59</td>
<td>2.43</td>
<td>3.49</td>
</tr>
</tbody>
</table>

Table 5.8: Simulated Depth RMS Error (\(\mu m\)) vs. Sampling Step \(\Delta\)

<table>
<thead>
<tr>
<th>Test #</th>
<th>2.5</th>
<th>5.0</th>
<th>7.5</th>
<th>10.0</th>
<th>12.5</th>
<th>15.0</th>
<th>17.5</th>
<th>20.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.23</td>
<td>0.45</td>
<td>0.30</td>
<td>0.42</td>
<td>0.80</td>
<td>1.40</td>
<td>2.61</td>
<td>3.24</td>
</tr>
<tr>
<td>2</td>
<td>0.15</td>
<td>0.18</td>
<td>0.20</td>
<td>0.44</td>
<td>0.74</td>
<td>1.41</td>
<td>2.69</td>
<td>3.27</td>
</tr>
<tr>
<td>3</td>
<td>0.22</td>
<td>0.22</td>
<td>0.24</td>
<td>0.33</td>
<td>0.68</td>
<td>1.59</td>
<td>2.43</td>
<td>3.49</td>
</tr>
</tbody>
</table>

Observations:

- For small values of \(\Delta\) the estimation RMS error rises much slower than linear (the dependence is approximately square root, as is suggested by theory). Measurement speed can be increased by four or five times by increasing \(\Delta\), with only a doubling in the error.

- In all cases (experimental and simulated) the error begins to rise very rapidly for \(\Delta \approx 12.5\mu m\).

In the spirit of producing results applicable at different levels of equipment scaling (Section 3.3) it may be more useful to quote the above threshold in terms of various system parameters. From Equation 3.50 the depth-of-focus is

\[
\delta = \frac{8.86d_l^2}{d_2d_l^2} \max \left( \frac{r_s}{2.215}, \frac{r_l}{4} \right) \simeq 10\mu m
\]  

(5.18)

Our sampling threshold, at which point the error rapidly increases, occurs for \(\Delta \simeq 12.5\mu m\), i.e,

\[
\text{Sampling Threshold } \Delta \simeq \frac{11.1d_l^2}{d_2d_l^2} \max \left( \frac{r_s}{2.215}, \frac{r_l}{4} \right)
\]  

(5.19)

Increasing the spacing between samples in depth has been shown to be feasible. Practical surface profiling implementations may anticipate speed increases of four to
five times over our finely sampled implementation.

5.3.2 Spot Pattern Improvement

Although Section 5.2.3 removed the bulk of the error associated with spot differences, it is interesting to consider briefly how the estimation error might be further reduced with a more carefully prepared pattern.

Figure 5-3 shows the relative magnitudes of the rms depth estimation error values: the radius of each circle is proportional to the rms error. The data is derived from a series of test runs from Section 5.2.4 (all errors except those due to mirror flatness deviations and stage motions are removed) and Section 5.2.5 (all possible errors except those due to stage motion are removed). Three observations should be made from this figure:

- A significant portion of the error is due to a relatively small number of spots on the pattern.
- A higher concentration of bad estimates is present around the edge of the screen – these spots can drift out of the camera field of view and confuse the estimator.
- During the experimental test runs, the light source was biased slightly upwards and to the right of center. The corresponding region in the figure has a conspicuously lower average error rate.

The largest five or six circles in the figure correspond to visible defects in the pattern (the spot corresponding to the large circle at the upper left is nearly completely opaque to light).

As an interesting exercise, the experimental data from Section 5.2.4 was reanalyzed under a variety of scenarios: discarding some fraction of the worst spots, discarding some fraction of the worst and best spots etc. Table 5.9 shows the actual scenarios tested and the resulting rms depth estimation errors. Although it is difficult to say which scenario most accurately corresponds to a high quality pattern achievable in practice, the results shown in the table suggest that error reductions between 25%
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Sect. 5.2.4 Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pass 3</td>
</tr>
<tr>
<td>No Changes</td>
<td>0.087</td>
</tr>
<tr>
<td>Remove Worst 10%</td>
<td>0.078</td>
</tr>
<tr>
<td>Remove Worst 25%</td>
<td>0.063</td>
</tr>
<tr>
<td>Remove Best &amp; Worst 20%</td>
<td>0.076</td>
</tr>
<tr>
<td>Remove Best &amp; Worst 40%</td>
<td>0.064</td>
</tr>
</tbody>
</table>

Table 5.9: Depth RMS Errors (in \(\mu m\)) for Improved Pattern

and 30% would not be unreasonable. This suggests a lower limit in accuracy for our particular apparatus of about 0.05\(\mu m\).

### 5.3.3 Depth Postprocessing

The depth estimator (Section 3.4.2) implemented in this thesis calculates not only the estimate \(\hat{d}\) itself, but also the residual

\[
\text{Residual} = \sum_{i=-\infty}^{\infty} (\alpha y(i) - \beta b(i - d - 1) - (1 - \beta) b(i - d))^2, \ 0 \leq \beta \leq 1 \quad (5.20)
\]

where \(y()\) is the calibration curve, \(b()\) the observed curve, and \(\alpha\) the gain constant between \(y()\) and \(b()\) (see Section 3.4.2).

The residual is a measure of the fit between the curves \(y()\) and \(b()\); a large residual implies a poor fit, which suggests a poor estimate. This leads to two possible uses:

- **Identification of Bad Estimates:**
  If the residual exceeds some threshold, the estimate at that point is considered invalid. Such an approach is used in Section 5.6 in distinguishing between areas on the object which are seen normally as opposed to obliquely.

- **Correction of Bad Estimates:**
  If the surface of the object is constrained in some way (for example, a limit to the rate of change of surface curvature), then a smoothing algorithm (e.g., Wiener Filtering[39]) might be applied to the object; in other words, if proximate points on the object are correlated then the estimates can be improved. Estimates
<table>
<thead>
<tr>
<th>Comments</th>
<th>$d_i$</th>
<th>$d_o$</th>
<th>$d_i$</th>
<th>View Size</th>
<th>Accuracy Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micron Scale</td>
<td>5mm</td>
<td>7mm</td>
<td>20cm</td>
<td>0.5mm</td>
<td>$\approx 0.1\mu m$</td>
</tr>
<tr>
<td>Centimeter Scale</td>
<td>2cm</td>
<td>5cm</td>
<td>7.1cm</td>
<td>1cm</td>
<td>$\approx 4\mu m$</td>
</tr>
<tr>
<td>Meter Scale</td>
<td>10cm</td>
<td>0.7m</td>
<td>5cm</td>
<td>20cm</td>
<td>$\approx 200\mu m$</td>
</tr>
</tbody>
</table>

Table 5.10: Accuracy Examples for Scaled Systems

having near zero residuals are unaffected; poor estimates (large residuals) are smoothed based on neighboring data. This algorithm was not implemented for this thesis.

5.3.4 System Scaling Examples

Section 3.3 discussed the form in which accuracy is expected to scale as system parameters are modified. Having determined a limit to measurement accuracy for mirrors, Table 5.10 gives some examples showing how this accuracy could scale to other applications. The variables listed in the table are based on Figure 3-13 (page 46). The equations governing the scaled error are as follows:

$$\text{Error} \propto \frac{d^2}{d_i d_o}$$

(5.21)

$$\frac{\text{Field-of-View} \cdot d_i}{d_o} = \text{Constant}$$

(5.22)

The latter equation represents the constraint imposed by a constant camera size.

The first line of the table shows system parameters similar to those in our system. The third line shows a system having a 20cm field of view - more realistic for many applications. Note however, that the camera is located 5cm away from a 10cm lens; in practice $d_i$ would need to be increased, sacrificing measurement accuracy.

5.4 Other Surfaces – With Pattern

The previous sections discuss in detail our ability to estimate surface data when viewing an ideal object. Certainly other problems arise when non-ideal objects are considered. The performance figures obtained using a mirror establish the limits
to accuracy from our apparatus; now we wish to establish approximate levels of
performance for other surfaces.

5.4.1 Chalk

This class of material was considered in Section 3.2.2. Chalk may be characterized
as highly diffuse, low reflectivity, and having a significant light diffusing length (ie, a
point spread function significantly larger than one pixel).

Expectations:

- The individual reflecting elements on the surface (chalk dust) are very small, so
  no glinting is expected.

- The low reflectivity and smearing effects of the point spread function will sig-
  nificantly reduce the signal-to-noise ratio, so frame averaging may be required.

- Since the point spread function is low pass in nature, whereas the difference-
  squared algorithm is high pass, the low pass FFT blur measure may in this case
  outperform the difference-squared measure.

- Since chalk is a loose, porous material, the degree to which the surface is even
  defined is limited, so even a perfect estimator would incur error.

We would like to predict the effect of the broad point spread function of chalk on
depth estimation accuracy. If we model the multiple reflections of light as a random
walk of steps having Gaussian probability distributions, then the PSF is Gaussian
shaped. Empirically, we observe that the PSF is on the order of the size of a patter
spot. The effect of such a PSF is studied with the following simulation:

1. Define a 3 by 3 array of spots separated by 16 pixels; each spot has radius 3.2
   pixels.
2. Apply a simple low pass filter to simulate diffraction effects\(^5\).
3. Normalize this image and find the difference-squared and FFT blur values.

\(^5\)The filter used is \(h(0, 0) = 0.4, h(1, 0) = h(-1, 0) = h(0, 1) = h(0, -1) = 0.2.\)
4. Simulate the PSF using the diffusion filter $\exp\left(-\left(x^2 + y^2\right)/36\right)$.
5. Normalize this image and find the difference-squared and FFT blur values.

Simulation results:
The peak of the difference-squared blur curve falls to $1/140$ its value from step 3 to step 5. The peak of the FFT blur curve falls to $1/41$ its value from step 3 to step 5.

If these results are applied to the blur curve peak values obtained from the mirror:

<table>
<thead>
<tr>
<th>Diff. Squared:</th>
<th>Mirror Peak: 15</th>
<th>Projected Ratio: 140</th>
<th>Chalk Peak: 0.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT:</td>
<td>Mirror Peak: 8500</td>
<td>Projected Ratio: 41</td>
<td>Chalk Peak: 220</td>
</tr>
</tbody>
</table>

These results suggest that, relative to the difference-squared blur measure, the SNR of the FFT measure improves by a factor of $140/41 \simeq 3.5$ in going from using a mirror as test object to chalk. Since the FFT gave slightly inferior results for the mirror, we expect it to outperform the difference-squared measure by a factor of 2.5 to 3.

The simulation was performed on other diffusion radii and other PSF. The projected values for the blur curve peak values changed, however the projected relative improvement of the FFT versus the difference-squared measure was insensitive to such changes.

The observed pixel values range from 35 to 50. The observed peak value for the FFT blur curve is 220; the peak value for the difference-squared blur curve is 0.15 (compare with projected values above).

So using the FFT measure and summing $N$ frames:

$$RMSError = \frac{\sqrt{3VM}}{\pi P}$$  \hspace{1cm} (5.23)

$$V \simeq N4 \cdot P \cdot \sigma^2_{fft}$$  \hspace{1cm} (5.24)

$$\simeq N4 \cdot P \cdot (16\sigma_{ccd}/45N)^2$$  \hspace{1cm} (5.25)

Where $P \simeq 220$  \hspace{1cm} (5.26)

So $RMSError \approx \frac{32\sqrt{3}M\sigma_{ccd}}{40\pi \sqrt{PN}}$  \hspace{1cm} (5.27)

$$\approx \frac{8.93 \mu m}{\sqrt{N}}$$  \hspace{1cm} (5.28)

Where we used $M = 20 \mu m$, $\sigma_{ccd} = 1.5$. 

113
<table>
<thead>
<tr>
<th>Blur Measure</th>
<th>Summed Frames $N$</th>
<th>Theory</th>
<th>Experiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>5</td>
<td>4.0</td>
<td>4.8</td>
</tr>
<tr>
<td>FFT</td>
<td>10</td>
<td>2.8</td>
<td>3.2</td>
</tr>
<tr>
<td>FFT</td>
<td>10</td>
<td>2.8</td>
<td>2.9</td>
</tr>
<tr>
<td>FFT</td>
<td>20</td>
<td>2.0</td>
<td>1.2</td>
</tr>
<tr>
<td>Diff Sq</td>
<td>1</td>
<td>21.4</td>
<td>19.3</td>
</tr>
<tr>
<td>Diff Sq</td>
<td>3</td>
<td>12.4</td>
<td>9.7</td>
</tr>
<tr>
<td>Diff Sq</td>
<td>5</td>
<td>9.6</td>
<td>5.1</td>
</tr>
<tr>
<td>Diff Sq</td>
<td>10</td>
<td>6.8</td>
<td>6.6</td>
</tr>
<tr>
<td>Diff Sq</td>
<td>20</td>
<td>4.8</td>
<td>3.3</td>
</tr>
<tr>
<td>Diff Sq</td>
<td>30</td>
<td>3.9</td>
<td>2.9</td>
</tr>
</tbody>
</table>

Table 5.11: Experimental Results using Chalk as Test Object

If we were to use the difference-squared measure,

$$\begin{align*}
RMS_{\text{Error}} &= \frac{2\sqrt{3}M\sigma_{\text{cd}}}{40\pi\sqrt{PN}} \\
&\approx \frac{21.4\mu m}{\sqrt{N}}
\end{align*}$$

(5.29) (5.30)

using the same values as before, but $P = 0.15$. \(^6\)

Table 5.11 lists a series of test results and compares them with the above simple theoretical expectations.

- Theory and experiment agree reasonably well. Clearly the bulk of the estimation error (particularly for lower values of $N$) is due to pixel noise.

- The FFT blur measure outperforms the difference-squared measure at comparable values of $N$. If frame grabbing is slow, this result suggests that the FFT measure may be appropriate in certain situations, in contrast to our conclusions based on mirror test results (see Section 5.2.3).

- The limit to estimation accuracy is approximately 1 to $2\mu m$.

---

\(6\) One might wonder whether such low peak values can be measured with any accuracy. When summing single frames, the peak is entirely obscured by noise. This value was calculated from a blur measure curve based on summing 10 frames, at which point the peak is small but observable.
5.4.2 Plastic

This class of material was considered in Section 3.2.2. Plastic may be characterized as somewhat specular, medium reflectivity, and having a small light diffusion length (ie, a point spread function significantly less than one pixel).

The object being studied was a microchip encased in hard black plastic. This test case is by no means representative of all types of plastic; in particular, other plastics may be considerably less specular and have a larger diffusion length. All tests were performed using the difference-squared blur measure.

Expectations:

- Viewing the plastic under a microscope suggests that the reflecting elements on the surface are commensurate with a pixel size. Image averaging over small sections of the surface may improve accuracy.

- The higher reflectivity and less smearing than in the case of chalk suggests that pixel noise will play a less dominant role in determining errors.

Test 1:

Test the accuracy of the estimator for varying number of averaged frames $N$. The object is shifted between calibration and estimation passes:

<table>
<thead>
<tr>
<th># Frames N</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS Error (μm)</td>
<td>1.91</td>
<td>1.83</td>
<td>1.79</td>
<td>1.63</td>
<td>1.61</td>
</tr>
</tbody>
</table>

The error decreases, but only slowly, as $N$ is increased. These results include the surface roughness of the plastic, which may be significant.

Test 2:

Test the accuracy of the estimator for various $N$, but do not shift the object between calibration and estimation passes (ie, surface roughness not included in error):

<table>
<thead>
<tr>
<th># Frames N</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS Error (μm)</td>
<td>2.08</td>
<td>1.50</td>
<td>1.14</td>
<td>0.88</td>
</tr>
</tbody>
</table>

The above error is proportional to $1/\sqrt{N}$, ie, it is white noise dominated.
Figure 5-4: Object Shifting Between Grabbed Frames

Test 3:
In an effort to remove the effects of glint around the pattern spots the following experiment is proposed (see Figure 5-4): a set of 4 frames is averaged as before, but between grabbing successive frames, the object is shifted slightly. The amount of shift is large enough (greater than the reflective facet size) to average out glint, but not so large that the surface depth at any spot varies significantly. Two test runs were performed (on different sections of plastic):

<table>
<thead>
<tr>
<th>Motion Type:</th>
<th>None</th>
<th>Up / Down</th>
<th>Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1. RMS Error (μm)</td>
<td>1.48</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>Run 2. RMS Error (μm)</td>
<td>2.11</td>
<td>0.50</td>
<td>0.48</td>
</tr>
</tbody>
</table>

The degree of improvement varies, however averaging is very effective, generally reducing the error by a factor of two. The type of shifting used (up/down versus square) is of lesser importance.

5.4.3 Copper

This class of material was considered in Section 3.2.2. Copper may be characterized as very specular, high reflectivity, and negligible light diffusion length.

---

7This condition is assumed here. It may not be valid for all objects (in particular those having steep slopes or edges).
Expectations:

- The surface is fairly reflective (more so than chalk or plastic), so pixel noise will not be significant.
- The reflecting facets for copper are larger than those for plastic. Image averaging over small sections of the surface should yield significant improvements in accuracy.

The following results were obtained for various number of summed frames $N$; the object was not shifted between grabbing frames or between successive depth passes.

<table>
<thead>
<tr>
<th># Frames $N$</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error $\mu$m</td>
<td>2.7</td>
<td>2.2</td>
<td>1.9</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Summing increasing number of frames (ie, increasing reduction of pixel noise levels) has little or no effect on the test results.

The remaining tests performed using copper as the test surface are shown in Table 5.12. We make the following observations:

- Positions 1, 2 and 4 (first two rows in the table) were selected to represent “clean” sections of the copper surface; position 3 represents a rougher section. Clearly the nature of the metal surface plays a strong role in determining depth accuracy.

- All rows in the table except for the fourth row are repeatability tests (no horizontal movement between successive depth passes); the test corresponding to the fourth row shifted the object horizontally between depth passes. That rows three and four give comparable accuracies suggests that the repeatability tests may be a realistic indicator of the accuracy limit.

- The benefit obtained from averaging over the image surface is undeniable; a reduction in error by a factor of two or three is not uncommon. Unlike the results obtained from plastic (previous section), the increased averaging offered by square shifting (see Figure 5-4) results in further reductions in error.
## Table 5.12: Penny Tests with Pattern, Errors in $\mu$m

<table>
<thead>
<tr>
<th># Frames $N$:</th>
<th>4</th>
<th>4</th>
<th>4</th>
<th>4</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movement:</td>
<td>None</td>
<td>Up/Down</td>
<td>Left/Right</td>
<td>Square</td>
<td>Up/Down</td>
<td>Square</td>
</tr>
<tr>
<td>Posn. 1</td>
<td>2.9</td>
<td>1.3</td>
<td>1.0</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Posn. 2</td>
<td>3.3</td>
<td>1.0</td>
<td>1.0</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Posn. 3</td>
<td>3.3</td>
<td>2.4</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Posn. 4</td>
<td>2.7</td>
<td>2.3</td>
<td>1.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Posn. 4</td>
<td>3.1</td>
<td>1.6</td>
<td>0.5</td>
<td>0.8</td>
<td>0.9</td>
<td></td>
</tr>
</tbody>
</table>

The test results above suggest that one micron accuracies may be achievable for many parts of the penny, although this accuracy cannot necessarily be guaranteed over the entire surface. An accuracy of two microns would seem attainable given the above results.

### 5.5 Other Surfaces – Without Pattern

Two primary difficulties arise in performing practical experiments without a pattern:

- Section 4.1.2 discussed tracking the position of pattern spots as the depth position was varied; the tracking problem is tractable since the number and approximate locations of the spots are known. Without the presence of a pattern, the object still shifts laterally as it is moved in depth;\(^8\) however tracking the motion is difficult.

- Our equipment operates at micron accuracies; we have no objects available for study with simple patterns (as would be the case for a system scaled to larger sizes).

A set of test results is shown in Table 5.13. Each test was performed as follows: perform two passes in depth with no horizontal shift between the passes; compute the statistics based on the difference in results between the two passes. The test just

---

\(^8\)This is not an inherent problem in the depth-from-focus method, rather it is caused by the failure, in our particular apparatus, of the Z-axis to be perfectly colinear with the camera’s optical axis. A more precise axis calibration would reduce the shifting effect. Nevertheless, the effect is present in our apparatus and affects our ability to perform certain experiments.
<table>
<thead>
<tr>
<th>Summed Frames $N$</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1:</td>
<td>5.84</td>
<td>2.45</td>
<td>1.08</td>
</tr>
<tr>
<td>Test 2:</td>
<td>5.01</td>
<td>2.51</td>
<td>0.37</td>
</tr>
<tr>
<td>Test 3:</td>
<td>5.05</td>
<td>2.74</td>
<td>0.92</td>
</tr>
<tr>
<td>Test 4:</td>
<td>5.01</td>
<td>2.74</td>
<td>0.54</td>
</tr>
</tbody>
</table>

Table 5.13: Test Results on Copper, No Pattern (Errors in $\mu m$)

The following preliminary conclusions can be made:

- Materials such as copper have sufficient intrinsic surface detail at one micron resolutions to permit repeatable depth measurements.

- As the number of summed frames ($N$) increases the estimation error decreases. Preliminary tests suggest that for sufficiently large $N$, the error increases again (possibly due to stage shifting over time). For our apparatus, the error increased for $N \geq 8$.

- If the repeatability results of Table 5.13 also hold as realistic accuracy values, then depth accuracy around the one micron level may be achievable.

- Accuracy varies considerably with the nature of the surface. Around pits and gouges in the metal (where reflectance decreased) the depth error (and the corresponding residual value) increased.

The work reported above is at best a proof of principle: depth estimation from metals at micron resolutions is a definite possibility, and accuracies approaching one micron may be expected. Further work is required to demonstrate this convincingly.

### 5.6 Practical Experiments

In terms of the test statistics reported, this chapter has primarily presented summarized data in table format. Although this is a simple, succinct form in which to convey results, more applications oriented readers may wonder whether the apparatus used
in this thesis is capable of measuring anything other than flat surfaces. Although it may not significantly increase our understanding of depth-from-focus estimation, this section will present a few practical applications of the depth estimation system.

Each test will include a short discussion and two plots: the depth estimates, and the residual values (see Equation 5.20). The residual is used to distinguish between good and poor estimates (see Section 5.3.3). Our purpose then in plotting residual values is to demonstrate the ability to distinguish the estimates well; the numerical values of the residuals are not of interest and are not shown on the plots.

Each contour plot uses Spot # as the unit for the X and Y axes. On the X axis, Spot # = n refers to the data value corresponding to the n\textsuperscript{th} pattern spot from the left edge of the camera image.

**Test 1 - Plastic**

The plastic microchip used in Section 5.4.2 has a small “9” stamped on one face (see Figure 5-5); a small fraction of this digit filled the camera’s field of view.

Figure 5-6 shows the resulting depth estimates and the estimator residual. The depth results clearly distinguish the raised digit from the surrounding area. The residual gives an excellent discrimination between data points on flat surfaces and transition values; the majority of the poor depth estimates lie within this transition.

120
region and are discarded based on a threshold test on the residual.

With the removal of the transition spots, the remaining points (considering the digit and the surrounding area separately) have a roughness of about 4 microns. This is somewhat inferior to our previous test results, however there are additional errors present here:

- The digit is not perfectly flat (as assumed in the 4 micron statistic), nor is the background perfectly flat; both may have been rounded in the stamping process.

- The choice of a threshold for the residual to distinguish between transition and data values is ad-hoc. The 4 micron statistic includes numerous data points that might have been rejected by human viewers as transition values.

**Test 2 - Copper**

The penny used in Section 5.4.3 will be used again here.

On the side of the penny showing the Lincoln Memorial may be found the inscription *E PLURIBUS UNUM*. On either side of *UNUM* may be found a small period; the left period is imaged in this case.

Figure 5-7 shows the depth estimates for the image (3D plot); the corresponding residual values are shown in the contour plot (Figure 5-8). The transition spots (those whose residual exceeds some threshold) have had their depth values lowered below the rest of the plot. The depth values plotted here are the raw output from the estimator - no curvature of focus correction has been applied.

The residual values accurately separate the period from the rest of the image; the shape of the period (which is not circular) is faithfully reproduced.

**Test 3 - Copper, No Pattern**

A final test is performed, again using the same copper penny, this time without a pattern. The image shown is the edge of one of the columns in the Lincoln Memorial. The depth results are shown in Figure 5-9; the corresponding residual values are plotted in Figure 5-10.
Figure 5-6: Plastic Test Results
Figure 5-7: Copper Depth Plot

Figure 5-8: Contour Plot of Residual Values
Figure 5-9: Copper Depth Plot - No Pattern

Figure 5-10: Contour Plot of Residual Values - No Pattern
All of the depth estimates (3D plot) are plotted, including the transition values. Even in the transition region the estimates form a relatively smooth junction between the background and the raised portion of the column. The residual values (contour plot) show the same discriminating power as those when the pattern was in place (previous two examples).

5.7 Conclusions

The results of this chapter are encouraging: RMS estimation errors below 0.1 micron for ideal objects, and are about 1-2 microns for a variety of sub-ideal test objects. Such error levels may make depth-from-focus a powerful tool in surface profiling.

It is important to keep the experimental context in mind under which the above results were obtained. The results presented in this chapter represent our estimate of the limit of performance; this limit may not be attainable under all circumstances:

- All of our tests use one field of view of the camera. Profiling a larger object requires combining a set of views; accuracy will likely deteriorate in performing this combination.

- The tests of this chapter observed either flat objects or objects having small height increments. Measurement of larger height offsets puts greater linearity demands on the micrometers; for sufficiently large offsets in depth the ability to position the object will be the limiting factor determining the error rather than our ability to estimate depth from focus information.
Chapter 6

Conclusions

This chapter lists the main achievements of the thesis, and a set of possibilities for future research.

6.1 Thesis Achievements

If there may have been doubts before, depth-from-focus techniques have demonstrated the potential to be used in serious surface profiling applications including, most importantly, advanced manufacturing processes. Although the work in this thesis falls short of a full-fledged system that could be used in production in terms of speed, the measurement accuracies documented here suggest that speed may be the last major hurdle in the way of making depth-from-focus systems a practical reality. If the suppositions of Chapter 1 are correct - that speed may always be increased by using high speed components and custom-built hardware, then this last hurdle may pose only a relatively minor barrier in terms of required research.

A large number of results have been documented in this thesis, including setting new depth estimation error limits and improving our intuition regarding the limits to performance of the apparatus under various settings.

The following items, although by no means forming a comprehensive list, highlight the more significant results:

- Past efforts using depth-from-focus methods concentrated on advanced diffrac-
tion equations to model the optical system of the apparatus. This thesis sug-
gests that simple diffraction equations (e.g., Rayleigh's criterion) are sufficient
to describe the nonclassical aspects of the system. Certain obvious features of
the system (asymmetry of focus) cannot be explained by thin-lens classical or
diffraction optics; however ray-traced thick-lens optics performs the task well.

• In researching a thesis such as this one, it is easy to address the problem at hand
in too narrow a context: how does one optimize depth measurement accuracy
for a particular apparatus at a particular level of magnification? An interest in
maintaining a broader perspective motivated the discussion on system scaling
which describes how accuracy varies as system dimensions are varied, and it
motivated the extensive tests on optically ideal objects (the limit towards which
most materials tend as dimensions increase).

• Any system such as ours, which is meant to operate at high accuracy, requires a
variety of mechanical and optical tests or calibrations. A variety of calibrations,
associated algorithms, and computer programs were presented and discussed.
It is hoped that future system designers will be able to take advantage of the
experience gained with this project. In particular, future experiments might be
designed with added foresight and avoid certain calibration issues altogether.

• A series of tests were performed on several different objects. The best results,
obtained using computationally intensive tests on an ideal object (a mirror)
yielded repeatable tests having RMS errors of about 0.05 microns. Simpler tests
on mirrors routinely produced RMS errors of about 0.1 microns. For less ideal
materials (such as plastic, chalk, and copper), RMS error levels were between
one and two microns.

6.2 Future Research Possibilities

Depth-from-focus research has by no means been exhausted by this thesis, as much
as the thesis author might wish this to be the case.
I should be disappointed if readers of this thesis could not find a number of interesting extensions to this research. Occasional items which require further consideration are mentioned throughout the thesis as they appear. No effort has been made to compile a complete list of them below, however the following represent some of the more significant tasks requiring completion before high speed depth-from-focus systems become a reality:

- In this thesis, the size of the area to which surface estimation techniques may be applied is limited by the field of view of the camera through the optical system. In practice, objects requiring surface profiling are frequently larger than 0.5mm, so provision must be made for combining depth estimates from multiple fields of view into an aggregate whole. This raises a series of practical issues:

  - Is object movement open loop (relying on the linearity of the micrometers), or do we observe and calibrate the position of the object by observing surface features?

  - How much overlap is required between adjacent images to ensure good data continuity?

- All tests in this thesis used objects mounted more or less perpendicular to the optical axis; portions of the object seen at large angles were denoted as transition areas, and no data were collected for these. While this approach may satisfy some applications, a general instrument must be capable of taking measurements at a variety of angles. Further tests should be performed to determine the sensitivity of the depth estimation algorithms to the angle at which the object is viewed. If the algorithms perform poorly in these tests, new less sensitive algorithms should be proposed and tested.

- A practical system will require large increases in speed. Certainly significant increases may be realized from custom hardware, however this may prove inadequate for very fast profiling. For such applications the “brute force” estimators described in this thesis may be inappropriate; linearized estimators may be
found which introduce only minimal accuracy sacrifices for significant speed increases.

- Many practical manufacturing applications require surface profiling over an area of many centimeters or even many meters (e.g., [22]). Clearly for such applications, combining many fields of view of 0.5mm each (see previous page) is a ridiculous proposal. Instead, the depth-from-focus system must be scaled. Efforts have been made in this thesis to address such changes of scale. While applying depth-from-focus methods to large scale systems appears promising, such an apparatus should be constructed to demonstrate viability.
Appendix A

Calibration Code Listings

<table>
<thead>
<tr>
<th>Calibration programs in this appendix:</th>
<th>Page #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cal_Curve.C</td>
<td>131</td>
</tr>
<tr>
<td>Cal_Pixel.C</td>
<td>145</td>
</tr>
<tr>
<td>Cal_Screen.C</td>
<td>150</td>
</tr>
<tr>
<td>Cal_Spot.C</td>
<td>156</td>
</tr>
</tbody>
</table>
/* This program does a careful analysis of the depth of each section of the screen. This program determines the second order curvature of depth, the ensemble averaged calibration curve, and the the noise about the calibration curve. These results may either be used to determine curvature of field corrections or the offset from perpendicularity between the object and the optical axis.

The initial calibration curve may be supplied in a file, extracted from a particular spot, or extracted from a region on the screen.

Output files:
  cal_curv.reg  Regression results
  cal_curv.ofs  Offsets of each curve from calibrated
  cal_curv.mul  Statistics of multiplier
  cal_curv.cal  Calibration curve estimates and noise values
*/

/* Assumptions:
Camera enabled
Mirror in place and in focus
Stepper Motors enabled
*/

/* Misc parameters */
#define Lamp_Level  255 /* 0 to 255 */

/* Calibration statistics parameters (at least during debugging) */
#define Side_Pixels   16 /* Power of two */
#define Depth_Slices  60
#define Depth_Increment    1
#define Blur_Measure  Diff_Squared
#define Summed_Frames  1 /* # frames to average before applying transform */
#define Frame_Buf_Size   (GRABBER_WIDTH*Side_Pixels)
#define Dep_Avg_Passes  2 /* Number of passes over depth to establish est. averages */
#define Dep_Nse_Passes  0 /* Number of passes over depth to determine noise levels */
#define Depth_Passes   (Dep_Avg_Passes + Dep_Nse_Passes)
#define Cal_Area_x   240
#define Cal_Area_y   224
#define Cal_Area_Side 2 /* Measured in units of Side_Pixels */
#define Cal_Depth_Ofs 10 /* Max. Offset to check for correlator */
#define ML_Depth_Ofs  3 /* Maximum error to check from correlation */
#define ML_Depth_Step 0.0625 /* Depth step in calibrating (test uses interpolation) */
#define Horz_Margin  0 /* Margins to avoid in gathering statistics, */
#define Vert_Margin  0 /* due to lapped transform or bad lighting */
/* measured in blocks of Side_Pixels */
#define Motor_Osc transported 10 /* Motor steps to exceed initial run to avoid backlash */
#define Capture_Wait 50 /* Delay in ms after screen capture */
#define Use_Cal_Coords 1 /* 1 - Use external calibrated coordinate centers */
#define Coord_File "\calibrat\spot.dat" /* from Cal_Spot.C */
/* I - external calibration curve given */
#define Curv_File "\temp\a.bak"
#define Curv_Spot 344 /* spot number to use if no file given */
#define Reg_Size 6 /* number of regression parameters */
#define Reg_File "cal_curv.reg"
#define Multi_File "cal_curv.mul"
#define Cal_File "cal_curv.cal"
#define Ofs_File "cal_curv.ofs"

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <alloc.h>
#include <alloca.h>
#include <math.h>
#include <fcntl.h>
#include <mem.h>
#include <io.h>
#include <sys/stat.h>
#include <string.h>
#include "new3d.h"

void Image_Normalize ( int *, int *, float *, int, int, int );

void main ( int argc, char **argv )
{
    /* Program Variables */
    FILE *coord_handle;
    float *Image_Norm, *Diff_Coeffs;
    unsigned char *Image_Mem;
    int *Frame_Mem[GRABBER_HEIGHT];
    coord_rec *root, *temp_ptr;
    float cal_curve[Depth_Slices], cal_curve_est[Depth_Slices];
    int cal_curve_count[Depth_Slices];
    float temp_float;

    /* Frame Grabbing and Storage Variables */
    int section, num_sections, start_line, end_line, lines_per_section;
    int depth, coeff, frame, test, line_cnt;
    int num_coeffs, num_vals, xvals, yvals, temp;
    int spot_num, depth_pass;

    /* Estimation Variables */
    float max_val, max_pos, depth_ofs, corr_sum, cal_sum;
    float lower_mult, upper_mult, curv_diff, avg_mult;
    float ml_min, ml_offset, ml_sum;
    int depth_floor;
    int cur_depth, low_index, high_index;

    /* Statistics Variables */
    float *cof_sum, *cof_sq, gen_sum, gen_sq;
    float *spot_ofs, *spot_prob, *spot_ofs_avg;
    float (*nse_ofts[2], (*nse_est)[3], (*nse_depth)[3];

    /* Regression fit variables */
    float matrix_el[Reg_Size*Reg_Size], matrix_vect[Reg_Size], temp_vect[Reg_Size];
    float xval, yval;
/* Misc and Dummy Variables */
int xcoord, ycoord, start_y, end_y, i, j;

/* Setup coordinate structure */
root = malloc( sizeof( coord_rec ) );
temp_ptr = root;
um vals = 0;
if (Use_Cal_Coords)
{
    /* external coordinates to be supplied */
    coord_handle = fopen( Coord_File, "r");
    if (coord_handle == NULL)
    {
        printf( "Coordinate file not found.\n" );
        exit(1);
    }
while (fseek( coord_handle ))
{
    fscanf( coord_handle, "%d %d", &xcoord, &ycoord );
    if (!fseek( coord_handle ))
    {
        /* adjust coordinate if outside of screen region */
        temp_ptr->x = min( 0, xcoord-Side_Pixels/2, GRABBER_WIDTH-Side_Pixels );
        temp_ptr->y = min( 0, ycoord-Side_Pixels/2, GRABBER_HEIGHT-Side_Pixels );
        temp_ptr->spot_num = num_vals;
        temp_ptr->next = malloc( sizeof( coord_rec ) );
        if (temp_ptr->next == NULL)
        {
            printf( "Insufficient memory for list.\n" );
            exit(1);
        }
        temp_ptr = temp_ptr->next;
        num_vals++;
    }
}
temp_ptr->next = NULL;
fclose( coord_handle );
xvals = num_vals;
yvals = 1;
else
{
    /* setup block coordinates */
    num_vals = 0;
    for (j=Vert_Margin; j<GRABBER_HEIGHT/Side_Pixels-Vert_Margin; j++)
    for (i=Horz_Margin; i<GRABBER_WIDTH/Side_Pixels-Horz_Margin; i++)
    {
        temp_ptr->x = i*Side_Pixels;
        temp_ptr->y = j*Side_Pixels;
        temp_ptr->spot_num = num_vals++;
        temp_ptr->next = malloc( sizeof( coord_rec ) );
        if (temp_ptr->next == NULL)
        {
            printf( "Insufficient memory for list.\n" );
            exit(1);
        }
    }
temp_ptr = temp_ptr->next;
}
      temp_ptr->next = NULL;
  xvals = GRABBER_WIDTH/Side_Pixels-2*Horz_Margin;
  yvals = GRABBER_HEIGHT/Side_Pixels-2*Vert_Margin;
}

/* setup spot tracking routine */
if (set_initial_spot_list( num_vals )) exit( 0 );

/* Allocate Memory */
Image_Mem = farmalloc( Frame_Buf_Size * sizeof( char ));
Image_Norm = farmalloc( Side_Pixels * Side_Pixels * sizeof( float ));
Diff_Coeffs = farmalloc( num_vals * sizeof( float ));
for (i=0; i<num_vals; i++)
  Diff_Coeffs[i] = farmalloc( Depth_Slices * sizeof( float ));

spot_ofs = malloc( num_vals * sizeof( float ) );
spot_ofs_avg = malloc( num_vals * sizeof( float ) );
spot_prob = malloc( num_vals * sizeof( float ) );
for (i=0; i<num_vals; i++) { spot_ofs[i] = 0.0; spot_ofs_avg[i] = 0.0; }

cof_sum = malloc( num_vals * sizeof( float ) );
cof_sq = malloc( num_vals * sizeof( float ) );
for (i=0; i<num_vals; i++) { cof_sum[i] = 0.0; cof_sq[i] = 0.0; }

nsc_spot = malloc( num_vals * 2 * sizeof( float ) );
nsc_est = malloc( Depth_Slices * 3 * sizeof( float ) );
nsc_depth = malloc( Depth_Slices * 3 * sizeof( float ) );

/* divide remaining memory into buffer sections */
lines_per_section = (coreleft()*9/10) / (Frame_Buf_Size * sizeof(int));
if ((lines_per_section < 2) || (Image_Mem == NULL) || (Image_Norm == NULL) ||
  (Diff_Coeffs != NULL) || (Diff_Coeffs[num_vals-1] == NULL))
  { 210
     /* nowhere near enough memory */
     printf( "Insufficient Memory.\n" );

     deallocate_spot_list();
     free( Image_Mem );
     free( Image_Norm );
     if (Diff_Coeffs != NULL)
     { 220
       for (i=0; i<num_vals; i++)
         if (Diff_Coeffs[i] != NULL) free( Diff_Coeffs[i] );
      free( Diff_Coeffs );
    }
    exit(1);
  }
if ((GRABBER_HEIGHT/Side_Pixels) % lines_per_section == 0)
  { num_sections = (GRABBER_HEIGHT/Side_Pixels)/lines_per_section; }
else
  num_sections = 1+(GRABBER_HEIGHT/Side_Pixels)/lines_per_section;

for (i=0; i<lines_per_section; i++)
  Frame_Mem[i] = farmalloc( Frame_Buf_Size * sizeof(int ));
/ * read in calibration curve if supplied * /
if (Use_Cal_Curve == 1)
{
    cal_file = fopen( Curv_File, "r" );
    for (i=0; i<Depth_Slices; i++)
        fscanf (cal_file, "%f %f", &temp_float, cal_curve+i );
    fclose (cal_file );
}

/* open files */
regr_file = fopen( Regr_File, "w" );
mult_file = fopen( Mult_File, "w" );
cal_file = fopen( Cal_File, "w" );
offs_file = fopen( Ofs_File, "w" );

/* setup normal camera operation */
lamp( Lamp_Level );
dis0();
discam();

/* GATHERING DEPTH BLUR INFO */
gen_sum = 0.0;
gen_sq = 0.0;
classr();

/* loop over depth passes */
for (depth_pass=0; depth_pass < Depth_Passes; depth_pass++)
{
    gotoxy( 1, 15 );
    printf ("Processing pass %d", depth_pass );

    /* Reset estimated calibration curve */
    if (depth_pass < Dep_Avg_Passes)
    {
        for (i=0; i<Depth_Slices; i++)
            { cal_curve_est[i] = 0.0;
                      cal_curve_count[i] = 0;
            }
    }

    /* Back up stage to farthest back */
delay ( Motor_Wait );
movel ( Z_Axis, (long) Motor_Overshoot + Depth_Increment * Depth_Slices / 2,
              Neg_Axis, Motor_Speed );
delay ( Motor_Wait );
movel ( Z_Axis, (long) Motor_Overshoot, Pos_Axis, Motor_Speed );

    /* Waste one frame to set up frame grabber */
cap0();
delay( Capture_Wait );
for (line_cnt=0; line_cnt<GRABBER_HEIGHT/Side_Pixels; line_cnt++)
    /* grab line */
    get_band( line_cnt*Side_Pixels, Side_Pixels, Image_Mem );


/* Loop over depth levels */
for (depth=0; depth<Depth_Slices; depth++)
{
    gotoxy( 1, 1 );
    printf( "Depth Layer \%d", depth );

    /* Generous delay for ringing to dampen */
    delay( Motor_Wait );

    /* get calibration value if first time through data */
    /* Remove comments if auto-calibration from screen area desired. */
    if (depth_pass == 0)
    {
        capt0();
        delay( Capture_Wait );

        cal_curve[depth] = 0.0;
        for (j=0; j<Cal_Area_Side; j++)
        {
            get_band( Cal_Area.x+j*Side_Pixels, Side_Pixels, Image_Mem );
            for (i=0; i<GRABBER_WIDTH*Side_Pixels; i++)
                Frame_Mem[0][i] = (int) Image_Mem[i];

            for (i=0; i<Cal_Area_Side; i++)
            {
                Image_Normalize( Frame_Mem[0], Frame_Mem[0], Image_Norm,
                    Cal_Area.x+i*Side_Pixels, 0, Side_Pixels );
                Blur_Measure( Image_Norm, Side_Pixels, &temp_float, &num_coeffs );
                cal_curve[depth] += temp_float;
            }
        }

        for (line_cnt=0; line_cnt<GRABBER_HEIGHT/Side_Pixels; line_cnt++)
            get_band( line_cnt*Side_Pixels, Side_Pixels, Image_Mem );
    }

/* Loop over frame subsections */
if (get_spot_list( root, &temp_ptr, Side_Pixels )) temp_ptr = root;
    section=0;
    do
    {
        start_line = section * (lines_per_section - 1);
        end_line = start_line + lines_per_section;

        start_y = section * (lines_per_section - 1) * Side_Pixels;
        end_y = start_y + (lines_per_section - 1) * Side_Pixels;

        /* clear frame sum buffer */
        for (i=0; i<lines_per_section; i++)
            memset( Frame_Mem[i], 0, Frame_Buf_Size * sizeof(int) );

        /* loop over number of frames to sum */
    }

    /* get next frame */
    for (i=0; i<Summed_Frames; i++)
    {
        capt0();
        delay( Capture_Wait );
    }
/* grab line and add to buffer */
for (line_cnt=0; line_cnt<GRABBER_HEIGHT/Side_Pixels; line_cnt++)
{
    /* grab line */
    get_band( line_cnt*Side_Pixels, Side_Pixels, Image_Mem );

    /* add frame buffer section to summing buffer */
    if (line_cnt >= start_line && line_cnt < end_line)
        for (j=0; j<Frame_Buf_Size; j++)
            Frame_Mem[line_cnt-start_line][j] += (int) Image_Mem[j];
}
/* analyze summed frames */
while ((temp_ptr->next != NULL) && (temp_ptr->y >= start_y) &&
    (temp_ptr->y <= end_y))
{
    /* normalize section and call transform routine */
    Image_Normalize( Frame_Mem[(temp_ptr->y-start_y)/Side_Pixels],
        Frame_Mem[1+(temp_ptr->y-start_y)/Side_Pixels], Image_Norm,
        temp_ptr->x, (temp_ptr->y-start_y) % Side_Pixels, Side_Pixels );
    Blur_Measure( Image_Norm, Side_Pixels, &temp_float, &num_coeffs );

    /* Add results to array */
    Diff_Coeffs[temp_ptr->spot_num][depth] = temp_float;
    temp_ptr = temp_ptr->next;
} section++;
} while (end_y < GRABBER_HEIGHT-Side_Pixels+1);

/* advance stage to next depth position */
movel( Z_Axis, long Depth_Increment, Pos_Axis, Motor_Speed );
}
/* restore stage to initial position */
delay( Motor_Wait );
movel( Z_Axis, long Depth_Increment * Depth_Slices - Depth_Increment * Depth_Slices / 2,
    Neg_Axis, Motor_Speed );

/* copy calibration curve if required */
if ((depth_pass == 0) && (Use_Cal_Curve != 1))
    for (i=0; i<Depth_Slices; i++)
        cal_curve[i] = Diff_Coeffs[Curv_Spot][i];

/* process gathered data and estimate each depth vs. calibration curve */
gotoxy( 1, 3 ); credo();

/* evaluate integral of calibration curve */
    cal_sum = 0;
    for (i=Cal_Depth_Ofs+1; i<Depth_Slices-Cal_Depth_Ofs; i++)
        cal_sum += cal_curve[i] * cal_curve[i];

/* loop over each screen section */
for (spot_num=0; spot_num<num_vals; spot_num++)
{ gotoxy(1, 3);
 printf("Performing depth estimates - spot %d", spot_num);

 /* find approximate offset using correlator */
 max_val = 0.0;
 max_pos = -2*Cal_Depth_Ofs;

 /* find optimal depth and multiplier value */
 for (cur_depth = -Cal_Depth_Ofs; cur_depth <= Cal_Depth_Ofs; cur_depth++)
 { corr_sum = 0.0;
   for (i=Cal_Depth_Ofs+1; i<Depth_Slices-Cal_Depth_Ofs; i++)
     corr_sum += cal_curve[i]*Diff_Coeffs[spot_num][cur_depth+i];
   if (corr_sum > max_val)
     { max_val = corr_sum;
       max_pos = cur_depth;
     }
 }
 max_val /= cal_sum;

 /* find better estimate using maximum likelihood */
 ml_min = 1e20;
 ml_offset = 0.0;
 for (depth ofs = -ML_Depth_Ofs; depth ofs <= ML_Depth_Ofs; depth ofs += ML_Depth_Step)
 { ml_sum = 0.0;
   depth_floor = floor( depth ofs );
   lower_mult = 1 - (depth ofs - depth_floor);
   upper_mult = depth ofs - depth_floor;
   low_index = (max_pos > 0) ? ML_Depth_Ofs+1 : ML_Depth_Ofs+1-max_pos;
   high_index = (max_pos > 0) ? Depth_Slices-ML_Depth_Ofs-max_pos+1 :
                 Depth_Slices-ML_Depth_Ofs-1;
   for (i=low_index; i<high_index; i++)
     { temp_float = lower_mult*Diff_Coeffs[spot_num][depth_floor+max_pos+i]+
               upper_mult*Diff_Coeffs[spot_num][depth_floor+max_pos+i+1];
       temp_float -= cal_curve[i] * max_val;
       ml_sum += temp_float * temp_float;
     }
   if (ml_sum < ml_min)
     { ml_min = ml_sum;
       ml_offset = depth ofs;
     }
 } max_pos += ml_offset;
 spot_prob[spot_num] = ml_min;

 if (depth_pass < Dep_Avg_Passes)
{ /* we are still averaging data to obtain estimation coefficients */
  /* save best depth offset estimate */
  spot_offs[spot_num] = max_pos;
  spot_offs_avg[spot_num] += max_pos;

  cof_sum[spot_num] += max_val;
  cof_sq[spot_num] += max_val * max_val;
  gen_sum += max_val;
  gen_sq += max_val * max_val;

  /* update new calibration curve estimate */
  low_index = ceil( -max_pos );
  if( low_index < 0 ) low_index = 0;
  high_index = ceil( Depth_Slices - max_pos - 1 );
  if( high_index > Depth_Slices-1 ) high_index = Depth_Slices-1;

  depth_floor = floor( max_pos );
  lower_mult = 1 - (max_pos - depth_floor);
  upper_mult = max_pos - depth_floor;

  for (i=low_index; i<high_index; i++)
  {
    cal_curve_est[i] += lower_mult*Diff_Coeffs[spot_num][depth_floor+i] +
    upper_mult*Diff_Coeffs[spot_num][depth_floor+i+1];
    cal_curve_count[i]++;
  }
  gotoxy( 1, 5 );
  printf( "Performing end of estimates analysis." );

  /* determine estimated calibration curve */
  if( depth_pass < Dep_Avg_Passes )
  {
    for (i=0; i<Depth_Slices; i++)
      if( cal_curve_count[i] != 0 ) cal_curve_est[i] /= cal_curve_count[i];

  /* normalize estimated curve */
  temp_float = 0.0;
  for (i=Cal_Depth_Offset+1; i<Depth_Slices-Cal_Depth_Offset; i++)
    temp_float += cal_curve_est[i] * cal_curve_est[i];
  temp_float = sqrt( gen_sum / temp_float );
  for (i=0; i<Depth_Slices; i++)
    cal_curve_est[i] *= temp_float;

  for (i=0; i<Depth_Slices; i++)
    if( cal_curve_count[i] == 0 ) cal_curve_est[i] = cal_curve[i];

  /* output curve to file */
  for (i=0; i<Depth_Slices; i++)
    fprintf( cal_file, "%d %f
", i, cal_curve_est[i] );

  /* set new calibration curve -- optional */
for (i=0; i<Depth_Slices; i++) cal_curve[i] = cal_curve_est[i];
}

/* output most recent depth estimates */
if (depth_pass < Dep_Avg_Passes)
{
    temp_ptr = root;
    while (temp_ptr->next != NULL)
    {
        i = temp_ptr->spot_num;
        fprintf(ofs_file, "%d %d %f %f\n", i,
                temp_ptr->x, temp_ptr->y, spot_ofs[i], spot_prob[i]);
        temp_ptr = temp_ptr->next;
    }
    fprintf(ofs_file, "\n");
}

/* output multiplier values */
if ((depth_pass == Dep_Avg_Passes - 1) && (Dep_Avg_Passes > 1))
{
    for (i=0; i<num_vals; i++)
    {
        cof_sum[i] /= Dep_Avg_Passes;
        cof_sq[i] /= Dep_Avg_Passes;
        fprintf(multi_file, "%d %f %f\n", i,
                cof_sum[i], (float) sqrt(cof_sum[i] - cof_sum[i] * cof_sum[i]));
    }
}

/* analyze noise about previous and estimated calibration curves */
for (i=0; i<num_vals; i++) nse_spot[i][0] = nse_spot[i][1] = 0.0;
for (i=0; i<Depth_Slices; i++)
{
    nse_depth[i][0] = nse_depth[i][1] = nse_depth[i][2] = 0.0;
    nse_est[i][0] = nse_est[i][1] = nse_est[i][2] = 0.0;
}
if (depth_pass >= Dep_Avg_Passes)
{
    for (spot_num=0; spot_num<num_vals; spot_num++)
    {
        /* use averaged multiplier values to estimate noise levels */
        temp_float = spot_ofs[spot_num];
        depth_floor = floor(temp_float);
        lower_mult = 1.0 - (temp_float - depth_floor);
        upper_mult = temp_float - depth_floor;

        /* cof_sum was averaged above in multiplier value output routine */
        avg_mult = cof_sum[spot_num];
        for (i=0; i<Depth_Slices; i++)
        {
            if ((depth_floor+i >= 0) && (depth_floor+i+1 < Depth_Slices))
            {
                temp_float = (lower_mult*Diff_Coeffs[spot_num][depth_floor+i]+
                              upper_mult*Diff_Coeffs[spot_num][depth_floor+i+1]);
            }
        }
    }
}

/* difference based on standard curve */
curv_diff = cal_curve[i] - temp_float / avg_mult;

/* curv_diff has offset value - develop statistics */
nse_spot[spot_num][0] += curv_diff;
nse_spot[spot_num][1] += curv_diff * curv_diff;

nse_depth[i][0] +=;
nse_depth[i][1] += curv_diff;
nse_depth[i][2] += curv_diff * curv_diff;

/* difference based on estimated curve */
curv_diff = cal_curve_est[i] - temp_float / avg_mult;

nse_est[i][0] +=;
nse_est[i][1] += curv_diff;
nse_est[i][2] += curv_diff * curv_diff;

}

/* analyze noise statistics */
for (i=0; i<Depth_Slices; i++)
{
    if (nse_depth[i][0] > 0)
    {
        nse_depth[i][1] /= nse_depth[i][0];
nse_depth[i][2] /= nse_depth[i][0];
nse_est[i][1] /= nse_depth[i][0];
nse_est[i][2] /= nse_depth[i][0];

        nse_depth[i][2] = sqrt( nse_depth[i][2] - nse_depth[i][1] * nse_depth[i][1] );
nse_est[i][2] = sqrt( nse_est[i][2] - nse_est[i][1] * nse_est[i][1] );
    }
}

/* output noise results to file */
for (i=0; i<Depth_Slices; i++) fprintf( cal_file,
"%d: %6.2f %6.2f %5.0f Act: %6.2f %6.2f Est: %6.2f %6.2f\n",
i, cal_curve[i], cal_curve_est[i], nse_depth[i][0], nse_depth[i][1],
nse_depth[i][2], nse_est[i][1], nse_est[i][2] );

/* output pass results to file */
for (i=0; i<Reg_Size*Reg_Size; i++) matrix_d[i] = 0.0;
for (i=0; i<Reg_Size; j++) matrix_vect[i] = 0.0;

if (depth_pass < Dep_Avg_Passes)
{
    /* calculate regression values */
    for (j=0; j<Reg_Size*Reg_Size; j++) matrix_d[j] = 0.0;
    for (j=0; j<Reg_Size; j++) matrix_vect[j] = 0.0;

    temp_ptr = root;
    while (temp_ptr->next != NULL)
    {
        /* calculate coefficient vector */
        spot_num = temp_ptr->spot_num;
xval = (float) temp_ptr->x + Side_Pixels / 2;
yval = (float) temp_ptr->y + Side_Pixels / 2;
temp_vect[0] = xval * xval;

        /* calculate regression values */
        for (j=0; j<Reg_Size*Reg_Size; j++) matrix_d[j] = 0.0;
        for (j=0; j<Reg_Size; j++) matrix_vect[j] = 0.0;

        temp_ptr = temp_ptr->next;
    }
}
temp_vect[1] = yval * yval;
temp_vect[2] = xval;
temp_vect[3] = yval;
temp_vect[4] = xval * yval;
temp_vect[5] = 1;
for (i=0; i<Reg_Size; i++)
    for (j=0; j<Reg_Size; j++)
        matrix_el[i*Reg_Size+j] += temp_vect[i]*temp_vect[j];
for (i=0; i<Reg_Size; i++)
    matrix_vect[i] += (spot_ofs_avg[spot_num]/(depth_pass+1))*temp_vect[i];
temp_ptr = temp_ptr->next;

/* solve matrix problem and get regression values */
if (matrix_inv( Reg_Size, matrix_el, matrix_vect ))
    fprintf( reg_file, "\nInsufficient Rank\n" );
fprintf( reg_file, "\nEquation \%fx'2 + \%fy'2 + \%fx + \%fy + \%fxy + \%f\n", matrix_vect[0],
    matrix_vect[1], matrix_vect[2], matrix_vect[3], matrix_vect[4], matrix_vect[5] );

/* calculate rms error with subtracted regression curve */
temp_ptr = root;
for (i=0; i<5; i++) temp_vect[i] = 0.0;
while (temp_ptr->next != NULL)
{
    /* get ideal offset */
    temp_float = (matrix_vect[0] * (temp_ptr->x + Side_Pixels/2) + matrix_vect[2]) *
                 (temp_ptr->x + Side_Pixels/2) + (matrix_vect[1] * (temp_ptr->y + Side_Pixels/2) +
                 matrix_vect[3]) * (temp_ptr->y + Side_Pixels/2) + matrix_vect[4] * (temp_ptr->x +
                 Side_Pixels/2) * (temp_ptr->y + Side_Pixels/2) + matrix_vect[5];

    /* calculate offset & collect statistics */
    i = temp_ptr->spot_num;
    temp_float = temp_float - spot_ofs[i];
    temp_vect[0] += temp_float;
    temp_vect[1] += temp_float * temp_float;
    temp_vect[2] += spot_ofs[i];
    temp_vect[3] += spot_ofs[i] * spot_ofs[i];
    temp_ptr = temp_ptr->next;
}
for (j=0; j<5; j++) temp_vect[j] /= num_vals;
fprintf( reg_file, "Regression offset mean: \%f\n", temp_vect[0] );
fprintf( reg_file, "Regression offset std. dev.: \%f\n",
    (float) sqrt( temp_vect[1] - temp_vect[0]*temp_vect[0] ) );

fprintf( reg_file, "Non regr. offset mean: \%f\n", temp_vect[2] );
fprintf( reg_file, "Non regr. offset std. dev.: \%f\n\n",
}

/* end looping over depth passes */
}
/* free allocated memory */
free( Image_Mem );
free( Image_Norm );
deallocate_spot_list();

/* close files */
fdose( regr_file );
fdose( multi_file );
fdose( cal_file );
fdose( ofis_file );
exit( 0 );
}

/* the following routine accepts an image segment and normalizes it */
void Image_Normalize( int * Image_Data_0, int * Image_Data_1, float * Output_Data, int horz_offset, int vert_offset, int trans_size )
{
    float sum;
    int array_offset, norm_offset, i, j;

    sum = 0.0;

    /* Calculate average */
    j = 0;
    while ( vert_offset + j < trans_size )
    {
        array_offset = ( vert_offset + j ) * GRABBER_WIDTH + horz_offset;
        for ( i=0; i<trans_size; i++)
            sum += (float) Image_Data_0[array_offset++];
        j++;
    }
    while ( j < trans_size )
    {
        array_offset = ( vert_offset - trans_size + j ) * GRABBER_WIDTH + horz_offset;
        for ( i=0; i<trans_size; i++)
            sum += (float) Image_Data_1[array_offset++];
        j++;
    }
    sum /= (float) trans_size * (float) trans_size;

    /* Normalize to Output */
    j = 0;
    while ( vert_offset + j < trans_size )
    {
        array_offset = ( vert_offset + j ) * GRABBER_WIDTH + horz_offset;
        norm_offset = j*trans_size;
        for ( i=0; i<trans_size; i++)
            Output_Data[norm_offset++] = (float) Image_Data_0[array_offset++] / sum;
        j++;
    }
    while ( j < trans_size )
    {
        array_offset = ( vert_offset - trans_size + j ) * GRABBER_WIDTH + horz_offset;
        norm_offset = j*trans_size;
        for ( i=0; i<trans_size; i++)
            Output_Data[norm_offset++] = (float) Image_Data_0[array_offset++] / sum;
        j++;
    }
}
Output_Data[norm_offset++] = (float) Image_Data[array_offset++] / sum;

} 750
}
/ * This program generates low pixel pixel statistics for the CCD camera being used.

Setup 1:
This program can generate the following:
- total average pixel intensity
- total average pixel variance about this intensity (report std. dev)
- histogram of pixel intensities for one frame
- histogram of pixel variances over time (report std. dev)
- pixel correction map
- transform image weighting map

Assumptions:
Camera is uniformly illuminated in some meaningful way. Possible tests:
1. Dark (cover camera)
2. Low intensity
3. High intensity
If lighting is not uniform, then spatial data (and pixel correction map) is useless; variance histogram and transform weightings remain meaningful.

Setup 2:
Generate variance map given any scenes (illumination can vary)
This setup was used to generate CCD data presented in thesis.
*/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include "new3d.h"

/* which program to run - setup 1 or setup 2 */
/* #define setup1 */
#define setup2

#ifdef setup1
#define pix_hist_out "pix_hist.dat"
#define pix_var_out "pix_var.dat"

#define number_frames 20
#define number_lines 20

#define hist_length 40

unsigned char temp_array[GRABBER_WIDTH];
long pix_hist[256];
long var_hist[hist_length];

void main()
{

FILE *out_handle;
float pix_sum, pix_sq;
float max_std_dev, std_dev_mult;

long array_length;
float *frame_sum, *frame_sq;

unsigned char datum, *frame_buf;
int i, j, k, l;
/* disable lamp — assume our lamp not used for illumination */
lamp(0);

/* waste first frame to set things up */
discam();
capt0();
for (i=0; i<GRABBER_HEIGHT; i++) get_band( i, 1, temp_array );

/* calculate spatial mean, and std dev */
capt0();
pix_sum = 0.0;
pix_sq = 0.0;
for (i=0; i<GRABBER_HEIGHT; i++)
{
    get_band(i, 1, temp_array );
    for (j=0; j<GRABBER_WIDTH; j++)
    {
        datum = temp_array[j];
        pix_sum += (float) datum;
        pix_sq += (float) (datum) * (float) (datum);
    }
}
pix_sum /= (float) GRABBER_WIDTH * GRABBER_HEIGHT;
pix_sq /= (float) GRABBER_WIDTH * GRABBER_HEIGHT;

/* report these results */
printf( "\nAverage pixel intensity \%f.\n", (float) pix_sum );
printf( "Std. Dev. (averaged variance) \%f.\n\n", (float) sqrt( pix_sq - pix_sum*pix_sum ) );

/* allocate memory */
array_length = GRABBER_WIDTH * number_lines;
frame_buf = malloc( array_length * sizeof( unsigned char ) );
frame_sum = malloc( array_length * sizeof( float ) );
frame_sq = malloc( array_length * sizeof( float ) );
if (frame_sq == NULL)
{
    if (frame_sum != NULL)
        free( frame_sum );
    if (frame_buf != NULL)
        free( frame_buf );
    printf( "Insufficient Memory.\n\n" );
    exit(1);
}

/* clear histograms */
for (i=0; i<256; i++) pix_hist[i] = 0;
for (i=0; i<hist_length; i++) var_hist[i] = 0;

/* work our way down frame */
max_std_dev = 0.0;
for (i=0; i<GRABBER_HEIGHT; i+=number_lines)
{
    gotoxy( 1, wherey() );
    printf( "Line section \%d", i );
}

/* clear statistics */
for (j=0; j<GRABBER_WIDTH*number_lines; j++)
    frame_sum[j] = frame_sq[j] = 0;

/* grab requested number of frames */
for (j=0; j<number_frames; j++)
{
    capt0();
    for (k=0; k<GRABBER_HEIGHT; k+=number_lines)
    {
        get_band( k, number_lines, frame_buf );
        if (k==0)
        {
            /* this is section of interest */
            for (l=0; l<array_length; l++)
            {
                pix_hist[frame_buf[l]]++;  
                frame_sum[l] += (float) frame_buf[l];  
                frame_sq[l] += (float) frame_buf[l] * (float) frame_buf[l];
            }
        }
    }
}

/* calculate variances and convert to std. dev. */
for (l=0; l<array_length; l++)
{
    frame_sum[l] /= number_frames;
    frame_sq[l] /= number_frames;
    frame_sq[l] -= frame_sum[l] * frame_sum[l];
    frame_sq[l] = (float) sqrt( frame_sq[l] );
}

/* if first pass, guess at end limit */
if (i==0)
{
    for (l=0; l<array_length; l++)
    {
        if (frame_sq[l] > max_std_dev)
        {
            max_std_dev = frame_sq[l];
            max_std_dev *= 2;
            std_dev_mult = hist_length / max_std_dev;
        }
    }

    /* include variances in histogram */
    for (l=0; l<array_length; l++)
    {
        var_hist[(int) (frame_sq[l]*std_dev_mult)]++; 
    }
}

/* output pixel histogram */
out_handle = fopen( pix_hist_out, "w" );
if (out_handle != NULL)
{
    for (i=0; i<256; i++)
    {
        fprintf( out_handle, \%d \%d\n", i, pix_hist[i] );
        fclose( out_handle );
    }
}
Cal_Pixel.C

/* output variance histogram */
out_handle = fopen( pix_var_out, "w" );
if (out_handle != NULL)
{
    for (i=0; i<hist_length; i++)
        fprintf( out_handle, "%f \%1d\n", (float) i/std_dev_mult, var_hist[i] );
fclose( out_handle );
}

/* release memory */
free( frame_buff );
free( frame_sum );
free( frame_sq );

exit( 0 );
#endif

#ifndef setup2
#define num_frames 10   /* how many frames to grab each time */
#define num_lines 60   /* how many lines per buffer */
#define rate_of_change

main()
{
    FILE  *handle;
    float variances[256];
    float sum, sumsq, temp_float;
    long  count[256];
    long  i,j;
    int   index, done, lamp_level, line;
    unsigned char *frame_data[num_frames];
    unsigned char temp_data[GRABBER_WIDTH];

    for (i=0; i<256; i++)
    {
        count[i] = 0;
        variances[i] = 0.0;
    }
    for (i=0; i<num_frames; i++)
        frame_data[i] = malloc( num_lines * GRABBER_WIDTH );

    /* waste first frame to set things up */
dis0();
discam();
capt0();
    for (i=0; i<GRABBER_HEIGHT; i++)
        get_band( i, 1, temp_data );

done = 0;
    while (done == 0)
    {
        discam();
        printf( "Lamp Level (-1 to quit): " );
        scanf( "%d", &lamp_level );
        if (lamp_level < 0) { done = 1; }
        else

{
    lamp( lamp_level );
    line = 0;
    while (line < GRABBER_HEIGHT)
    {
        /* collect data */
        for (i=0; i<num_frames; i++)
        {
            cap0();
            delay(70);

            for (j=0; j<line; j++)
            {
                get_band( j, 1, temp_data );
                get_band( line, num_lines, frame_data[i] );
            }

            for (j=line+num_lines; j<GRABBER_HEIGHT; j++)
            {
                get_band( j, 1, temp_data );
                discam();
            }
        }

        /* calculate statistics */
        for (i=0; i<GRABBER_WIDTH*num_lines; i++)
        {
            sum = sumsq = 0;
            for (j=0; j<num_frames; j++)
            {
                temp_float = (float) frame_data[j][i];
                sum += temp_float;
                sumsq += temp_float * temp_float;
            }
            index = (int) floor( 0.5 + sum / num_frames );
            count[index]++;
            variances[index] += sum / num_frames -
                (sum / num_frames) * (sum / num_frames);
        }
        line += num_lines;
    }
}

/* output statistics */
handle = fopen( "ccd.dat", "w" );
for (i=0; i<256; i++)
    if (count[i] == 0) { fprintf( handle, "%ld %ld %10.5f\n", i, (long) 0l, (float) 0.0 ); } else fprintf( handle, "%ld %ld %10.5f\n", i, count[i], (float) sqrt( variances[i] / count[i] ) );
fclose( handle );
exit(0);

#endif
/ * This program generates a map of screen positions vs. stepper motor positions. A variety of graphics commands are used – these are not required for a minimal implementation of this program.

Assumptions:
1. Camera and stepper motors enabled
2. Pattern removed
3. Center of screen has obvious bright spot
*/

#include <stdio.h>
#include <ctype.h>
#include <malloc.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>
#include "new3d.h"

#define screen_cal_out "cal_scrn.out"
#define lamp_level 180
#define motor_steps 5L /* number of steps per measurement */
#define motor_backlash 15L
#define screen_width 400 /* approximate # microns across screen */
#define edge_amplitude 0.5 /* definition of edge of spot */
#define rough_tests 3 /* 1/2 tests on side of rough square */
#define fine_tests 2 /* 1/2 tests on side of fine square */

unsigned char temp_array[GRABBER_WIDTH];

void main()
{
 /* graphics variables */
 int xloc, yloc, step;
 int a, b;
 char entry;

 /* spot processing */
 unsigned char *left_ref, *temp_ref;
 int height_ref, ref_size, ref_offs, half_spot;
 int xmotor, ymotor, x, y, x_old, y_old;
 int step_pixels, step_x, step_y, left_step_x, left_y;
 int x_test, y_test;

 /* misc */
 FILE *out_handle;
 int i, j;

 void Find_Spot( int *, int *, int, unsigned char * );

 /* setup lamp and camera */
 lamp(lamp_level);
 discam();

 /* remove backlash */
 movel( Y_Axis, motor_backlash, Neg_Axis, Motor_Speed_y );
 movel( Y_Axis, motor_backlash, Pos_Axis, Motor_Speed_x );
 delay( Motor_Wait );
 movel( Y_Axis, motor_backlash, Pos_Axis, Motor_Speed_y );
move1( X_Axis, motor_backlash, Neg_Axis, Motor_Speed_x );

/* waste first frame to set up frame grabber */
capto();
discam();
for (i=0; i<GRABBER_HEIGHT; i++)
    get_band( i, 1, temp_array );

/* (optional) setup graphics */
a = VGA;
b = VGA_HI;
initgraph( &a, &b, "\borland\c\bg1" );

/* (optional) get frame and plot – find spot */
capto();
for (i=0; i<GRABBER_HEIGHT; i++)
{
    get_band( i, 1, temp_array );
    for (j=0; j<GRABBER_WIDTH; j++)
        putpixel( j, i, (temp_array[j]>>3) % 14+1 );
}

/* (optional) identify spot */
xloc = GRABBER_WIDTH/2; yloc = GRABBER_HEIGHT/2;
entry = ' ';
while (entry!='13')
{
    setwritemode(1);
    moveto( xloc - 10, yloc - 10 );
    lineto( xloc + 10, yloc - 10 );
    lineto( xloc + 10, yloc + 10 );
    lineto( xloc - 10, yloc + 10 );
    lineto( xloc - 10, yloc - 10 );
    entry = getch();
    lineto( xloc + 10, yloc - 10 );
    lineto( xloc + 10, yloc + 10 );
    lineto( xloc - 10, yloc + 10 );
    lineto( xloc - 10, yloc - 10 );
    step = isupper(entry) ? 3 : 1;
    entry = tolower(entry);
    if (entry == 'i') yloc-=step; if (entry == 'j') xloc-=step;
    if (entry == 'k') xloc+=step; if (entry == 'm') yloc+=step;
}
setwritemode(0);

/* size spot and get reference image */
get_band( yloc, 1, temp_array );
height_ref = ((int)temp_array[xloc-1]+(int)temp_array[xloc]+(int)temp_array[xloc+1])/3;
j = i = xloc;
while( temp_array[j] > height_ref * edge_amplitude ) i++;
while( temp_array[j] > height_ref * edge_amplitude ) j--;
ref_side = 3*(i-j) / 1;
ref_size = ref_side * ref_side;
ref_offs = ref_side/2;
half_spot = (i-j+1)/2;

temp_ref = malloc( ref_size * sizeof( unsigned char ) );
left_ref = malloc( ref_size * sizeof( unsigned char ) );
for (i=0; i<ref_side; i++)
{
    get_band( yloc-ref_offs+i, 1, temp_array );
    for (j=0; j<ref_side; j++)
temp_ref[j*ref_side+i] = temp_array[xloc-ref_ofs+j];
}

/* move to top of screen */
xmotor = 0;
ymotor = 0;

/* double step to roughly calibrate motor step size on screen */
movel( Y_Axis, 2L, Pos_Axis, Motor_Speed_y );
delay( Motor_Wait );
ymotor += 2;
x = xloc;
y = yloc - 2 * Motor_Step_Size * CAMERA_ASPECT *
        GRABBER_HEIGHT / screen_width;
Find_Spot( &x, &y, half_spot, temp_ref );
step_pixels = (yloc - y) * motor_steps / 2;

/* normal steps, update calibration */
while (y > 2*step_pixels)
{
    y_old = y;
    y -= step_pixels;
movel( Y_Axis, motor_steps, Pos_Axis, Motor_Speed_y );
delay( Motor_Wait );
ymotor += motor_steps;
Find_Spot( &x, &y, half_spot, temp_ref );
step_pixels = (step_pixels + (y_old - y)) / 2;
}
step_y = step_pixels;

/* move to left side — same idea as for vertical motion above */
/* double step */
movel( X_Axis, 2L, Neg_Axis, Motor_Speed_x );
delay( Motor_Wait );
xmotor -= 2;
xloc = x;
x = xloc - (2 * step_pixels) / motor_steps;
Find_Spot( &x, &y, half_spot, temp_ref );
step_pixels = (xloc - x) * motor_steps / 2;

/* normal steps */
while (x > 2*step_pixels)
{
    x_old = x;
    x -= step_pixels;
movel( X_Axis, motor_steps, Neg_Axis, Motor_Speed_x );
delay( Motor_Wait );
xmotor -= motor_steps;
Find_Spot( &x, &y, half_spot, temp_ref );
step_pixels = (step_pixels + (x_old - x)) / 2;
}
step_x = step_pixels;

/* remove backlash */
movel( Y_Axis, motor_backlash, Pos_Axis, Motor_Speed_y );
movel( X_Axis, motor_backlash, Neg_Axis, Motor_Speed_x );
delay( Motor_Wait );
movel( Y_Axis, motor_backlash, Neg_Axis, Motor_Speed_y );

120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
move1( X_Axis, motor_backlash, Pos_Axis, Motor_Speed_x );

/* main calibration section */
cleardevice();
setcolor( getmaxcolor() );

/* open output file */
out_handle = fopen( screen_cal_out, "w" );

/* save left side info */
left_step_x = step_x;
left_x = x;

/* can swap order of next two loops to change pattern of motion on screen */
/* loop over vertical positions */
y_test = 0;
y_odd = y - step_y;
while ( y < GRABBER_HEIGHT - 2*step_y )
{
/* move horizontally and gather points */
step_x = left_step_x;
x_test = 0;
x = left_x;
x_odd = x - step_x;

while ( x < GRABBER_WIDTH - 2*step_x )
{
/* find point and output position */
delay( Motor_Wait );
Find_Spot( &x, &y, half_spot, temp_ref );
fprintf( out_handle, "%d %d %d\n", x_test, y_test, x, y );
/* move to next location */
move1( X_Axis, motor_steps, Pos_Axis, Motor_Speed_x );
step_x = (step_x + (x - x_odd)) / 2;

/* adjust left step if this is first move */
if( x_test == 0 )
{
left_x = x;
left_step_x = (left_step_x + step_x) / 2;
step_y = (step_y + (y - y_odd)) / 2;
y_odd = y;
}
x_test++;

/* update estimate of x */
x_odd = x;
x += step_x;
}

/* move down a line and back to left side */
move1( Y_Axis, motor_steps, Neg_Axis, Motor_Speed_y );
ymotor += motor_steps;
move1( X_Axis, x_test*motor_steps + motor_backlash, Neg_Axis, Motor_Speed_x );
delay( Motor_Wait );
move1( X_Axis, motor_backlash, Pos_Axis, Motor_Speed_x );

/* update estimate of y */
y = y_odd + step_y;
```
y_test++;
}
ymotor -= y_test*motor_steps;

/* close file */
fclose(out_handle);

/* restore initial object position */
movel(X_Axis, labs(xmotor), xmotor > 0 ? Neg_Axis : Pos_Axis, Motor_Speed_X);
movel(Y_Axis, labs(ymotor), ymotor > 0 ? Neg_Axis : Pos_Axis, Motor_Speed_Y);

/* release memory */
free(temp_ref);
free(left_ref);
closegraph();
exit(0);
}

/* the following routine finds the spot given a guess and a pattern to match */
void Find_Spot(int *xposn, int *yposn, int ref_offset, unsigned char *reference)
{
  unsigned char *grabber_mem;

  long sum_val, max_val;
  int rough_edge, start_line, test_step, num_tests;
  int max_x, max_y, test, i, j, x, y;
  int array_x, array_y, grabber_pos, reference_pos;

  /* get memory */
  rough_edge = 2*(rough_tests+1)*ref_offset+1;
  grabber_mem = malloc(GRABBER_WIDTH * rough_edge);

  /* load frame */
  capt0();
  discam();
  start_line = *yposn - (rough_tests+1)*ref_offset;
  if(start_line < 0) start_line = 0;
  if(start_line + rough_edge > GRABBER_HEIGHT) start_line = GRABBER_HEIGHT - rough_edge;

  for(i=0; i<start_line; i++)
    get_band(i, 1, temp_array);
  get_band(start_line, rough_edge, grabber_mem);
  for(i=start_line+rough_edge; i<GRABBER_HEIGHT; i++)
    get_band(i, 1, temp_array);
  test_step = ref_offset / 2;
  num_tests = rough_tests;
  for(test=0; test<2; test++)
  {
    /* test = 0; rough search  test = 1: fine search */
    max_val = 0;
    max_x = max_y = 0;
    for(x=-num_tests; x<=num_tests; x++)
      for(y=-num_tests; y<=num_tests; y++)
      {
        sum_val = 0;
        array_x = *xposn + x*test_step - test_step;
        array_y = (*yposn + y*test_step - test_step - start_line);
        if(!((array_x < 0) || (array_x > GRABBER_WIDTH - 2*test_step)) ||
```
(array_y < 0) || (array_y > rough_edge))

  }
  grabber_pos = array_x + array_y * GRABBER_WIDTH;
  reference_pos = 0;

  /* find value at this point */
  for (j=0; j<2*test_step+1; j++)
  {
    for (i=0; i<(2*test_step+1); i++)
      sum_val += (long) reference[reference_pos++] *
                   (long) grabber_mem[grabber_pos++];
  }
  grabber_pos += (GRABBER_WIDTH-2*test_step-1);

  /* apply weighting function */
  if (test == 0) sum_val = (long) (sum_val * ((6.0*num_tests-abs(x)) /
                                     (6.0*num_tests)) * (6.0*num_tests-abs(y))/(6.0*num_tests));
  if (sum_val > max_val)
  {
    max_val = sum_val;
    max_x = x; max_y = y;
  }

  /* modify values for second iteration : fine test */
  *xpose += test_step * max_x;
  *ypose += test_step * max_y;
  test_step /= (fine_tests + 1);
  num_tests = fine_tests;
}

/* release memory */
free(grabber_mem);
/* This program determines the locations of pattern spots on the CCD array.

The program determines the spots from camera images and writes the x,y
coordinates to a disk file.

Assumptions:
  Camera enabled
  Material in fairly good focus (material must be reasonably reflective)
  Slide in position */

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <alloca.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>
#include "new3d.h"

#define lamp_level 155

/* The following describe the limits on spots centers in order for the spot to be output to file.
   Have limits exceed screen size to include all found spots.  Note that the
   limits below refer to spot centers, not nXn subimage corner */
#define left_limit  12  /* for 16X16 transform, avoid bad pixels */
#define right_limit GRABBER_WIDTH-6
#define upper_limit 6
#define lower_limit GRABBER_HEIGHT-6

/* structure element — used in linked list of spot location candidates */
typedef struct
{
    int x,y;
    float spot;
    void *prev;
} spot_rec;

char    temp_array[GRABBER_WIDTH];

/* on command line, usage:    cal_spot dx dy
   dx and dy are integers specifying the desired offset to be applied
   to the existing calibrated spot positions. */
void    main( int argc, char **argv )
{
    FILE *out_file;
    float *mask, prod, delta sq, offset;
    unsigned char *data[50], *data_ptr;
    spot_rec *root, *temp_ptr;
    void **old_ptr;
    int spot_sep, delta, spot_wid, spot_center;
    int pass, line_offset, start_line, end_line, line_dirc;
    int cur_x, cur_y, temp_x, temp_y, guess_x, guess_y;
    int i, j, k, pix, line, insert, shift;

    /* setup lamp and camera */
lamp(lamp_level);
discam();
/* waste first frame to set up frame grabber */
capt0();
for (i=0; i<GRABBER_HEIGHT; i++)
    get_hand( i, 1, temp_array );
discam();
/* (optional) — remove comments to just test current spot locations */
/* versus camera image. Useful to check calibration or to see whether */
/* the pattern has shifted over time. */
/* test_list( argc, argv[1], argv[2] );
exit(0);
*/
/* get rough spot separation in pixels */
printf( "Enter approximate spot separation in pixels: " );
scanf( "%d", &spot_sep );
delta = 2 * spot_sep / 3; /* min. separation for distinct spots */
delta_sq = (float) delta*delta;
spot_wid = spot_sep / 2; /* approximate spot diameter */
spot_wid += -(spot_wid % 2)+1; /* make value odd */
spot_center = spot_wid / 2;
/* setup root record in list */
root = malloc( sizeof( spot_rec ) );
root->x = -delta;
root->y = -delta;
root->spot = 0.0;
root->prev = NULL;
/* allocate memory */
mask = malloc( spot_wid * spot_wid * sizeof( float ) );
for (i=0; i<spot_wid; i++)
    data[i] = malloc( (GRABBER_WIDTH + spot_wid) * sizeof( char ) );
/* setup mask values */
for (i=0; i<spot_wid; i++)
    for (j=0; j<spot_wid; j++)
        mask[i+j*spot_wid] = exp(-sqrt(pow(i-spot_center,2)+
                                 pow(j-spot_center,2))/spot_center);
/* grab next screen */
capt0();
for (pass=0; pass<2; pass++)
{
    for (i=0; i<spot_wid; i++)
        memset( data[i], 0, (GRABBER_WIDTH + spot_wid) * sizeof( char ) );
    start_line = 0;
    end_line = GRABBER_HEIGHT + spot_center;
    line_dirm = 1;
    line_offset = 0;
    if (pass == 1)
    {
        start_line = GRABBER_HEIGHT + spot_wid;
        end_line = GRABBER_HEIGHT - spot_wid;
    }
line_dirn = -1;
line_offset = spot_wid-1;
}

/* loop down screen */
line = start_line;
do {
  /* shift previous lines in buffer */
data_ptr = data[0];
for (i=0; i<spot_wid-1; i++)
  data[i] = data[i+1];
data[spot_wid-1] = data_ptr;

/* read line from screen */
if (line - line_offset < GRABBER_HEIGHT)
  { get_band( line-line_offset, 1, data[spot_wid-1]+spot_center ); }
else
  { memset( data[spot_wid-1], 0, GRABBER_WIDTH ); }

/* work across screen */
for (pix=0; pix<GRABBER_WIDTH; pix++)
  {
    /* calculate mask dot product */
    prod = 0.0;
    for (i=0; i<spot_wid; i++)
      for (j=0; j<spot_wid; j++)
        prod += data[i][pix+j] * mask[i+spot_wid*];

    /* look through list, discard lower values, insert this if better */
    temp_ptr = root;
    old_ptr = &root;
    insert = 1;
    while ((temp_ptr != NULL) && (temp_ptr->prev != NULL) &&
           (temp_ptr->y > line_delta))
      {
        shift = 1;
        offset = ((float) pix-temp_ptr->x) * ((float) pix-temp_ptr->x) +
                  ((float) line-temp_ptr->y) * ((float) line-temp_ptr->y);
        if (offset <= delta_sq)
          {  
            if (prod <= temp_ptr->spot)
              { insert = 0; }
            else
              {  
                old_ptr = temp_ptr->prev;
                free( temp_ptr );
                temp_ptr = (spot_rec *) old_ptr;
                shift = 0;
              }
          }
        if (shift)
          { 
            old_ptr = &(temp_ptr->prev);
            temp_ptr = temp_ptr->prev;
          }
      }
    if (insert)
```c
{  
temp_ptr = malloc( sizeof( spot_rec ) );
if (temp_ptr == NULL)
{
    dump_list( root );
    exit(1);
}
temp_ptr->prev = root;
root = temp_ptr;
temp_ptr->x = pix;
temp_ptr->y = line;
temp_ptr->spot = prod;
}
line += line_dir;
gotoxy( 1, wherey() );
printf( "Done line %d", line );
} while (line != end_line);

/* output spot centers */
/* the output is sorted to make later use easier, this is a very bad
 sorting routine, it is not meant to be used in practice */
out_file = fopen( "spot.dat", "w" );
cur_x = 0;
cur_y = 0;
do
{
    guess_x = 0;
    guess_y = GRABBER_HEIGHT + spot_wid;
    temp_ptr = root;
    while (temp_ptr->prev != NULL)
    {
        temp_x = temp_ptr->x;
        temp_y = temp_ptr->y;
        if ((temp_y == cur_y) && (temp_x > cur_x))
        {
            guess_x = temp_x; guess_y = temp_y;
        }
        else if ((temp_y > cur_y) && (temp_y < guess_y))
        {
            guess_x = temp_x; guess_y = temp_y;
        }
        else if ((temp_y == cur_y) && (temp_x < guess_x) && (cur_y < guess_y))
        {
            guess_x = temp_x;
        }
        temp_ptr = temp_ptr->prev;
    }
    if (guess_y < GRABBER_HEIGHT + spot_wid)
    {
        temp_y = guess_y - spot_center;
        /* output value only if within desired bounds */
        if ((guess_x >= left_limit) && (guess_x <= right_limit) &&
            (temp_y >= upper_limit) && (temp_y <= lower_limit))
            fprintf( out_file, "%d %d\n", guess_x, guess_y - spot_center );
    }
    cur_x = guess_x;
    cur_y = guess_y;
} while (guess_y < GRABBER_HEIGHT + spot_wid);
fclose( out_file );
dump_list( root );
```
exit(0);

dump_list( spot_rec *root )
{
    spot_rec  *temp;

    do
    {
        temp = root->prev;
        free( root );
        root = temp;
    } while ( root != NULL );
}

test_list( int argc, char *arg1, char *arg2 )
{
    FILE  *in_file, *out_file;
    int   graphdriver, graphmode;
    int   dx, dy, i, j, bitcol;
    int   x, y, xe, ye;

    dx = dy = 0;
    if ( argc > 3 )
    {
        dx = atoi( arg1 ); dy = atoi( arg2 );
    }

    in_file = fopen( "spot.dat", "r" );
    if( in_file != NULL )
    {
        graphdriver = DETECT;
        initgraph( &graphdriver, &graphmode, "\borland\borl1\bgm" );

        /* plot camera image to graphics screen */
        for (i=0; i<GRABBER_HEIGHT; i++)
        {
            getband( i, 1, temp_array );
            for (j=0; j<GRABBER_WIDTH; j++)
                putpixel( j, i, (temp_array[j] /8) % 14 + 1 );
        }

        /* flash calibrated spot positions on screen */
        bitcol = 0;
        while ( fbhit() )
        {
            rewind( in_file );
            while ( !feof( in_file ) )
            {
                fscanf( in_file, "%d %d", &i, &j );
                if( !feof( in_file ) )
                    putpixel( i+dx, j+dy, bitcol );
            }
            bitcol = getmaxcolor() - bitcol;
            delay(1000);
        }
}
/ * special — if hit return modify spot file with used supplied shift */
if (i == 'r')
    if ((dx != 0) || (dy != 0))
        { rewind( in_file );
          out_file = fopen( "tempspot.dat", "w" );
          while (!feof( in_file ))
              { fscanf( in_file, "%d %d", &i, &j );
                if (!feof(in_file)) fprintf( out_file, "%d %d\n", i+dx, j+dy );
              }
          fclose( in_file );
          fclose( out_file );
          remove( "spot.dat" );
          rename( "tempspot.dat", "spot.dat" );
        }
    fclose( in_file );
closegraph();
}
Appendix B

Theory & Simulation Code Listings

Simulation programs in this appendix:  

<table>
<thead>
<tr>
<th>Program Name</th>
<th>Page #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray_Trace.C</td>
<td>163</td>
</tr>
<tr>
<td>Sim_Sampling.C</td>
<td>166</td>
</tr>
<tr>
<td>Sim_Spots.C</td>
<td>169</td>
</tr>
</tbody>
</table>
/* this program performs simple ray tracing through lenses — lenses have
height and thickness, no width */

#include <stdio.h>
#include <math.h>
define Pi 3.1415926535
define num_rays 2000 /* number of test rays through lens */

void line_sphere( float, float, float, float, float, float, int, float *);
void trace( float lens_height, float lens_distance, float radius1, float radius2,
      float point_offset, float screen_distance, int hist_points, int *yhist );

/* main program — dumps results to file "temp.dat". This header file just
calls the raytracing routine and writes output to file. */
main()
{
    FILE *handle;
    float dist, sum, diffs;
    int yhist[2000];
    int entry, i, j, k;

    handle = fopen("temp.dat", "w");
    dist = 0.011;
    entry = 0;
    for (dist=0.007; dist<0.015; dist+=0.00008)
    {
        for (j=0; j<2000; j++) yhist[j] = 0;
        trace( 0.005, dist, 0.015, 0.015, 0, 0.2, 200, yhist+800 );
        for (j=0; j<250; j++)
        {
            yhist[j] = 0;
            for (k=0; k<13; k++) yhist[j] += yhist[5*j+k];
        }
        for (i=140; i<220; i++) fprintf(handle,"%d ", yhist[i]);
        fprintf(handle,"\n ");
    }
    fclose( handle );
    exit(0);
}

/* ray tracing subroutine; all parameters are passed except for the index of
refraction of the lens (defined near start of function) */
void trace( float lens_height, float lens_distance, float radius1, float radius2,
      float point_offset, float screen_distance, int hist_points, int *yhist )
{
    float max_angle, min_angle, angle, slope;
    float xsplh, ysplh, alpha, temp1, temp2;
    float anglespl, th1, th2;

    int array_ofs;
    float ystep, yimage;
    float n_lens, n_air;
    int i;

    n_lens = 1.75; n_air = 1.0;

    for (i=0; i<hist_points; i++) yhist[i] = 0;
    array_ofs = 30 + hist_points / 2;
}
ystep = lens_height / (hist_points/2-1);

/* extreme beam angles to test — light ray must always pass through lens */
max_angle = atan2(0.99 * lens_height - point_offset, lens_distance);
min_angle = atan2(-0.99 * lens_height - point_offset, lens_distance);

temp1 = sqrt(radius1*radius1 - lens_height*lens_height);
temp2 = sqrt(radius2*radius2 - lens_height*lens_height);

/* loop over point source angles */
for (angle=min_angle; angle<=max_angle; angle+=(max_angle-min_angle)/num_rays)
{
    /* near side of lens */
slope = tan(angle);
line_sphere(-(lens_distance+temp1), point_offset, slope, 0, 0, radius1, -1, &alpha);
xsph = -(lens_distance+temp1) + alpha;
ysph = point_offset + slope * alpha;
angsph = atan(ysph / (xsph));
angsph = Pi/2 - angsph;
th1 = Pi/2 + angle - angsph;
th2 = asin(sin(th1) * n_air/n_lens);
slope = tan(th2 + angsph - Pi/2);

/* far side of lens */
line_sphere(xmph + temp1+temp2, ysph, slope, 0, 0, radius2, 1, &alpha);
xsph = xsph + temp1+temp2 + alpha;
ysph = ysph + slope * alpha;
angsph = atan(ysph / xsph);
th1 = angsph - atan(slope);
th2 = asin(sin(th1) * n_lens/n_air);
slope = tan(angsph - th2);

/* intersection with image plane */
xsph -= temp2;
yimage = ysph + (screen_distance - xsph) * slope;
yhist[(int)(array_ofs+yimage/ystep)]++;
}
return;

/* 2 dimensional line and circle intersection calculation function */
void line_sphere(float xline, float yline, float slope, float xsph,
    float ysph, float radius, int xsid, float *alpha)
{
    float a, b, c;

    /* transform coordinate system */
xline -= xsph; yline -= ysph;

    /* setup and solve quadratic */
a = 1 + slope*slope;
b = 2 * xline + 2 * yline * slope;
c = xline * xline + yline * yline - radius * radius;
if (xsid < 0) {
    *alpha = (-b - sqrt(b * b - 4 * a * c)) / (2 * a);
} else {
    *alpha = (-b + sqrt(b * b - 4 * a * c)) / (2 * a);
}
return;
}
/* this program simulates the effect of changing the depth sampling step. It uses the same estimator and calibration curve as the experimental program */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
#include <values.h>

float ML_Depth_Ofs[10] = {3, 3, 3, 4, 5, 6, 7, 9, 11, 13};
float ML_Depth_Step[10] = {0.0625, 0.0625, 0.0625, 0.125, 0.125, 0.125, 0.25, 0.25, 0.25, 0.25};

float random_gauss( float );

/* standard continuous time functions */
float sinc( float x )
{
  if (x != 0) { return( 13 * sin( x*1.9/7 ) / (x*1.9/7) ); } else return( 13.0 );
}
float actual( float x )
{
  if (x > 0) { return( 13 * exp( -(x*x/16)*log(2) ) ); }
  else return( 13 * exp( -(x*x/50)*log(2) ) );
}

main()
{
  #define num_tests 300

  FILE *handle, *handle2;
  float sample_step, ml_step, std_dev;
  float c1[200], c2[200];
  float of1, of2, temp;
  float self_prod, max_prod, temp_prod, mult, max_ofs;
  float max_ml, ml_pos, ml_ofs, ml_sum, max_pos, limult, hmult;
  float x, a1, a2, a3, y1, y2, y3, x2;
  float error, err_sum[20], err_sumsq[20], totvar;
  int tnum, i, j, k, t;

  float cal_curve[60];
  int spacing, offset, cur_depth, num_samples, depth_floor;
  float max_val, corr_sum, cal_sum, ml_min, ml_offset, depth_ofs;
  float lower_mult, upper_mult, temp_float;

  #define Cal_Depth_Ofs 3
  #define Depth_Slices 60

  handle = fopen( "cur.dat", "r" );
  for (i=0; i<60; i++) fscanf(handle, "%f %f", &temp, cal_curve+i);
  fclose(handle);

  /* evaluate integral of calibration curve */
  cal_sum = 0;
  for (i=0; i<Depth_Slices; i++)
    cal_sum += cal_curve[i] * cal_curve[i];

  handle = fopen( "output.dat", "w" );
  fprintf(handle,"%n");
  ml_step = 0.0625;
sample_step = 1;

/* spacing limits to test — a spacing of one means 2.5 micron sampling */
#define Fine_Spacing 1
#define Coarse_Spacing 10
for (std_dev=0; std_dev<=0.51; std_dev+=0.1)
{
  for (i=Fine_Spacing; i<Coarse_Spacing; i++)
    err_sum[i] = err_sumsq[i] = 0.0;

  for (tnum=0; tnum<num_tests; tnum++)
    {
      of2 = random( 10000 ) / 10000.0;
      mult = fabs(1.0 + random_gauss( 0.1 ));
      for (j=0; j<=Depth_Slices; j++)
        {
          temp = (j - Depth_Slices/2) * sample_step + 1.0/20000.0;
          c2[j] = mult * actual( temp + sample_step * of2 );
          c2[j] += random_gauss( std_dev );
        }

    /* all code within this loop is taken directly from experimental program */
    for (spacing=Fine_Spacing; spacing<Coarse_Spacing; spacing++)
    {
      offset = random( spacing );

      /* find approximate offset using correlator */
      max_val = 0.0;
      max_pos = -2*Cal_Depth_Ofs;

      /* find optimal depth and multiplier value */
      for (cur_depth = offset - Cal_Depth_Ofs;
           cur_depth <= offset + Cal_Depth_Ofs; cur_depth++)
        {
          corr_sum = 0.0;

          j = 0;
          for (i=0; i<Depth_Slices; i++)
            if ((cur_depth + i)>=0) && (cur_depth+i<Depth_Slices) &&
                (i+1+offset<Depth_Slices))
              {
                j++;
                corr_sum += cal_curve[cur_depth+i] * c2[i+offset];
              }
          if (corr_sum > max_val)
            {
              max_val = corr_sum;
              max_pos = cur_depth;
            }
          max_val /= (j * cal_sum / Depth_Slices);

      /* find better estimate using maximum likelihood */
      ml_min = 1e20;
      ml_offset = 0.0;
      for (depth_ofs = -ML_Depth_Ofs[spacing]; depth_ofs <= ML_Depth_Ofs[spacing];
           depth_ofs += ML_Depth_Step[spacing])
        {

    60
    70
    80
    90
    100
    110

167
ml_sum = 0.0;

num_samples = Depth_Slices / spacing;

depth_floor = floor( depth_offs );
lower_mult = 1 - (depth_offs - depth_floor);
upper_mult = 1 - lower_mult;

for (i=0; i<Depth_Slices; i+=spacing)
  if (num_samples > 0)
    if (i+offset < Depth_Slices)
      if ((i+max_pos+depth_floor >= 0) &&
          (i+max_pos+depth_floor+1<Depth_Slices))
        { temp_float = max_val * (lower_mult *
          cal_curve[i+max_pos+depth_floor] +
          upper_mult * cal_curve[i+max_pos+depth_floor+1]); }
      else { temp_float = max_val * cal_curve[0]; }
      temp_float -= c2[offset+i];
      ml_sum += temp_float * temp_float;
      num_samples--;
    }
  if (ml_sum < ml_min)
    { ml_min = ml_sum;
      ml_offset = depth_offs;
    }

max_pos = max_pos + ml_offset - offset;

/* update error */
error = max_pos - of2;
err_sum[spacing] += error;
err_sumsq[spacing] += error * error;
}

for (i=FineSpacing; i<CoarseSpacing; i++)
  { err_sum[i] /= num_tests;
    err_sumsq[i] /= num_tests;
    err_sumsq[i] = sqrt( fabs(err_sumsq[i] - err_sum[i] * err_sum[i]) );
    fprintf( handle, "%d %f %f
", i, std_dev, err_sumsq[i] );
  }

fclose(handle);
exit(0);
/* the program simulates the effects of spot differences. A standard
calibration curve is used, to which are added sinusoidal variations
having magnitudes based on empirical data from program Cal_Curve.C */

#include <stdlib.h>
#include <math.h>
#include <values.h>

float random_gauss( float );

/* standard continuous time functions */
float sinc( float x )
{ if (x ! = 0) { return ( 13 * sin( x*1.9/7 ) / (x*1.9/7) ); } else return ( 13.0 ); }
float actual( float x )
{ if (x > 0) { return ( 13 * exp( -(x*x/50)*log2 ) ); } else return ( 13 * exp( -(x*x/16)*log2 ) ); } 

main()
{

#define num_tests 30 /* number of simulation on given deviation */
#define num_runs 120 /* number of deviations per data point */
#define side_samples 30 /* number of data points / 2 */

float sample_step, m1_step, std_dev;
float cl[200], c2[200];
float off1, of2, temp;
float self_prod, max_prod, temp_wd, mult, max_ofs;
float max_m1, m1_pos, m1_ofs, m1_sum, max_pos, lmult, lmult;
float x;
float s1, s2, s3, sin_ph, sin_amp, sin_per;
float error, err_sum, err_sumsq, totvar;
int i, j, k, t;

printf("\n");
sample_step = 1.0;
m1_step = 0.125;
std_dev = 0.0;
for (std_dev=0; std_dev<0.55; std_dev+=0.05)
{
totvar = 0;
for(t=0;t<num_runs; t++)
{ /* set second order noise dependence values (empirically) */
    s1 = 0.5;
    s2 = (1.5/81-0.5/169)/(1/9.0-1/13.0);
    s3 = (1.5 - 9*s2)/81;
    sin_ph = 2 * 3.14159 * random( 360 ) / 360.0;
    sin_amp = random_gauss( 1 );
    sin_per = 8.0 + 16.0 * random( 100 ) / 100.0;
    err_sum = err_sumsq = 0.0;
    for (i=0; i<num_tests; i++)
    { of1 = random( 10000 ) / 10000.0;
      of2 = random( 10000 ) / 10000.0;


169
/ * supply gaussian multiplier */
mult = 1 + random_gauss( 0.2 ); if (mult < 0.2) mult = 0.2;
for (j=0; j<=2*side_samples; j++)
{
    temp = (j - side_samples) * sample_step + 1.0/20000.0;
    c1[j] = mult * actual( temp + sample_step * off1 );
    c2[j] = actual( temp + sample_step * of2 );
    c2[j] += ((s3*c2[j]+s2)*c2[j]+s1)*sin_amp*
        sin(sin_ph+(temp+sample_step*of2)*6.28/sin_per);
    c2[j] += random_gauss( std_dev );
}

/* correlator to determine multiplier */
self_prod = 0;
for (j=1; j<2*side_samples; j++) self_prod += c1[j]*c1[j];
max_prod = 0;
for (j=-1; j<1; j++)
{
    temp_prod = 0;
    for (k=1; k<2*side_samples; k++)
        temp_prod += c1[k] * c2[k+j];
    if (temp_prod > max_prod)
        { max_prod = temp_prod; max_ofs = j; }
}
mult = self_prod / max_prod;

/* ML sub sample estimator */
max_ml = MAXINT;
for (ml_pos=-2; ml_pos<2; ml_pos+=ml_step)
{
    ml_sum = 0;
    ml_ofs = floor( ml_pos );
    lmult = 1 - (ml_pos - ml_ofs);
    lmult = 1 - lmult;
    for (k=3; k<2*side_samples-3; k++)
        ml_sum += pow( c1[k] - mult * (lmult*c2[k+max_ofs+ml_ofs]+ml_ofs) +
                lmult*c2[k+max_ofs+ml_ofs+1]), 2.0 );
    if (ml_sum < max_ml)
        { max_ml = ml_sum; max_pos = ml_pos; }
}
/* update error */
error = max_ofs + max_pos - of1 + of2;
err_sum += error;
err_sumsq += error * error;
err_sum /= num_tests;
err_sumsq /= num_tests;
err_sumsq = sqrt( fabs(err_sumsq - err_sum * err_sum) );
totvar += err_sumsq * err_sumsq;
printf( "%f  %f\n", std_dev, (float) sqrt( totvar/num_runs ) );
ext(0);
}
Bibliography


Optics References


171


Object Profiling References


172


Part i: Stylus Instruments, Electron Microscopy, & Non Optical Comparators
Part ii: Optical Instruments
Wear, Vol. 125, pp. 271–308, 1988


Material Surface Properties References


Estimation Theory References


Image Processing References


A Brief Biography

Paul W. Fieguth was born in Winnipeg, Manitoba, Canada on August 5, 1968, but left for Ontario within a few weeks of birth.

He grew up in Mississauga, a suburb west of Toronto, Ontario. He obtained his secondary and high school education at University of Toronto Schools (UTS), graduating in 1986.

He attended the University of Waterloo in a five year cooperative education program. The co-op program permits students to acquire work experience while at school; under this program Paul worked for Ontario Hydro — Pickering, Ontario Hydro — NPD, Seastar Instruments, MacDonald-Dettwiler & Assoc., and the University of Waterloo VLSI lab. He obtained an Honours BSc. in Electrical Engineering with an option in Physics in 1991. Awards include the University of Waterloo Sanford Fleming medal for Academic Achievement, the Association of Professional Engineers of Ontario Gold Medal, and a NSERC 1967 Graduate Fellowship (which helped to fund this research).


He is a student member of IEEE, and an associate member of Sigma Chi.