MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Technical Report No. 1498 August, 1994

Computing 3-D Motion in Custom Analog and Digital VLSI

by

Lisa G. Dron

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

This thesis examines a complete design framework for a real-time, autonomous system with specialized VLSI hardware for computing 3-D camera motion. In the proposed architecture, the first step is to determine point correspondences between two images. Two processors, a CCD array edge detector and a mixed analog/digital binary block correlator, are proposed for this task. The report is divided into three parts. Part I covers the algorithmic analysis; part II describes the design and test of a 32×32 CCD edge detector fabricated through MOSIS; and part III compares the design of the mixed analog/digital correlator to a fully digital implementation.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology in collaboration with the Analog VLSI for Machine Vision research project. Support for the Laboratory's research is provided in part by the ARPA contract N00014-85-K-0124. Funding for the Analog VLSI for Machine Vision research project is provided by NSF Grant MIP-91-17724 and NSF and ARPA contracts MIP-88-14612. The author has been supported by a GRPW fellowship from AT&T Bell Laboratories.

Computing 3-D Motion in Custom Analog and Digital VLSI

by

Lisa G. Dron

S.B., Massachusetts Institute of Technology (1976)
M.S.E., University of Texas at Austin (1989)

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

Doctor of Philosophy

at the

Massachusetts Institute of Technology

September 1994

Signature of Author____
Department of Electrical Engineering and Computer Science
August 9, 1994

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

Accepted by____
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Computing 3-D Motion in Custom Analog and Digital VLSI

by

Lisa G. Dron

Submitted to the Department of Electrical Engineering and Computer Science
on August 9, 1994, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Computing the general 3-D motion of a camera from the changes in its images as it moves through the environment is an important problem in autonomous navigation and in machine vision in general. The goal of this research has been to establish the complete design framework for a small, autonomous system with specialized analog and digital VLSI hardware for computing 3-D camera motion in real-time. Such a system would be suitable for mounting on mobile or remote platforms that cannot be tethered to a computer and for which the size, weight and power consumption of the components are critical factors.

Combining algorithmic design with circuit design is essential for building a robust size- and power-efficient system, as there are constraints imposed both by technology and by the nature of the problem which can be more efficiently satisfied jointly. The first part of this thesis is thus devoted to the analysis and development of the algorithms used in the system and implemented by the special processors. Among the major theoretical results presented in Part I are the development of the multi-scale veto edge detection algorithm, the derivation of a simplified method for solving the motion equations, and a complete analysis of the effects of measurement errors on the reliability of the motion estimates.

In the proposed system architecture, the first step is to determine point correspondences between two successive images. Two specialized processors, the first a CCD array edge detector implementing the multi-scale veto algorithm, and the second a mixed analog/digital binary block correlator, are proposed and designed for this task. A prototype CCD edge detector was fabricated through MOSIS, and based on the test results from this chip, improvements are suggested so that a full-size focal-plane processor can be built. The design of the mixed analog/digital correlator is compared with a fully digital implementation and is seen to yield a significant reduction in silicon area without compromising operating speed. In the conclusions, the theoretical and experimental results from the different parts of this thesis are combined into a single design proposal for the complete motion system.

Thesis Supervisor: Berthold K. P. Horn

Title: Professor of Computer Science and Engineering

Acknowledgments

My years at MIT would not have been as meaningful without the help, support, and advice of many people. I would first like to thank my thesis advisor Berthold Horn for the complete freedom he gave me to pursue the research I was interested in, and I am grateful to my other committee members, Charlie Sodini and Jim Roberge, for their time spent in reading this thesis and for their helpful comments along the way.

I wish to give special thanks to the other present and former members of the Analog VLSI for Machine Vision group: Ig McQuirk, Steve Decker, Chris Umminger, Craig Keast, Mikko Hakkarainen, and John Harris who provided much needed technical advice and help. Special thanks also goes to my graduate counselor, Jacob White, for years of helpful advice that was never judgemental or critical—even when he was telling me I was wrong.

I certainly could have never gotten as much done without the help of three special support staff members of the AI Lab. Ron Wiken was always there to help me find or build everything I needed. Bruce Walton kept the computers running, and regardless of how much direct help he gave me on my thesis, he deserves to be acknowledged as the friendliest and most patient system manager I have ever known in my life. Laurel Simmons facilitated everything, from bringing in electricians when I needed them, to soldering the bulkhead connectors in the Vision Lab, to letting me take home all of my equipment this past year so that I could spend more time with my family and still graduate. In addition to these three, Eric Grimson patiently supported everything I did, from Vision Night 92, to letting me set up the Vision Lab the way I wanted to.

I owe a tremendous debt to AT&T Bell Labs for supporting me under their GRPW fellowship program for the last five years. Almost as important as the financial support was the real personal support I received from my mentor, Behrokh Samadi, and from Dexter Johnston. I also owe a great deal to Chuck McCallum for sponsoring my candidacy to the program. I only hope he is not still disappointed that I didn't go into Operations Research.

Finally, my deepest appreciation goes to those who are closest to me: to my sons, David and William, who have now spent half their lives watching their mother go through graduate school; and to Michael, whose love and support made everything possible.

This thesis describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology in collaboration with the Analog VLSI for Machine Vision research project. Support for the Laboratory's research is provided in part by the ARPA contract N00014-85-K-0124. Funding for the Analog VLSI for Machine Vision research project is provided by NSF Grant MIP-91-17724 and NSF and ARPA contracts MIP-88-14612. The author has been supported by a GRPW fellowship from AT&T Bell Laboratories.

To all my guys: Michael, David, and William

Contents

1 Introduction **20**

I Algorithms and Theory **23**

2 Methods for Computing Motion and Structure **24**

2.1 Basic Equations 25

2.1.1 Perspective geometry 25

2.1.2 The epipolar constraint 27

2.2 Computing Motion by Matching Features 29

2.2.1 Representing rotation 30

2.2.2 Exact solution of the motion equations 32

2.2.3 Least-squares methods 34

2.3 Correspondenceless Methods 37

2.3.1 Optical flow 38

2.3.2 Direct methods 39

2.3.3 Limitations of correspondenceless methods 40

2.4 Motion Vision and VLSI 45

3 Matching Points in Images **48**

3.1 Brightness-Based Methods 48

3.2 Gray-level Correlation Techniques 50

3.3 Edge-Based Methods 54

4 System Architecture **59**

5 An Algorithm for an Analog Edge Detector **62**
5.1 The Multi-Scale Veto Rule 63
5.2 Other Methods and the Use of Multiple Scales 66
5.3 Circuit Models 68
5.4 Choosing the Parameters 72
5.5 Simulations 75

6 The Matching Procedure **79**
6.1 Finding the Best Match 79
6.2 Other Tests 84
6.3 Simulations 85

7 Solving the Motion Equations **89**
7.1 The Simplified Algorithm 90
7.1.1 Pure translation or known rotation 91
7.1.2 Rotation with known translation 91
7.1.3 Pure rotation ($|\mathbf{b}| = 0$) 93
7.1.4 The complete algorithm 95
7.2 Ambiguities and Multiple Solutions 96
7.3 Simulations 99

8 The Effects of Measurement Errors **104**
8.1 Numerical Stability 105
8.1.1 Eigenvalues and eigenvectors of $\mathbf{C}$ 106
8.1.2 Condition of the matrix $\mathbf{A}$ 109
8.1.3 Minimizers of S 113
8.2 Error Analysis 120
8.2.1 First-order error in the baseline (rotation known) 122
8.2.2 First-order error in the rotation (baseline known) 128
8.2.3 Coupling between estimation errors 134
8.3 Experimental Verification 138

9 Summary of Part I **150**

II Design of a CCD-CMOS Multi-Scale Veto Chip **153**

10 Basic Requirements **154**

11 Charge Coupled Device Fundamentals **156**

11.1 Surface Channel Devices 156

11.2 Buried Channel Devices (BCCDs) 158

11.3 Charge Transfer and Clocking 166

11.4 Transfer Efficiency 168

11.5 Power Dissipation 170

11.5.1 On-chip dissipation 170

11.5.2 Power dissipation in the clock drivers 171

11.6 Charge Input and Output 172

11.6.1 Fill-and-Spill input method 172

11.6.2 Floating gate amplifier output structures 175

12 CCD Edge Detector Design **179**

12.1 CCD Processing Array 180

12.1.1 Charge sensing 183

12.1.2 Clock sequences 187

12.1.3 Boundary processing and charge input 192

12.2 Edge Detection Circuit 197

12.2.1 Double-ended differential amplifier 198

12.2.2 Comparator circuit 201

12.2.3 Edge storage 204

13 Edge Detector Test Results **206**

13.1 Test System 209

13.2 Driving Signals Off-Chip 212

13.3 CCD Tests 215

13.3.1 Channel potential measurements 215

13.3.2 Input and output 217

13.3.3 Dark current measurements 220

13.3.4 Transfer efficiency 222

13.4 Differencing and Threshold Test Circuits 224
13.4.1 Differential amplifier 224
13.4.2 Sense amplifier voltage comparator 227
13.4.3 Combined edge veto test circuit 228
13.5 Operation of the Full Array Processors 230
13.6 Recommendations for Improving the Design 237

III A Mixed Analog/Digital Edge Correlator **240**

14 Design Specifications **241**
14.1 Required Block Size 243
14.2 Required Search Area 244
14.3 Precision 246

15 Case Study: A Purely Digital Design **248**
15.1 First Method: Moving the Search Window 249
15.2 Second Method: One Processor Per Offset 253

16 Mixing Analog and Digital **260**
16.1 Unit Cell Design 262
16.2 Global Test Circuits 266
16.2.1 Current scaling and conversion 266
16.2.2 Validation and V_{min} tests 270
16.3 Test Pattern Simulation 274
16.4 Comparison with Digital Architectures 279

IV Conclusions **281**

17 Recommendations for Putting It All Together **282**

Bibliography **286**

Appendices **296**

A Quaternion Algebra **297**

B Special Integrals **300**

B.1 $\int\int |\ell'|^2 \ell' \, \xi \, d\xi \, d\alpha$ 301

B.2 $\int\int |\ell'|^4 \, \xi \, d\xi \, d\alpha$ 302

B.3 $\int\int \ell' \ell'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha$ 302

B.4 $\int\int |\ell'|^2 \ell' \ell'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha$ 303

B.5 $\int\int (\mathbf{u} \cdot \ell') \ell' \ell'^{T} \, \xi \, d\xi \, d\alpha$ 304

B.6 $\int\int (\mathbf{u}_1 \cdot \ell')(\mathbf{u}_2 \cdot \ell') \ell' \ell'^{T} \, \xi \, d\xi \, d\alpha$ 305

List of Figures

2-1 Geometry of perspective projection. 26

2-2 Epipolar geometry. 29

3-1 Example of matching edges despite foreshortening. 55

4-1 System block diagram 59

5-1 Results of applying the multi-scale veto rule to an ideal step edge and to an impulse. 65

5-2 1-D resistive smoothing network with controllable space constant. 69

5-3 2-D MSV edge detection array using charge-coupled devices (CCDs) . . . 70

5-4 Conceptual model of the 'Edge precharge circuit'. 71

5-5 Ideal 2-D image features. 74

5-6 Simulated motion sequence: Frontal plane at 10 (baseline) units. Motion is given by: $\mathbf{b} = (1, 0, 0)$, $\theta = 5^o$, $\hat{\omega} = (0, 1, 0)$. 77

5-7 Real motion sequence: Motion with respect to motion stage coordinate system (not camera system): $\mathbf{b} = (1, 0, 0)$, $\theta = 5^o$, $\hat{\omega} = (0, 0, 1)$. 78

6-1 Threshold selection for a decision rule that chooses between two hypotheses H_0 and H_1. 81

6-2 Binary edge maps of astronaut sequence with correspondence points found by the matching procedure. 86

6-3 Binary edge maps of real motion sequence (lab scene) with correspondence points found by the matching procedure. 87

7-1 Binary edge maps of real motion sequence with *hand picked* point matches. 103

8-1 Geometry of the position vector for a point on the image plane and cone of all vectors for a given field of view. 105
8-2 Change in $\mathbf{r}_i$ caused by error in determining the exact location of the correspondence with the left image. 121
8-3 Probability density function used to obtain random Z values. 139
8-4 Estimation errors in translation and rotation with $\mathbf{b} \perp \hat{v}_3$ (uncorrelated measurement errors). $\mathbf{b} = (1,0,0), \hat{\omega} = (0,0,1), \theta = 5^o$ 141
8-5 Estimation errors in translation and rotation with $\mathbf{b} \parallel \hat{v}_3$ (uncorrelated measurement errors). $\mathbf{b} = (0,0,1), \hat{\omega} = (0,0,1), \theta = 5^o$ 142
8-6 Actual and predicted values of μ_2/μ_3 for $\mathbf{b} \perp \hat{v}_3$, for $\phi = 20^o$, 40^o, and 60^o with $N = 50$ and $N = 100$. 143
8-7 Actual and predicted values of μ_2/μ_3 for $\mathbf{b} \parallel \hat{v}_3$, for $\phi = 20^o$, 40^o, and 60^o with $N = 50$ and $N = 100$. 144
8-8 Errors in estimates of translation and rotation with $\mathbf{b} \perp \hat{v}_3$ (Systematic measurement errors). $\mathbf{b} = (1,0,0), \hat{\omega} = (0,0,1), \theta = 5^o$. 147
8-9 Errors in estimates of translation and rotation with $\mathbf{b} \parallel \hat{v}_3$ (Systematic measurement errors). $\mathbf{b} = (0,0,1), \hat{\omega} = (0,0,1), \theta = 5^o$. 148

11-1 'Bucket brigade' analogy of charge coupled devices. 157
11-2 The MOS capacitor. 158
11-3 States of the MOS capacitor. 159
11-4 Potential profile with depth of a buried channel CCD 160
11-5 Lumped capacitance model of a BCCD 161
11-6 Charge density profile of a buried channel CCD with signal charge Q_{sig} (using depletion approximation) 163
11-7 Charge transfer sequence with four-phase clocking. 167
11-8 Fill-and-spill method for charge input 173
11-9 Charge sensing using the floating gate technique 176

12-1 Floor plan of the complete MSV processor. 180
12-2 Unit cell architecture. 181
12-3 Unit cell layout. 182
12-4 Floating gate design. 183

12-5 Simulated source follower characteristic for floating gate amplifier (V_{bias} = 1V). 186
12-6 Charge averaging operation. 188
12-7 Backspill caused by potential mismatch. 191
12-8 Level shift circuit for correcting potential mismatch problem. 191
12-9 Boundary cell design (north). 193
12-10 Boundary cell design (east). 193
12-11 Boundary cell design (west). 194
12-12 CCD input structure. 195
12-13 layout of the fill-and-spill structure. 195
12-14 Multi-scale veto edge detection circuit block diagram. 197
12-15 Clock waveforms for driving edge detection circuit. 198
12-16 Differential amplifier circuit. 199
12-17 Simulated differential amplifier characteristic for common mode voltages V_c = 1.2V, 2.0V, and 2.8V (V_{bias} = 1V). 199
12-18 Sense amplifier voltage comparator for threshold tests. 202
12-19 Sense amplifier response with resolution of δ = 10mV. 202
12-20 Edge charge storage latch. 204

13-1 Die photograph of 32×32 array. 207
13-2 Die photograph of the test structures chip. 208
13-3 Test system design 210
13-4 Clock driver circuit used in the test system 212
13-5 Analog output pad driver circuit 213
13-6 Analog output pad driver characteristic – simulated vs. actual (V_{low} = 1.2V, V_{high} = 3.8V). 214
13-7 CCD channel potentials (4-27-94 run). 216
13-8 CCD channel potentials (10-28-93 run). 216
13-9 Measured source follower characteristic for floating gate amplifier (V_{bias} = 1V). 218
13-10 Measured floating gate amplifier output vs. fill-and-spill input, $V_{in} - V_{ref}$. 219
13-11 Dark current accumulation for cooled and uncooled chips. 221
13-12 Charge transfer efficiency measurement. 223

13-13 Measured differential amplifier characteristic for common mode voltages $V_c = 1.2$V, 2.0V, and 2.8V with $V_{bias} = 1$V. 225
13-14 Combined maximum differential amplifier outputs vs. $|V_1 - V_2|$ for common mode voltages $V_c = 1.2$V, 2.0V, and 2.8V with $V_{bias} = 1$V. 226
13-15 Measured sense amplifier switching characteristic. 227
13-16 Measured response of one absolute-value-of-difference circuit. 229
13-17 Measured response of one absolute-value-of-difference circuit, plotted against $|V_1 - V_2|$. 229
13-18 Composite response of twelve AVD circuits. 231
13-19 Floating-gate outputs of 4×4 array processor. 232
13-20 Floating-gate outputs of 32×32 array processor. 232
13-21 Smoothing of one-pixel impulse on 4×4 array. 233
13-22 Edge detection results for 4×4 array with impulse input. 234
13-23 Test image used on 32×32 processor and corresponding edges. 236
13-24 Proposed architecture for focal-plane processor 238
13-25 Unit cell structure in focal-plane processor. 238

15-1 Processing array with base image block held in fixed position. 250
15-2 Individual processing cell. 250
15-3 Tally circuit construction. 251
15-4 Layout of a unit cell including one full adder circuit. 254
15-5 Circuit diagram for the layout of Figure 15-4. 255
15-6 Offset processor array. 256
15-7 Block diagram of one processor node with 8-bit counter/accumulator cell. 257
15-8 Layout of 8-bit counter and accumulator with inhibit on overflow. 258

16-1 Processing array with both analog and digital elements. 261
16-2 Circuit diagram of unit cell with switched current sources. 263
16-3 Unit cell layout. 263
16-4 Transient behavior of switched current sources. 264
16-5 Output current vs. load voltage for unit cell current sources. 264
16-6 I–V conversion circuits. 267
16-7 I_B current divider characteristic. 268

16-8 Simulated I–V characteristic for computing $I_B \rightarrow \alpha_1 V_B$. 268
16-9 Output score voltage vs. number of processors with non-matching edge pixels. 269
16-10 Rise and fall times of V_V for a 0.3mA input after switching on current sources and before turning on reset transistor. 269
16-11 V_{min} circuit. 271
16-12 Toggle circuit for signalling update to V_{min}. 271
16-13 Input/output characteristic for source followers used in comparator circuits. 272
16-14 Test pattern for simulating the 5×5 array. 274
16-15 Layout of 5×5 array. 275
16-16 Clock waveforms. 276
16-17 V_V and $\alpha_1 V_B$ outputs sampled at end of each cycle. 277
16-18 Response of threshold test circuit at each offset of test pattern. 278
16-19 Response of V_{min} toggle circuit at each offset of test pattern. 278

17-1 Single ‘board’ motion estimation system. 283

List of Tables

5.1 Attenuation factors for different types of features as a function of smoothing 73

7.1 Simulation results on the astronaut sequence. 100
7.2 Simulation results on the lab sequence with points from automatic matching procedure. 101
7.3 Simulation results on the lab sequence with hand-picked correspondence points. 101

12.1 Orbit CCD/CMOS process parameters (from 10-28-93 run). 184
12.2 Floating gate parameter values for $Vg = 2.5$V and $N_e = N_{e,max}$. 185
12.3 Capacitances loading the floating node gate. 187

Chapter 1

Introduction

It is difficult to overstate the potential usefulness of an automated system to compute motion from visual information for numerous applications involving the control and navigation of moving vehicles. Deducing 3-D motion by measuring the changes in sucessive images taken by a camera moving through the environment involves determining the perspective transformation between the coordinate systems defined by the camera's principal axes at its different locations. Solving this problem is important not only for motion estimation, but also for determining depth from binocular stereopairs. It is in fact equivalent to the classic problem in photogrammetry of relative orientation, for which methods were developed by cartographers over a hundred years ago to measure the topography of large scale land masses [1], [2], [3], [4], [5].

Computing relative orientation involves two difficult subproblems. First, corresponding points must be identified in the two images, and second, the nonlinear equations defining the perspective transformation must be inverted to solve for the parameters of the motion. Not all image pairs allow an unambiguous determination of their relative orientation, however. For some configurations of points there is not a unique solution to the motion equations and for many others the problem is ill-conditioned [6], [7], [3], [4], [8]. The methods developed long ago for cartography relied on considerable human intervention to overcome these difficulties. Large optical devices known as stereoplotters were invented to align matching features using a floating mark positioned by an operator, while general knowledge of the geometry of the scene and the camera positions was used to aid in solving the motion equations[1].

Efforts to construct autonomous systems have also been limited by the complexity of the task. At present, machine vision algorithms for computing camera motion and alignment have reached a level of sophistication in which they can operate under special conditions in restricted environments. Among the systems which have been developed, however, there is a strong correlation between robustness and the amount of computational resources employed. Two approaches which are commonly taken are to either impose very restrictive conditions on the type and amount of relative motion allowed, in which case simple algorithms can be used to yield qualitatively correct results as long as the basic assumptions are not violated; or to relax the restrictions and therefore implement the system with complex algorithms that require powerful processors.

The goal of this thesis is to go beyond these limitations and to design a system that is both unrestrictive and that uses minimal hardware. Specifically, the objectives of such a system are the following:

- The complete system should be physically small and should operate with minimal power. This is particularly important if it is to be used in remote or inaccessible environments.

- It should allow real-time, frame-rate operation.

- It must be able to either produce an accurate estimate of the motion for the majority of situations it is likely to encounter or to recognize and report that a reliable estimate cannot be obtained from the given images.

- The system should be self-contained, in the sense that neither external processing nor outside intervention is required to determine accurate estimates of the camera motion.

A central tenet of this thesis is that in order to meet the first two requirements, specialized VLSI processors, combining both analog and digital technology, are needed to perform specific tasks within the system. Clearly, meeting the last two requirements does not necessitate special hardware since they influence only the choice of the algorithms to

[1]Cartography is still a labor intensive process; although in the interest of developing geographic information systems (GIS), there have been many efforts in the last decade to automate mapping techniques by applying algorithms developed for machine vision ([9], see also the April 1983 issue of *Photogrammetric Engineering and Remote Sensing*).

be implemented. Obviously, any algorithm which can be wired into a circuit can be programmed on general purpose digital hardware. The motivation for designing specialized processors is the idea that in doing so a significant reduction in size and power consumption can be achieved over general purpose hardware.

There are many aspects to a completely general system for computing camera motion and alignment, and it is necessary to define the limits of this study. Specifically,

- Only image sequences from passive navigation will be examined. In other words, it is always assumed that the environment is static and all differences observed in the images are due to differences in camera position. The case of multiple independently moving objects in the scene will not be explicitly addressed.

- The system design is based entirely on the problem of estimating motion from two frames only. Many researchers [10], [11], [12] have proposed the use of multiple frames in order to improve reliability, on the grounds that the results from two frames are overly sensitive to error and are numerically unstable. The philosophy of the present approach is that it is necessary to build a system which can extract the best results possible from two frames in order to make a multi-frame system even more reliable. Nothing in the design of the present system will prevent it from being used as a module within a more comprehensive multi-frame system.

The goal of this thesis is not to build the complete motion system, but to develop the theory on which it is based, and to design the specialized processors needed for its operation. This thesis is divided into three major parts. Part I covers the theoretical issues of selecting and adapting the algorithms to be implemented by the special processors. It also includes a complete analysis of the numerical stability of the motion algorithm and of the sensitivity of its estimates to errors in the data. Parts II and III are concerned with the design of the processors needed for detemining point correspondences. One of the conclusions of Part I is that matching edges in the two images by binary block correlation is the most suitable method for implementation in VLSI. Part II describes a prototype edge detector built in CCD-CMOS technology which implements the multi-scale veto algorithm presented in Chapter 5, while Part III examines the benefits of combining analog and digital processing to design an area-efficient edge matching circuit.

Part I

Algorithms and Theory

Chapter 2

Methods for Computing Motion and Structure

Computing motion and structure from different views involves two operations: matching features in the different images, and solving the motion equations for the rigid body translation and rotation which best describes the observed displacements of brightness patterns in the image plane. Methods can be grouped into two categories according to whether features in the two images are matched explicitly or implicitly. Explicit methods generate a discrete set of feature correspondences and solve the motion equations using the known coordinates of the pairs of matched points in the set. Implicit methods formulate the motion equations in terms of the temporal and spatial derivatives of brightness and the incremental displacements in the image plane, or optical flow. In order to avoid explicit matching, these methods incorporate additional constraints, such as brightness constancy and smoothness of the optical flow, and derive the motion from a global optimization procedure.

There are advantages and weaknesses to both approaches. Explicit methods require few assumptions other than rigid motion; however, they must first solve the difficult problem of finding an accurate set of point correspondences. In addition, although more of a concern for determining structure from binocular stereo than for computing motion, depth can only be recovered for points in the set. Implicit methods circumvent the correspondence problem but in exchange must make more restrictive assumptions on the environment. Since they are based on approximating brightness derivatives from sampled data, they are both sensitive to noise and sensor variation and susceptible to aliasing.

In the interest of removing as many restrictions as possible, an explicit approach has been adopted for the present system. Details of the specific methods which will be used to perform the tasks of finding point correspondences and solving the motion equations will be described in the following chapters. In this chapter, in order to situate this research with respect to related work, the basic methods for computing motion and alignment are presented in a more general context. I will first rederive the fundamental equations of perspective geometry, presenting the notation which is used throughout the thesis, and will then discuss several of the more significant algorithms which have been developed for both the explicit and the implicit approaches. Finally, I will review several previous and ongoing efforts to build systems in VLSI based on these different methods.

2.1 Basic Equations

In order to express the motion equations in terms of image plane coordinates, it is first necessary to formulate the relation between the 3-D coordinates of objects in the scene and their 2-D projections. Once the motion is known, the projection equations can be inverted to yield the 3-D coordinates of the features which were matched in the images.

The exact projective relation for real imaging systems is in general nonlinear. However, it can usually be well approximated by a linear model. If greater precision is needed, nonlinearities can be accounted for either by adding higher order terms or by pre-warping the image plane coordinates to fit the linear model. Of the two choices most commonly used for the basic linear relation—orthographic and perspective—only perspective projection can meet the requirements of the present system. Orthographic projection, which approximates rays from the image plane to objects in the scene as parallel straight lines, is the limiting case of perspective projection as the field of view goes to zero or as the the distance to objects in the scene goes to infinity. Orthographic projection has often been used in machine vision for the recovery of structure and motion [13], [14] because it simplifies the motion equations. In the orthographic model the projected coordinates are independent of depth and are therefore uncoupled. For the same reason, however, it is impossible to uniquely determine motion from two orthographic views [15].

2.1.1 Perspective geometry

Under the assumption of perfect perspective projection, such as would be obtained with an ideal pinhole camera, we define the camera coordinate system as shown in Figure 2-1 with origin at the center of projection. The image plane is perpendicular to the z axis and is

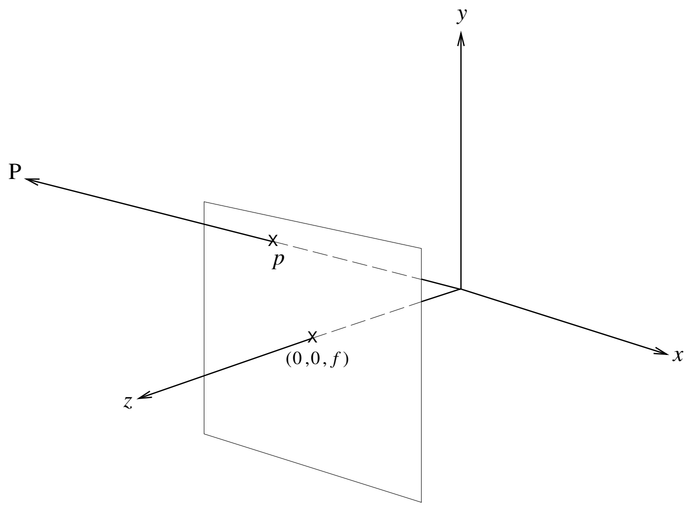


FIGURE 2-1: Geometry of perspective projection.

located at a distance f, which is the *effective focal length*, from the origin. In a real camera, of course, the image sensor is located behind the optical center of projection at $z = -f$. It is customary, however, to represent the image plane as shown in the diagram in order to avoid the use of negative coordinates. The point where the $\hat{z}$, or optical, axis pierces the image plane is referred to as the *principal point*.

Let $\mathbf{p} = (X, Y, Z)^{\mathrm{T}}$ represent the position vector in the camera coordinate system of a point P in the scene and let $\tilde{\mathbf{p}} = (X/Z, Y/Z, 1)^{\mathrm{T}}$ denote the 2-dimensional homogeneous representation of $\mathbf{p}$. Two world points P_i and P_j are projectively equivalent with respect to a plane perpendicular to the z-axis if and only if $\tilde{\mathbf{p}}_i = \tilde{\mathbf{p}}_j$. For the world point P to be imaged on the plane $z = f$ at P', whose coordinates are (x, y, f), P and P' must be projectively equivalent. In other words

$$\frac{x}{f} = \frac{X}{Z}, \quad \text{and,} \quad \frac{y}{f} = \frac{Y}{Z} \tag{2.1}$$

Since image irradiance, or brightness, is always sampled discretely by an array of photosensors, it is convenient to define a secondary set of coordinates, (m_x, m_y) on the array of picture cells such that the centers of each pixel are located at integer values of m_x and m_y. The vector $\mathbf{m} = (m_x, m_y, 1)^{\mathrm{T}}$ is related by a linear transformation matrix $\mathbf{K}_c$ to the 2-D homogeneous representation, in the camera coordinate system, of all points which are projectively equivalent to (x, y, f)

$$\mathbf{m} = \mathbf{K}_c \tilde{\mathbf{p}} \tag{2.2}$$

Under the conditions illustrated in Figure 2-1, $\mathbf{K}_c$ has the special form

$$\mathbf{K}_c = \begin{pmatrix} f/s_x & 0 & m_{x0} \\ 0 & f/s_y & m_{y0} \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

where f is the effective focal length; s_x and s_y are the physical distances, measured in the same units as f, between pixel centers along the orthogonal x- and y-axes; and (m_{x0}, m_{y0}) is the location, in pixel coordinates, of the principal point. The matrix $\mathbf{K}_c$ is referred to as the *internal camera calibration matrix* and must be known before scene structure and camera motion can be recovered from the apparent motion of brightness patterns projected onto the image plane.

In real devices, $\mathbf{K}_c$ seldom has exactly the form of (2.3) due to factors such as the misalignment of the image sensor and spherical aberrations in the lens. Finding the appropriate transformation is a difficult problem, and consequently numerous methods have been developed, involving varying degrees of complexity, to determine internal calibration [16], [17], [18], [19], [20], [21]. Discussing these methods, however, goes well beyond the scope of this thesis, and so it will be assumed for present purposes that the calibration is known.

2.1.2 The epipolar constraint

Suppose we have two images taken at two different camera positions, which we will refer to as *right* and *left*. Let $\mathbf{p}_r$ and $\mathbf{p}_l$ denote the position vectors with respect to the right and left coordinate systems to a fixed point P in the environment, $\mathbf{p}_r = (X_r, Y_r, Z_r)^{\mathrm{T}}$ and $\mathbf{p}_l = (X_\ell, Y_\ell, Z_\ell)^{\mathrm{T}}$. Assuming a fixed environment so that rigid body motion is applicable, $\mathbf{p}_r$ and $\mathbf{p}_l$ are related by

$$\mathbf{p}_r = \mathbf{R}\mathbf{p}_l + \mathbf{b} \tag{2.4}$$

where $\mathbf{R}$ denotes an orthonormal rotation matrix, and $\mathbf{b}$ is the baseline vector connecting the origins of the two systems. A necessary condition for the vectors $\mathbf{p}_r$ and $\mathbf{p}_l$ to intersect at P is that they be coplanar with the baseline, $\mathbf{b}$, or equivalently, that the triple product of the three vectors vanish

$$\mathbf{p}_r \cdot (\mathbf{b} \times \mathbf{R}\mathbf{p}_l) = 0 \tag{2.5}$$

Given $\mathbf{R}$, $\mathbf{b}$, and the coordinates (x_ℓ, y_ℓ, f) of the point P'_ℓ, which is the projection of P in the left image, equation (2.5) defines a line in the right image upon which the corresponding point P'_r at (x_r, y_r, f) must lie. This line, known as the *epipolar line*, is the intersection of the image plane with the plane containing the point P and the baseline $\mathbf{b}$. The position of P'_r on the epipolar line is determined by the depth, or z-coordinate, of P in the left camera system. To see this, define the variables ξ, η, and ϕ to represent the components of the rotated homogeneous vector $\mathbf{R}\tilde{\mathbf{p}}_\ell$

$$\mathbf{R}\tilde{\mathbf{p}}_\ell = \begin{pmatrix} \xi \\ \eta \\ \phi \end{pmatrix} \tag{2.6}$$

Then equation (2.4) can be expressed in component form as

$$\begin{pmatrix} X_r \\ Y_r \\ Z_r \end{pmatrix} = Z_\ell \begin{pmatrix} \xi \\ \eta \\ \phi \end{pmatrix} + \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} \tag{2.7}$$

The projection of P onto the right image is found from

$$x_r = f\frac{X_r}{Z_r} = f\frac{Z_\ell \xi + b_x}{Z_\ell \phi + b_z} \tag{2.8}$$

and

$$y_r = f\frac{Y_r}{Z_r} = f\frac{Z_\ell \eta + b_y}{Z_\ell \phi + b_z} \tag{2.9}$$

P'_r thus varies along the epipolar line between $f(b_x/b_z, b_y/b_z)$ when $Z_\ell = 0$ to $f(\xi/\phi, \eta/\phi)$ when $Z_\ell = \infty$. The first point, known as the *epipole*, is independent of x_ℓ and y_ℓ, and is therefore common to all epipolar lines. By rewriting (2.4) as

$$\mathbf{p}_l = \mathbf{R}^{\mathrm{T}}\mathbf{p}_r + \mathbf{b}' \tag{2.10}$$

where $\mathbf{b}' = -\mathbf{R}^{\mathrm{T}}\mathbf{b}$, a similar relation can be obtained for the coordinates of the point P'_ℓ, the projection of P onto the left image plane, in terms of Z_r and the components of the rotated vector $\mathbf{R}^{\mathrm{T}}\tilde{\mathbf{p}}_r$. The geometry of the epipolar transformation is illustrated in Figure 2-2.

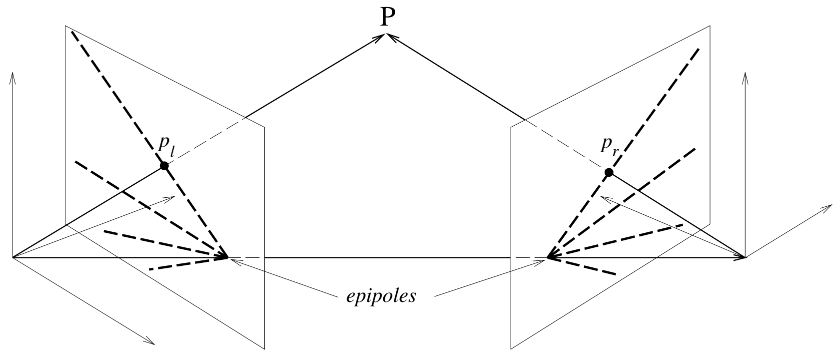


FIGURE 2-2: Epipolar geometry.

Binocular stereopsis is based on the fact that the depth of objects in the scene can be recovered from their projections in two images with a known relative orientation. From (2.8) and (2.9) it is easily seen that

$$Z_\ell = \frac{fb_x - b_z x_r}{\phi x_r - f\xi} = \frac{fb_y - b_z y_r}{\phi y_r - f\eta} \tag{2.11}$$

The quantity $\phi x_r - f\xi$ is known as the *horizontal disparity*, d_H, and the quantity $\phi y_r - f\eta$ as the *vertical disparity*, d_V. If $\mathbf{R} = \mathbf{I}$ and $\mathbf{b} = (|\mathbf{b}|, 0, 0)$, then $\xi = x_\ell / f$, $\phi = 1$ and equation (2.11) reduces to the familiar parallel geometry case in which

$$z_r = z_\ell = \frac{f|\mathbf{b}|}{|x'_r - x'_\ell|} \tag{2.12}$$

and for which the vertical disparity is necessarily zero.

2.2 Computing Motion by Matching Features

In this section, we will examine only those methods which compute motion from point correspondences. Although algorithms using higher level features, such as lines and planes have been proposed ([22], [23], [24], [25]), these usually require more than two views and also are not as practical for hardware implementation.

To compute camera motion, or relative orientation, we need to find the rotation and baseline vector that best describe the transformation between the right and left camera systems. We assume that we have a set of N pairs, $\{(\mathbf{r}_i, \boldsymbol{\ell}_i)\}$, $i = 1, \ldots, N$, where $\mathbf{r}_i$ and $\boldsymbol{\ell}_i$ are the position vectors of the points P'_{r_i} and P'_{ℓ_i} which are projectively equivalent, in the right and left systems respectively, to the same world point P.

The fundamental equation for computing the coordinate transformation between the two camera systems is the coplanarity constraint given by equation (2.5). Since this equation is homogeneous, it can be written in terms of $\mathbf{r}_i$ and $\boldsymbol{\ell}_i$ as

$$\mathbf{r}_i \cdot (\mathbf{b} \times \mathbf{R}\boldsymbol{\ell}_i) = 0 \tag{2.13}$$

Equation (2.13) is unaffected by the lengths of any of the vectors $\mathbf{r}_i$, $\boldsymbol{\ell}_i$, or $\mathbf{b}$. We usually set $|\mathbf{b}| = 1$ so that the baseline length becomes the unit of measure for all distances in the scene. The vectors $\mathbf{r}_i$ and $\boldsymbol{\ell}_i$ may be set equal to the homogeneous vectors $\widetilde{\mathbf{p}}_{r\,i}$ and $\widetilde{\mathbf{p}}_{\ell i}$

$$\mathbf{r}_i = \widetilde{\mathbf{p}}_{r\,i} \text{ and, } \boldsymbol{\ell}_i = \widetilde{\mathbf{p}}_{\ell i} \tag{2.14}$$

or may also be assigned unit length:

$$\mathbf{r}_i = \frac{\widetilde{\mathbf{p}}_{r\,i}}{|\widetilde{\mathbf{p}}_{r\,i}|} \text{ and, } \boldsymbol{\ell}_i = \frac{\widetilde{\mathbf{p}}_{\ell i}}{|\widetilde{\mathbf{p}}_{\ell i}|} \tag{2.15}$$

The second choice is often referred to as *spherical* projection.

2.2.1 Representing rotation

There are several ways to represent rotation, including orthonormal matrices as in (2.13). The matrix form is not always the best choice, however, as it requires nine coefficients, even though there are only three degrees of freedom. A rotation is completely specified by the pair $(\theta, \widehat{\boldsymbol{\omega}})$, where θ represents the angle of rotation about the axis $\widehat{\boldsymbol{\omega}} = (\omega_x, \omega_y, \omega_z)^{\mathrm{T}}$, with $|\widehat{\boldsymbol{\omega}}| = 1$. The relation between the orthonormal matrix $\mathbf{R}$ and $(\theta, \widehat{\boldsymbol{\omega}})$ is given by

$$\mathbf{R} = \begin{pmatrix} \cos\theta + \omega_x^2(1-\cos\theta) & \omega_x\omega_y(1-\cos\theta) - \omega_z\sin\theta & \omega_x\omega_z(1-\cos\theta) + \omega_y\sin\theta \\ \omega_x\omega_y(1-\cos\theta) + \omega_z\sin\theta & \cos\theta + \omega_y^2(1-\cos\theta) & \omega_y\omega_z(1-\cos\theta) - \omega_x\sin\theta \\ \omega_x\omega_z(1-\cos\theta) - \omega_y\sin\theta & \omega_y\omega_z(1-\cos\theta) + \omega_x\sin\theta & \cos\theta + \omega_z^2(1-\cos\theta) \end{pmatrix} \tag{2.16}$$

which can also be expressed as

$$\mathbf{R} = \cos\theta\mathbf{I} + (1-\cos\theta)\hat{\omega}\hat{\omega}^{\mathrm{T}} + \sin\theta\boldsymbol{\Omega}_{\times} \tag{2.17}$$

where

$$\boldsymbol{\Omega}_\times = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \tag{2.18}$$

Equation (2.17) leads directly to Rodrigues' well-known formula for the rotation of a vector $\boldsymbol{\ell}$

$$\begin{aligned} \mathbf{R}\boldsymbol{\ell} &= \cos\theta\boldsymbol{\ell} + (1-\cos\theta)(\hat{\omega}\cdot\boldsymbol{\ell})\hat{\omega} + \sin\theta\hat{\omega}\times\boldsymbol{\ell} \\ &= \boldsymbol{\ell} + \sin\theta\hat{\omega}\times\boldsymbol{\ell} + (1-\cos\theta)\hat{\omega}\times(\hat{\omega}\times\boldsymbol{\ell}) \end{aligned} \tag{2.19}$$

A frequently useful and compact representation of rotation is the unit quaternion. Quaternions are vectors in $\mathbb{R}^4$ which may be thought of as the composition of a scalar and 'vector' part [4].

$$\mathring{a} = (a_0, \mathbf{a}) \tag{2.20}$$

where $\mathbf{a} = (a_x, a_y, a_z)^{\mathrm{T}}$ is a vector in $\mathbb{R}^3$. An ordinary vector $\mathbf{v}$ in $\mathbb{R}^3$ is represented in quaternion form as

$$\mathring{v} = (0, \mathbf{v}) \tag{2.21}$$

A unit quaternion is one whose magnitude, defined as the square root of its dot product with itself, is unity.

$$\mathring{q}\cdot\mathring{q} = 1 \tag{2.22}$$

Unlike vectors in $\mathbb{R}^3$, quaternions are endowed with special operations of multiplication and conjugation, and thus form the basis of a complete algebra. The fundamental operations and identities of quaternion algebra are summarized in Appendix A.

The usefulness of quaternions lies in the simplicity with which rotation about an arbitrary axis can be represented. Every unit quaternion may be written as

$$\mathring{q} = \left(\cos\frac{\theta}{2},\, \hat{\boldsymbol{\omega}}\,\sin\frac{\theta}{2}\right) \tag{2.23}$$

and the rotation of the vector $\mathbf{v}$ by an angle θ about $\hat{\boldsymbol{\omega}}$ by

$$\mathring{v}' = \mathring{q}\mathring{v}\,\mathring{q}^* \tag{2.24}$$

where $\mathring{q}^*$ represents the conjugate of $\mathring{q}$.

In the following discussions I will alternate between these different representations, to use whichever form is best suited to the problem at hand.

2.2.2 Exact solution of the motion equations

There are five unknown parameters in equation (2.13), two for the direction of the baseline and three for the rotation. It has long been known that relative orientation can be determined from a minimum of five points, as long as these do not lie on a degenerate surface. Due to the rotational component, however, the equations are nonlinear and must be solved by iterative methods. Furthermore, the five-point formulation admits multiple solutions[1] [7], [4].

Thompson [26] first showed how the coplanarity conditions could be formulated as a set of nine homogeneous linear equations, and Longuet-Higgins [27] proposed an algorithm to derive the baseline vector and rotation matrix from the solution to the equations obtained from eight point correspondences. This algorithm is summarized as follows:
The first step is to rewrite the coplanarity constraint (2.13) as

$$\begin{aligned} \mathbf{r}_i \cdot (\mathbf{b} \times \mathbf{R}\boldsymbol{\ell}_i) &= -\mathbf{R}\boldsymbol{\ell}_i \cdot (\mathbf{b} \times \mathbf{r}_i) \\ &= -\boldsymbol{\ell}_i^{\mathrm{T}} \mathbf{R}^{\mathrm{T}} \mathbf{B}_\times \mathbf{r}_i \end{aligned} \tag{2.25}$$

where

$$\mathbf{B}_\times = \begin{pmatrix} 0 & -b_z & b_y \\ b_z & 0 & -b_x \\ -b_y & b_x & 0 \end{pmatrix} \tag{2.26}$$

We define the matrix $\mathbf{E}$ as

$$\mathbf{E} = \mathbf{R}^{\mathrm{T}} \mathbf{B}_\times \tag{2.27}$$

[1] Faugeras and Maybank [7] first proved that there are at most 10 solutions for the camera motion given 5 correspondences, thereby correcting a longstanding error by Kruppa [5] who had thought there were at most 11.

and order the components of $\mathbf{E}$ as

$$\mathbf{E} = \begin{pmatrix} e_1 & e_4 & e_7 \\ e_2 & e_5 & e_8 \\ e_3 & e_6 & e_9 \end{pmatrix} \tag{2.28}$$

Let $\mathbf{a}_i$ denote the 9×1 vector formed from the products of the components of $\mathbf{r}_i$ and $\boldsymbol{\ell}_i$

$$\mathbf{a}_i = \begin{pmatrix} r_{xi}\ell_{xi} \\ r_{xi}\ell_{yi} \\ r_{xi}\ell_{zi} \\ r_{yi}\ell_{xi} \\ r_{yi}\ell_{yi} \\ r_{yi}\ell_{zi} \\ r_{zi}\ell_{xi} \\ r_{zi}\ell_{yi} \\ r_{zi}\ell_{zi} \end{pmatrix} \tag{2.29}$$

and let $\mathbf{e}$ denote the 9×1 vector of the elements of $\mathbf{E}$. Then the coplanarity constraint for each pair of rays results in an equation of the form

$$\mathbf{a_i}^{\mathrm{T}}\mathbf{e} = 0 \tag{2.30}$$

Eight correspondences result in eight equations which can be solved to within a scale factor for the elements of $\mathbf{E}$. Given $\mathbf{E}$, the baseline vector is identified as the eigenvector corresponding to the zero eigenvalue of $\mathbf{E}^{\mathrm{T}}\mathbf{E}$

$$\mathbf{E}^{\mathrm{T}}\mathbf{E}\mathbf{b} = 0 \tag{2.31}$$

as can be seen from the fact that

$$\begin{aligned} \mathbf{E}^{\mathrm{T}}\mathbf{E} &= \mathbf{B}_{\times}{}^{\mathrm{T}}\mathbf{B}_{\times} \\ &= \mathbf{I} - \mathbf{b}\mathbf{b}^{\mathrm{T}} \end{aligned} \tag{2.32}$$

The rotation matrix $\mathbf{R}$ is found from

$$\mathbf{R} = \mathbf{B}_\times \mathbf{E}^{\mathrm{T}} - \mathrm{Cof}(\mathbf{E}^{\mathrm{T}})^{\mathrm{T}} \tag{2.33}$$

where $\mathrm{Cof}(\mathbf{E}^{\mathrm{T}})$ is the matrix of cofactors of $\mathbf{E}^{\mathrm{T}}$.

2.2.3 Least-squares methods

If there is no error in the data, the Longuet-Higgins algorithm will give a unique solution for the motion[2] except for certain configurations of points which lie on special surfaces [8], [28]. It is extremely difficult, however, to obtain error-free data, particularly if the correspondences are determined by an automatic procedure. It turns out that the 8-point linear algorithm is extremely unstable in the presence of noise, due largely to the fact that the equations (2.30) do not take into account dependencies between the elements of $\mathbf{E}$, and hence their solution cannot be decomposed into the product form of equation (2.27).

Even when nonlinear methods are used to solve the coplanarity constraint equations, the solution is very sensitive to noise when few correspondences are used [29]. With error in the data, the ray pairs are not exactly coplanar and equation (2.13) should be written as

$$\mathbf{R}\boldsymbol{\ell}_i \cdot (\mathbf{r}_i \times \mathbf{b}) = \lambda_i \tag{2.34}$$

Instead of trying to solve the constraint equations exactly, it is better to find the solution that minimizes the error norm

$$S = \sum_{i=1}^{N} \lambda_i^2 \tag{2.35}$$

A somewhat improved approach over the 8-point algorithm was proposed by Weng *et al.* [30] based on a modification of a method originally presented by Tsai and Huang [28].

[2]There is an intrinsic fourfold ambiguity to every solution; however, these are all counted as one. This ambiguity will be discussed in more detail in Chapter 7.

They defined an $N \times 9$ matrix $\mathbf{A}$ as

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{a}_N^{\mathrm{T}} \end{pmatrix} \tag{2.36}$$

such that $S = |\mathbf{Ae}|^2$. The vector $\mathbf{e}$ which minimizes S is the eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ with the smallest eigenvalue. The baseline direction and rotation are derived from the resulting matrix $\mathbf{E}$ such that $\mathbf{b}$ minimizes $|\mathbf{E}^{\mathrm{T}}\mathbf{Eb}|$ and $\mathbf{R}$ is the orthonormal rotation matrix that minimizes $|\mathbf{E} - \mathbf{R}^{\mathrm{T}}\mathbf{B}_\times|$.

This method, however, also neglects the dependencies between the elements of $\mathbf{E}$, and consequently is still very sensitive to errors in the data. The matrix formed from the elements of the vector $\mathbf{e}$ that minimizes $|\mathbf{Ae}|$ is not necessarily close to the product of the matrices $\mathbf{R}$ and $\mathbf{B}_\times$ which correspond to the true motion.

Several researchers have pointed out the problems of computing motion by unconstrained minimization of the error [12], [3]. Horn [3] proposed the most general algorithm to solve the direct nonlinear constrained optimization problem iteratively. This method was later revised and reformulated in [4] using unit quaternions.

The vectors $\mathbf{r}_i$, $\boldsymbol{\ell}_i$, and $\mathbf{b}$ are given in quaternion form by

$$\mathring{\mathrm{r}}_i = (0, \mathbf{r}), \ \mathring{\ell}_i = (0, \boldsymbol{\ell}), \text{ and, } \mathring{\mathrm{b}} = (0, \mathbf{b}) \tag{2.37}$$

while that of $\boldsymbol{\ell}'_i = \mathbf{R}\boldsymbol{\ell}_i$ is given by

$$\mathring{\ell}'_i = \mathring{\mathrm{q}}\mathring{\ell}_i\,\mathring{\mathrm{q}}^* \tag{2.38}$$

Using the identity (A.10) given in Appendix A, the triple product λ_i (2.34) can be written as

$$\begin{aligned} \lambda_i &= \mathring{\mathrm{r}}_i\mathring{\mathrm{b}} \cdot \mathring{\mathrm{q}}\mathring{\ell}_i\,\mathring{\mathrm{q}}^* \\ &= \mathring{\mathrm{r}}_i\mathring{\mathrm{b}}\mathring{\mathrm{q}} \cdot \mathring{\mathrm{q}}\,\mathring{\ell}_i \\ &= \mathring{\mathrm{r}}_i\mathring{\mathrm{d}} \cdot \mathring{\mathrm{q}}\,\mathring{\ell}_i \end{aligned} \tag{2.39}$$

where $\mathring{\mathrm{d}} = \mathring{\mathrm{b}}\mathring{\mathrm{q}}$.

In the latter version of Horn's algorithm, S, the sum of squared errors, is written as a first order perturbation about a given $\mathring{\mathrm{d}}$ and $\mathring{\mathrm{q}}$. The idea is to find the incremental changes $\delta\mathring{\mathrm{d}}$ and $\delta\mathring{\mathrm{q}}$ which minimize the linearized equation subject to the constraints

$$\mathring{\mathrm{q}} \cdot \mathring{\mathrm{q}} = 1, \quad \mathring{\mathrm{d}} \cdot \mathring{\mathrm{d}} = 1, \quad \text{and, } \mathring{\mathrm{q}} \cdot \mathring{\mathrm{d}} = 0 \tag{2.40}$$

The updated vectors $\mathring{\mathrm{q}} + \delta\mathring{\mathrm{q}}$ and $\mathring{\mathrm{d}} + \delta\mathring{\mathrm{d}}$ must also satisfy these conditions and, neglecting second order terms, this results in the incremental constraints

$$\mathring{\mathrm{q}} \cdot \delta\mathring{\mathrm{q}} = 0, \quad \mathring{\mathrm{d}} \cdot \delta\mathring{\mathrm{d}} = 0, \quad \text{and, } \mathring{\mathrm{q}} \cdot \delta\mathring{\mathrm{d}} + \mathring{\mathrm{d}} \cdot \delta\mathring{\mathrm{q}} = 0 \tag{2.41}$$

Differentiating the constrained objective function with respect to $\delta\mathring{\mathrm{q}}$, $\delta\mathring{\mathrm{d}}$, and the Lagrange multipliers λ, μ, ν associated with each of the constraints (2.41) and setting the result to zero results in a linear system of equations of the form

$$\mathbf{J} \begin{pmatrix} \delta\mathring{\mathrm{q}} \\ \delta\mathring{\mathrm{d}} \\ \lambda \\ \mu \\ \nu \end{pmatrix} = \mathbf{h} \tag{2.42}$$

where the matrix $\mathbf{J}$ and the vector $\mathbf{h}$ are both known, given the current value of $\mathring{\mathrm{q}}$ and $\mathring{\mathrm{d}}$ (see [4] for details). Equation (2.42) can thus be solved for the 11 unknowns, which are the four components each of $\delta\mathring{\mathrm{q}}$ and $\delta\mathring{\mathrm{d}}$ and the three Lagrange multipliers. After updating $\mathring{\mathrm{q}}$ and $\mathring{\mathrm{d}}$ with the new increments $\delta\mathring{\mathrm{q}}$ and $\delta\mathring{\mathrm{d}}$, the procedure can be repeated until the percentage change in the total error falls below some limit. This algorithm has been shown to be very accurate and efficient in most cases for estimating motion, even with noisy correspondence data, as long as there are a sufficiently large number of matches. As presented, however, it is too complex to be implemented efficiently on a simple processor, given the need to solve an 11×11 system of equations at each iteration. We will present a simplified adaptation of this algorithm in Chapter 7.

2.3 Correspondenceless Methods

If the displacements in the image are small, they can be approximated by the time derivatives of the position vectors to points in the scene. For small θ, $\cos\theta \sim 1$, $\sin\theta \sim \theta$, and equation (2.17) reduces to

$$\mathbf{R} \approx \mathbf{I} + \theta\mathbf{\Omega}_\times \tag{2.43}$$

The equation of rigid body motion (2.4) then becomes

$$\begin{aligned} \mathbf{p}_r &= \mathbf{R}\mathbf{p}_l + \mathbf{b} \\ &= \mathbf{p}_l + \theta(\hat{\boldsymbol{\omega}} \times \mathbf{p}_l) + \mathbf{b} \end{aligned} \tag{2.44}$$

Let $\dot{\mathbf{p}}$ denote the time derivative of the vector $\mathbf{p}$, which can be approximated as $\dot{\mathbf{p}} = \mathbf{p}_l - \mathbf{p}_r$. From (2.44)

$$\dot{\mathbf{p}} = -\theta(\hat{\boldsymbol{\omega}} \times \mathbf{p}) - \mathbf{b} \tag{2.45}$$

which, expanded into component form, results in

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix} = \begin{pmatrix} \theta(\omega_z Y - \omega_y Z) - b_x \\ \theta(\omega_x Z - \omega_z X) - b_y \\ \theta(\omega_x Y - \omega_y X) - b_z \end{pmatrix} \tag{2.46}$$

Image plane displacements are given by

$$u = \frac{dx}{dt} = x_\ell - x_r, \quad \text{and,} \quad v = \frac{dy}{dt} = y_\ell - y_r \tag{2.47}$$

From the equations of perspective projection (2.1) we have

$$\frac{dx}{dt} = f\frac{d}{dt}\left(\frac{X}{Z}\right) = \frac{f}{Z}\left(\dot{X} - \frac{X}{Z}\dot{Z}\right) \tag{2.48}$$

and

$$\frac{dy}{dt} = \frac{f}{Z}\left(\dot{Y} - \frac{Y}{Z}\dot{Z}\right) \tag{2.49}$$

Combining (2.46) through (2.49), we obtain the equations for the incremental optical flow

$$u = \frac{-fb_x + b_z x}{Z} + \frac{\theta}{f}\left(\omega_x xy - \omega_y(x^2 + f^2) + \omega_z yf\right) \tag{2.50}$$

$$v = \frac{-fb_y + b_z y}{Z} - \frac{\theta}{f}\left(\omega_y xy - \omega_x(y^2 + f^2) + \omega_z xf\right) \tag{2.51}$$

first derived by Longuet-Higgins and Prazdny [31].

If it is assumed that the brightness E of a point in the image does not change as the point moves, then the total derivative of brightness with time must be zero, that is,

$$\begin{aligned} \frac{dE}{dt} = 0 &= \frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} \\ &= E_x u + E_y v + E_t \end{aligned} \tag{2.52}$$

Equation (2.52) is known as the *brightness change constraint equation.*

There are two approaches to using these equations. The first, and earliest proposed, is to compute the optical flow over the entire image and to invert (2.50) and (2.51) to find the global motion and depth at each pixel. The second, known as the direct approach, skips the computation of the optical flow and uses only the constant brightness assumption combined with the incremental rigid body equations. Neither approach requires finding explicit point correspondences.

2.3.1 Optical flow

Horn and Schunck developed the first algorithm for determining optical flow from local image brightness derivatives [32] based on minimizing the error in the brightness change constraint equation

$$\epsilon_b = E_x u + E_y v + E_t \tag{2.53}$$

Since there are two unknowns at each pixel, the constant brightness assumption is not sufficient to determine u and v uniquely and a second constraint is required. Horn and Schunck chose the smoothness of the optical flow and added a second error term

$$\epsilon_s^2 = |\nabla u|^2 + |\nabla v|^2 \tag{2.54}$$

The total error to be minimized is therefore

$$\epsilon_{tot}^2 = \int\int \left(\epsilon_b^2 + \lambda^2\epsilon_s^2\right)\, dx\, dy \tag{2.55}$$

where λ^2 is a penalty term that weights the relative importance of the two constraints. The functions u and v which minimize ϵ_{tot}^2 for a given λ can be found using the calculus of variations.

One problem with computing optical flow by applying a smoothness constraint is that the flow is not smooth at boundaries between objects at different depths. The global optimization procedure causes errors generated at depth discontinuities to propagate to neighboring regions [33]. Segmenting the optical flow at depth discontinuities would appear to be the solution to this problem except that one does not know *a priori* where they are. Murray and Buxton [34] proposed incorporating discontinuities by adding *line processes* to the objective function, using an idea originated by Geman and Geman for segmenting gray-scale images by simulated annealing [35]. The resulting optimization problem is non-convex, however, and requires special procedures to converge to a global minimum energy state.

Once the optical flow is determined it is necessary to solve equations (2.50) and (2.51) to find motion and depth. As was the case for the explicit methods, absolute distances cannot be recovered since scaling Z and $\mathbf{b}$ by the same factor has no effect on u and v. Longuet-Higgins and Prazdny [31] showed how motion and depth parameters could be determined from the first and second derivatives of the optical flow after first computing the location of the epipole. Heeger *et al.* [36] proposed a method to recover the motion by applying rotation insensitive center-surround operators that allow the translational and rotational components of the motion to be determined separately. Ambiguities in interpreting the optical flow in the case of special surfaces have been analyzed in [37], [38], [39], and [40].

2.3.2 Direct methods

The method of Horn and Schunck, or one of its variations, requires a great deal of computation to determine the optical flow—which is only an intermediate step in obtaining the actual parameters of interest. The direct approach of Horn and Weldon [41] and Negahdaripour and Horn [42] avoids computing optical flow by substituting u and v from equations (2.50) and (2.51) directly into (2.52). The brightness change constraint equation is thus expressed as

$$(\mathbf{v} \cdot \widehat{\omega})\theta + \frac{\mathbf{s} \cdot \mathbf{b}}{\mathbf{Z}} = -E_t \tag{2.56}$$

where

$$\mathbf{s} = \begin{pmatrix} -fE_x \\ -fE_y \\ xE_x + yE_y \end{pmatrix} \tag{2.57}$$

and

$$\mathbf{v} = \begin{pmatrix} fE_y + y(xE_x + yE_y)/f \\ -fE_x - x(xE_x + yE_y)/f \\ yE_x - xE_y \end{pmatrix} \tag{2.58}$$

Note that the vectors $\mathbf{s}$ and $\mathbf{v}$ are entirely computable from measurements in the image.

Assuming the image contains N pixels, there are $N + 5$ unknowns in equation (2.56): the five independent parameters of $\mathbf{b}$ and $\hat{\boldsymbol{\omega}}\theta$ (recall that $\mathbf{b}$ is a unit vector due to the scale factor ambiguity), and the N depth values Z. Since there is only one equation (2.56) for each pixel, the problem is mildly underconstrained. Given two images it can be solved only for a few special cases in which either the motion or the surface structure is restricted. With more than two views of the same scene, however, the problem is no longer underconstrained [11]. It should be noted that it is never required to incorporate the assumption that the optical flow is smooth, and hence the problems associated with discontinuities in the flow are avoided.

Several methods have been developed to solve the special cases where the problem is not underconstrained for two views. Three of these were developed by Negahdaripour and Horn who gave a closed form solution for motion with respect to a planar surface [43]; showed how the constraint that depth must be positive could be used to recover translational motion when the rotation is zero, or is known [44]; and derived a method for locating the focus of expansion [45]. Taalebinezhaad [46], [47] showed how motion and depth could be determined in the general case by fixating on a single point in the image. He essentially demonstrated that obtaining one point correspondence would provide enough information to enable the general problem to be solved.

2.3.3 Limitations of correspondenceless methods

Methods for computing motion and depth from the local spatio-temporal derivatives of image brightness must rely on specific assumptions in order to work. The most important of these, on which all of the methods just described are based, is that brightness is constant (2.52). Verri and Poggio [48] criticized differential methods on the grounds that brightness constancy is often violated. Their arguments, however, were based on considering shading effects which are important only for specular surfaces or when the motion is large enough to significantly affect surface orientation. Furthermore, these effects dominate only when the magnitude of the spatial brightness gradient is small. There are clearly cases, such as the rotating uniform sphere or the moving point light source, as pointed out by Horn [49] and others, in which the optical flow and the motion field are different. In areas of the image where the brightness derivatives are small, it is difficult to constrain the motion or to determine depth. However, this problem is not specific to differential methods. Gennert and Negahdaripour [50] investigated the use of a linear transformation model to account for brightness changes due to shading effects on lightly textured surfaces. Their method was applied only to computing optical flow and involved modifying the objective function (2.55) to add new constraints. Direct methods do not lend themselves as easily to relaxing the brightness constancy assumption. With these it is simpler to ignore areas where the spatial derivatives are small.

One of the more important assumptions underlying differential methods is that the interframe motion must be small so that the approximations (2.43)–(2.45) will be valid, and so that the spatial and temporal sampling rates will not violate the Nyquist criterion.

It is useful to perform some sample calculations to see what is meant by "small". The approximations $\sin\theta \sim \theta$, $\cos\theta \sim 1$ are accurate to within 1.5% to about 10^o of rotation. Approximations (2.45) and (2.47) which express the derivatives of the position vector as the difference between the left and right rays, and which incorporate the approximation $\mathbf{R} \approx \mathbf{I} + \theta\mathbf{\Omega}_\times$, are thus reasonable as long as the velocity of the point in the scene is constant between frames and $\theta < 10^o$. These conditions should not be difficult to achieve with video-rate motion sequences. The angular restriction may rule out some binocular stereo arrangements, however.

The primary concern is thus not the validity of the incremental optical flow equations, but whether the sampling rates are high enough to avoid aliasing. The Nyquist criterion which bounds the maximum rate at which an image sequence can be sampled in space and time can be derived as follows.

The constant brightness assumption requires that

$$E_x u + E_y v + E_t = 0 \tag{2.59}$$

If the Fourier transform of $E(x, y, t)$ is given by

$$E(x, y, t) \Longleftrightarrow \mathcal{E}(\xi, \eta, \nu) \tag{2.60}$$

then

$$E_x u + E_y v + E_t \Longleftrightarrow j(\xi u + \eta v + \nu)\mathcal{E}(\xi, \eta, \nu) \tag{2.61}$$

If the constant brightness assumption is valid, then by linearity of the Fourier transform, either

$$j(\xi u + \eta v + \nu) = 0 \tag{2.62}$$

or,

$$|\mathcal{E}(\xi, \eta, \nu)| = 0 \tag{2.63}$$

for all ξ, η, ν.

If $E(x, y)$ is bandlimited so that $|\xi| < \Omega_x$ and $|\eta| < \Omega_y$, then (2.62) requires that $|\nu| < \Omega_t$ where

$$\Omega_t = \Omega_x |u| + \Omega_y |v| \tag{2.64}$$

Note that Ω_x, Ω_y, and Ω_t represent angular frequencies and should not be confused with the matrix $\mathbf{\Omega}_\times$.

To avoid aliasing, the temporal sampling rate τ must satisfy $\tau > 2\Omega_t$. However, τ often cannot be changed, for instance in video sequences where images are produced at a rate of 30 frames/sec.[3] Although it is possible to design video cameras to operate at higher rates, other factors, such as the amount of available light or interframe processing time requirements, may limit how far one can go.

For a given τ, the Nyquist criterion thus imposes a restriction on the maximum image plane displacement which can be tolerated. If the spatial bandwidths Ω_x and Ω_y are approximately the same, so that we can set $\Omega_x = \Omega_y \equiv \Omega_s$, then

$$\tau > 2\Omega_t = 2\Omega_s(|u| + |v|)_{\max} \tag{2.65}$$

[3] According to the American NTSC standard. Other countries outside North America and Japan use the PAL and SECAM standards which produce 25 frames/sec.

or,

$$(|u| + |v|)_{\max} < \frac{\tau}{2\Omega_s} \tag{2.66}$$

The quantities u/τ and v/τ have the units of pixels/frame, while Ω_s has units of 1/pixel. Since the images are spatially sampled, it is also true that $0 \leq |\Omega_s| \leq 1/2$.

If the image is highly textured and contains significant energy in frequencies near the upper limit, the maximum tolerable displacement will be around 1 to 2 pixels per frame. In most cases, disparities in binocular stereo pairs will be greater than this. It is worthwhile to compute some typical displacements in images generated by a moving camera. From (2.50) and (2.51) we can find the optical flow for the following special cases:

1. Pure translation along the $\hat{x}$ direction

$$(u, v) = -\frac{b_x}{Z}(f, 0) \tag{2.67}$$

2. Pure translation along the $\hat{z}$ direction

$$(u, v) = \frac{b_z}{Z}(x, y) \tag{2.68}$$

3. Pure rotation of θ about $\widehat{\boldsymbol{\omega}} = \hat{y}$

$$(u, v) = \frac{\theta}{f}(x^2 + f^2, xy) \tag{2.69}$$

In normal imaging systems the effective focal length f is several hundred times longer than the interpixel spacing. For the purpose of calculating displacements, let $f = 200$ pixels. For the first case, pure translation in the $\hat{x}$ direction, suppose the camera is moving at 30mph (48 km/hr) and viewing an object at a distance of 10m while generating a video sequence at 30 frames/sec. This could be the situation of a camera attached to a car door viewing the side of the road. In 1/30 sec, the camera has moved .444m in the $\hat{x}$ direction. We thus find

$$(u, v) = -\frac{.444}{10}(200, 0) = (-8.9, 0) \tag{2.70}$$

which is considerably larger than the maximum allowed displacement.

In the second case, pure translation along the $\hat{z}$ direction, we can identify the quantity b_z/Z as $1/T$, where T is the time-to-impact of the object being viewed. Suppose $T = 10$

sec and the frame rate is still 1/30 sec, then

$$(u, v) = \frac{1}{300}(x, y) \tag{2.71}$$

For most, or all, of the image, *i.e.,* $x, y < 300$, the displacement will be less than 1 pixel, and hence there should be no problem in applying differential methods to compute the time to crash. (For $T < 10$ sec, the displacements may become too large, but at that point it may be too late to care.)

In the last case, pure rotation about $\hat{y}$, the motion depends only on θ and the position in the image. The smallest displacement occurs at the principal point, $x = 0$, $y = 0$, where

$$(u, v) = \theta(f, 0) \tag{2.72}$$

Every 1^o (.0175 radians) of rotation corresponds to a displacement of 3.5 pixels, with $f = 200$ as before. At 1/30 sec, this corresponds to 30^o per second, which is easily exceeded by ordinary vibrations from moving the camera.

When the displacements are too large for the given frame rate and sensor dimensions, the situation can be remedied by low-pass filtering the image, or equivalently, by reducing the spatial sampling rate. Rewriting (2.66) as

$$\Omega_s < \frac{\tau}{2(|u| + |v|)_{\max}} \tag{2.73}$$

gives the maximum bandwidth for a given sampling rate and maximum image plane displacement.

Unfortunately, it is not possible to know in advance what $(|u| + |v|)_{\max}$ will be. If the resolution is set lower than necessary, useful information is lost, while if it is set too high, aliasing will occur. More insidiously, one cannot determine from the local brightness gradients alone if aliasing has occurred. One solution proposed by Anandan [51] is to perform motion estimation at multiple scales by separating the image into a hierarchical pyramid structure in which each level represents a different spatial bandwidth and sampling rate. Information can thus propagate from coarse to fine levels to determine the highest resolution at which optical flow can be computed from brightness gradients.

The pyramid structure is not the most practical option for hardware implementation as it involves a great deal of processing, and there is not a simple alternative for finding the appropriate filter bandwidth using only gradient information. By combining explicit matching and differential methods, however, it should be possible to devise a reliable system which takes advantage of the best of each. Explicit matching methods can determine the maximum image displacements and compute the motion parameters, while differential methods can more easily obtain information that relates motion and structure from all parts of the image. For example, given the motion and assuming the image has been appropriately lowpass-filtered, depth can be computed from equation (2.56) as

$$Z = -\frac{\mathbf{s} \cdot \mathbf{b}}{\mathbf{E_t} + (\mathbf{v} \cdot \hat{\boldsymbol{\omega}})\theta} \tag{2.74}$$

In this thesis we are primarily concerned with the design of a system to compute relative motion from explicit point correspondences and will not explore further the benefits, or the details, of interfacing to other systems based on different approaches. It should be understood, however, that the choice of an explicit strategy does not rule out its use in conjunction with other methods.

2.4 Motion Vision and VLSI

The tremendous computational complexity of many of the algorithms for determining motion and the need to perform these computations in real time has led to the design of several specialized VLSI systems. Analog processing has been a major component in most of these as it offers the posibility of performing parallel operations on large amounts of data with compact, low-power circuits. In [52], Horn presents the theory and gives several examples of useful computations which can be performed by analog networks.

One of the first circuits was the correlating motion detector of Tanner and Mead [53] which was a simple 1-D detection circuit that allowed a maximum motion of ± 1 pixel. A linear array of photodiodes converted incident light to a 1-bit signal which was compared, via a binary correlation circuit, to the stored signals from the previous cycle. The peak correlation value at each pixel was detected by mutual inhibition among neighboring comparators, and the output was summed on a global bus.

A later design by Tanner implemented a 2-D non-clocked array of photosensors to compute optical flow by gradient descent on a feedback network [54], [55, Chapter 14]. This system was limited to constant flow, as would arise from a pure translation parallel to the image plane. In this case the error function (2.55) to be minimized reduces to

$$\epsilon_{tot}^2 = \int\int (E_x u + E_y v + E_t)^2 \, dx dy \tag{2.75}$$

Since the flow is constant, the problem is solved by taking derivatives with respect to u and v and setting these to zero. We find that

$$\frac{d\epsilon_{tot}^2}{du} = \int\int (E_x u + E_y v + E_t) \, E_x dx dy \tag{2.76}$$

$$\frac{d\epsilon_{tot}^2}{dv} = \int\int (E_x u + E_y v + E_t) \, E_x dx dy \tag{2.77}$$

In the gradient descent approach, currents proportional to quantities in the integrals are fed into a negative feedback loop which drives the variable voltages representing u and v to values which force the derivatives to zero. This circuit was designed to operate in continuous time to avoid temporal aliasing. The first chip built was an 8×8 array with processors at each pixel to compute the local multiplications and two global busses to carry the values of u and v.

Other circuits developed by the Computation and Neural Systems Program group at CalTech are described by Horiuchi *et al.* in [56] where they discuss a comparative study of four experimental designs for 1-D motion estimation. Among these were a 1-D version of Tanner's gradient descent optical flow chip and a fully digital circuit composed of off-the-shelf components to implement correlation. The other two designs were a pulse-coded correlation circuit (based on a model of structures found in the auditory system of owls) which detects time differences between neighboring pulses, and a mixed analog/digital system to track zero-crossings of a difference of Gaussians (DOG) filtered image. In their results, they report that the fully digital circuit, composed of a Fairchild Linear CCD 256 pixel array and a Harris RTX2001A microprocessor, had the best performance in overall robustness, while the Tanner 1-D optical flow chip had the least reliable performance. They also reported difficulties using gradient methods due to the 120Hz flicker found in ordinary room lights.

There has been a great deal of interest, motivated by the desire to reduce interchip communication requirements, in developing one-chip circuits that incorporate photosensing and local processing at each pixel [57]. With focal-plane processing, however, the area taken up by the processing circuitry increases pixel size and therby reduces the maximum array size which can be placed on the chip. Since technology limitations restrict the maximum die size to about 1cm^2, either resolution or field of view must be sacrificed.

Gottardi and Yang [58] recently reported the development of a single chip 1-D motion sensor in CCD/CMOS technology with a 115-pixel linear image sensor and CCD charge subtraction circuits to perform correlation. McQuirk is currently working on a one-chip design for a focus of expansion (FOE) detector using the direct approach developed by Negahdaripour and Horn [45]. The system architecture and results of a preliminary test chip are reported in [59]. In order to obtain a reasonable array size (64×64 in 2μ technology), McQuirk chose not to implement a fully parallel processor array, but to time-multiplex the computation using one processor per column. Instead of the continuous-time gradient descent method performed by Tanner's optical flow chip, this system computes a discrete-time iterative approximation to minimize the associated error function. Results from the final design of the complete 64×64 array chip are not yet available.

The common feature of the above systems is that they deal with only a very limited aspect of the problem. Most assume constant optical flow, and none allow for rotation. Given current technology limitations and the complexity of computing general motion, it is probably safe to conclude that it cannot be done with a single chip design at any time in the forseeable future. One reason for designing simpler subsystems is so that they can be combined to solve more complex problems. As yet, however, no one has built or proposed a complete system which includes the design of specialized processors for computing general motion, or relative orientation, in unrestricted environments.

This is the problem which is addressed in this thesis.

Chapter 3

Matching Points in Images

Having chosen to build the system based on an explicit matching approach, we must now determine the approach for finding the point correspondences. There are several reasons why obtaining accurate and reliable point correspondences is a hard problem. One is that the same features in two different images do not necessarily look the same due to differences in foreshortening. Features which appear in one image may be occluded or outside the field of view in the other, or there may be multiple solutions for matching if there are repeating patterns in the images. The high computational cost of computing similarity measures is an additional drawback to obtaining a large number of accurate matches.

Methods which have been proposed for determining correspondences can be grouped into three broad categories: brightness-based methods, gray-level correlation, and edge-based methods. These differ primarily in the types of features used and in their strategy for solving the problem. Hybrid methods, which combine aspects from each of the approaches, have also been developed; however, these are best understood by examining the major categories individually. In this chapter, I will review the advantages and weaknesses of the different approaches and discuss their practicality for hardware implementation with respect to the goals of the present system.

3.1 Brightness-Based Methods

The idea in brightness-based methods is to avoid explicitly searching for the best match for each pixel by formulating a global minimization problem whose solution gives the relative displacement of every point. These methods are similar to those developed for computing optical flow by minimizing the error in the brightness change constraint equation. In fact, there are only minor differences in the formulation of the two problems. In addition to avoiding search, the two primary advantages to this approach, which are not shared by the correlation and feature-based methods discussed next, are that information from the entire image is used in determining the offsets and that it is possible to obtain a dense set of correspondences, even in areas of the image which lack distinctive features.

The procedure, as described in [49] and [60], consists of assuming that the gray-levels of corresponding points are approximately the same and finding disparity functions $d_H(x, y)$ and $d_V(x, y)$ such that, ideally,

$$E_L(x + \frac{1}{2}d_H(x, y), y + \frac{1}{2}d_V(x, y)) = E_R(x + \frac{1}{2}d_H(x, y), y + \frac{1}{2}d_V(x, y)) \tag{3.1}$$

where $E_L(x, y)$ and $E_R(x, y)$ are the brightness functions associated with the left and right images respectively.

Due to variations and offset between the two sensors, it is not expected that equation (3.1) can be solved exactly. Instead, the desired solution is the one which minimizes an error function composed of different penalty terms. Horn [49, Chapter 13] suggested

$$\epsilon_{tot}^2 = \int\int \epsilon_i^2 + \lambda^2 \epsilon_s^2 dx dy \tag{3.2}$$

where ϵ_i is the error resulting from the failure of (3.1) to hold exactly,

$$\epsilon_i = E_L - E_R \tag{3.3}$$

and ϵ_s^2 represents the departure from smoothness of the disparity functions as measured by the squared Laplacian

$$\epsilon_s^2 = (\nabla^2 d_H)^2 + (\nabla^2 d_V)^2 \tag{3.4}$$

The coefficient λ^2 defines the relative weighting of the two error terms.

Gennert [60] proposed a similar, though more elaborate, energy function which included a multiplicative model for the transformation of brightnesses in the two images

$$E_R \longrightarrow mE_L \tag{3.5}$$

The multiplier m takes into account changes in reflectance due both to changes in albedo and in the orientation of the surface being imaged.

The only significant difference between equations (3.2) and (2.55), the error function for optical flow, is that the constant brightness assumption is not expressed in terms of the spatial and temporal derivatives of brightness. However, the derivatives reappear in the Euler equations which for equation (3.2) are

$$\lambda^2\nabla^2(\nabla^2 d_H) = \frac{1}{2}\left(\frac{\partial E_L}{\partial x}+\frac{\partial E_R}{\partial x}\right)(E_L - E_R) \tag{3.6}$$

$$\lambda^2\nabla^2(\nabla^2 d_V) = \frac{1}{2}\left(\frac{\partial E_L}{\partial y}+\frac{\partial E_R}{\partial y}\right)(E_L - E_R) \tag{3.7}$$

The functions in these equations are evaluated at the points $(x+\frac{1}{2}d_H(x,y), y)$ and $(x-\frac{1}{2}d_H(x,y), y)$ in the left and right images, respectively.

The aliasing problem discussed in Section 2.3.3 thus arises in a different form. If the image is not sufficiently bandlimited, or if good initial values for d_H and d_V are not available, it is unlikely that a minimization procedure based on gradient descent will converge. If the derivatives are evaluated too far away from the correct point, the gradient will not point in the direction of the solution. It would thus be very difficult to implement this method in circuit form using analog networks as was done for optical flow by Tanner [54], and for finding the focus of expansion (FOE) by McQuirk [59]. Furthermore, it should be added that the full power of this method is not needed for computing 3-D motion since a dense set of point correspondences is not required to solve the rigid-body motion equations.

3.2 Gray-level Correlation Techniques

Gray-level correlation merits close attention since it is widely used in commercial applications. The theoretical basis for the use of correlation in determining point matches between two images is the well known result from classical detection and estimation theory that, under certain conditions, an optimum decision rule for detecting the presence of a known signal in an observed noisy waveform can be obtained from the cross-correlation of the signal with the waveform [61]. The decision is based on whether the value of the correlation function is above a threshold determined from either a Bayes cost function or a desired false alarm rate in a Neyman-Pearson test. The position of the maximum correlation value gives the most likely position of the signal, under the hypothesis that it is present.

In a typical procedure, one image is divided into N, possibly overlapping, $M \times M$ blocks. Let i, $1 \leq i \leq N$ denote the ith block and let (j,k) be the coordinates in the first image of the center of the block. The correlation function of the ith image block centered at (j',k') in the second image is given by

$$C_i(j',k') = \sum_l \sum_m E_L(j+l,k+m)E_R(j'+l,k'+m) \tag{3.8}$$

where the indices l and m range in integer steps from $-(M-1)/2$ to $+(M-1)/2$.

The underlying assumption which makes the value of the cross-correlation function a sufficient statistic for testing the presence of a signal, is that the observed waveform is a stationary white Gaussian noise process upon which the signal may, or may not, be superimposed. If the background noise process is nonstationary, but is additive, white and Gaussian, an optimal test can still be formulated, but the detection threshold corresponding to a given false alarm rate will be a function of position and must be computed for each block. If the noise is non-white, a "whitening" filter should be applied before computing the correlation function.

In real images, the background process is generally non-stationary and non-white. When different cameras are used, sensor offset, combined with differences in the illumination of the same object viewed from two different positions, will ensure that the brightness values measured from the same feature will almost never be identical in the two images, even in the absence of other noise. In addition, with either one or two cameras, the background noise—which usually means the other features in the image as well as variations in the number of photons collected—will seldom have zero mean value. Practical methods for eliminating the effects of sensor and illumination differences are to preprocess the image data with a band-pass filter to both remove dc offsets and reduce the variance of high frequency noise, or to compute the normalized correlation coefficient, defined by

$$\rho_i(j',k') = \frac{\frac{1}{M^2}C_i(j',k') - \mu_L(j,k)\mu_R(j',k')}{\sigma_L(j,k)\sigma_R(j',k')} \tag{3.9}$$

where μ_L, μ_R and σ_L, σ_R denote the sample means and standard deviations of E_L and E_R over their respective blocks.

It can be easily verified that the normalized correlation coefficient is unchanged by any linear transformation of the brightness functions, E_L and E_R. The search for the maximum correlation value of the ith block is confined to a pre-specified window W_i of area A in the second image. The total computational cost for determining the best match for all N blocks is therefore $O(NM^2A)$ if the function is evaluated at every offset of each search window. If M is large enough, the total complexity may be reduced using fast Fourier transform (FFT) techniques to $O(NA\log_2 A)$. The cost of preprocessing the images with a band-pass filter is $O(L^2\log_2 L)$ for $L \times L$ images and is $O(L^2M^2)$ for computing the local means and variances needed for the correlation coefficient in (3.9).

The cost of brute force search by computing the correlation function at every offset of a large search window is prohibitively high, even with modern fast processors. Systems which have been designed to perform matching based on gray-level correlation generally implement some intelligent method for reducing the search space. One simple method, suggested by Barnea and Silverman [62], is to use a sub-optimal, but more easily computed, similarity measure such as the sum of absolute values of differences

$$V_i(j',k') = \sum_l \sum_m |E_L(j+l,k+m) - E_R(j'+l,k'+m)| \tag{3.10}$$

with the best match being given by the location of the minimum value of V_i.

A detection test based on the sum of absolute values of differences *is* a computationally efficient approximation to one based on the correlation coefficient, and, under certain conditions, the two are in fact equivalent. If E_L and E_R are quantized to integer values, the absolute value of the difference between two pixel values is always less than or equal to the square of the difference. That is,

$$\begin{aligned} V_i(j',k') &= \sum_l \sum_m |E_L(j+l,k+m) - E_R(j'+l,k'+m)| \\ &\leq \sum_l \sum_m (E_L(j+l,k+m) - E_R(j'+l,k'+m))^2 \end{aligned} \tag{3.11}$$

Using equation (3.9) and the definitions of the sample means and variances

$$\rho = \frac{1}{M^2} \sum_l \sum_m E(j+l,k+m) \tag{3.12}$$

and

$$\sigma^2 = \frac{1}{M^2} \sum_l \sum_m (E(j+l,k+m) - \mu)^2 \tag{3.13}$$

it can be shown that

$$V_i(j', k') \leq M^2 \left[(\sigma_L - \sigma_R)^2 + (\mu_L - \mu_R)^2 + 2\sigma_L \sigma_R (1 - \rho_i) \right] \quad (3.14)$$

Thus the test which accepts a match when $\rho_i > \tau_c$ is equivalent to the test which accepts a match when $V_i < \tau_V = M^2[(\sigma_L - \sigma_R)^2 + (\mu_L - \mu_R)^2 + 2\sigma_L\sigma_R(1 - \tau_c)]$. Also, as long as (μ_L, μ_R) and (σ_L, σ_R) are approximately constant over the search window, the position of the minimum value of V_i will be the same as that of the maximum value of ρ_i.

Many commercial hardware systems for feature detection have been developed based on the measures just described. A few examples among the many currently available systems are: 1) the alignment system developed by Cognex Corporation which uses normalized correlation search [63]. This system, which is contained on a single 340mm×366mm printed-circuit board, along with image capture hardware, frame memory and other interfaces, uses intelligent search strategies and clever programming tricks to achieve high-speed alignment. In a recent brochure, Cognex claims to be able to align a 128×128 pixel template in a 500×400 pixel image in 200 milliseconds. 2.) The real-time image processing and alignment board by Sharp Digital Information Products, Inc., which fits into a personal computer. Using software which runs on the host computer's CPU and which interfaces to *two* special-purpose processor boards, they claim that the system can find a 100×100 template within a 512×512 search area in less than 100 milliseconds. 3.) The MaxVideo 20 system by Datacube, Inc., which is perhaps the most widely used system, interfaces to a VME bus and performs numerous image processing applications, along with alignment. 4.) The STI3220 single-chip motion estimation processor designed by SGS Thompson Microelectronics used to implement the MPEG data compression algorithm at video rates.[1] This chip finds the minimum sum-of-absolute-values-of-differences between two blocks over a maximum displacement of $+7/-8$ pixels in both horizontal and vertical directions.

Some experimental designs have also been based on gray-level correlation. Recently Hakkarainen [65] developed a test system in analog CCD/CMOS technology to compute stereo disparities along a single horizontal row of pixels (parallel epipolar geometry assumed). The matching circuit incorporated a 40×40 pixel absolute-value-of-difference array designed to find candidate matches within a maximum disparity range of 11 pixels.

[1]MPEG is a motion picture compression technique based on coding the offsets between blocks in two frames. See [64] for more details.

Despite the preponderance of gray-level correlation in commercial vision applications and elsewhere, it does have limitations which make it unsuitable for computing general 3-D motion. An important drawback is that it is very sensitive to differences in foreshortening of surfaces which are viewed from different angles. Unless the motion of the cameras is defined by a pure translation parallel to the image plane, and all objects in the scene are at the same depth, the two images will not simply be shifted versions of each other. The conditions for optimality of the correlation-based decision rule no longer hold in the presence of foreshortening since the signal is distorted. The systems just cited were developed for applications in which foreshortening is not a major problem. In industrial settings, the scene structure can be controlled, and the choice of the template to be matched is guided by the user. Furthermore, the parts to be located or aligned are usually confined to a single plane which is held at a fixed orientation to the optical axis. In the MPEG compression algorithm, small offset errors are not very important because the human visual system cannot perceive fine spatial detail in moving image sequences. In more general settings, however, particularly when long baselines are used as is often the case in binocular stereo, foreshortening cannot be neglected.

A second limitation of gray-level correlation is that it is not by itself sufficient for computing reliable matches given arbitrary blocks within an image. The performance of a correlation test depends on the signal-to-noise ratio, which is low if the block does not contain distinctive features. For example, it is very difficult to find the correct offset for a patch of constant or smoothly varying brightness within a larger region also of constant or smoothly varying brightness. Industrial applications avoid this problem by using large, previously selected templates. However, if the blocks are chosen by arbitrarily dividing the image, there is no guarantee that each one can be reliably matched. Since further processing, such as finding edges, is required to determine the distinctiveness of each block, methods which are based on matching the edges themselves are more attractive in general than those based on gray level correlation.

3.3 Edge-Based Methods

Edges, which are locations of rapid and significant change in the image brightness function, are usually caused by changes in the surface reflectance or orientation of the imaged objects. These occur at changes in surface markings as well as at the boundaries of objects

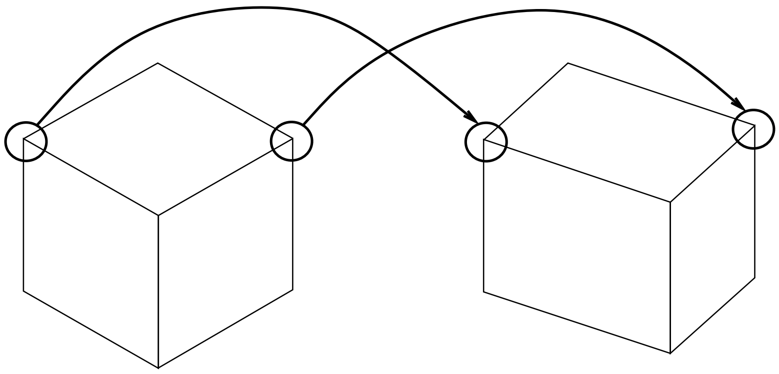


FIGURE 3-1: Example of matching edges despite foreshortening.

and are thus intrinsic characteristics of the scene which transform under the same rules of rigid body motion as the surfaces to which they are associated.

Among the advantages of using edges, as opposed to gray levels, as matching primitives are that they are insensitive to photometric effects and that they are less strongly affected than gray levels by foreshortening. A simple example of the latter issue is shown in Figure 3-1 which depicts a hypothetical perspective transformation of a wire-frame box. Although the lengths of some of the sides change, the edge patterns at the corner points retain enough similarity that they can be uniquely matched between the two views. The often-cited disadvantage of edge-based methods—that they can only generate sparse matches—is a problem for obtaining depth from binocular stereopairs, but not for computing motion.

Edge-based methods can be divided into two categories according to the manner in which they represent edges. Methods in the first category operate directly on the binary image, or *edge map*, produced by the edge detection algorithm, using some form of correlation matching similar to those described for gray level images. Methods in the second category, however, take the output of the edge detector and extract higher-level primitives, such as lines and corners. The attributes of these primitives, *i.e.,* length, end-point coordinates, direction, *etc.,* are then compiled into a symbolic description of the principal features of the image, originally referred to by Marr as the full primal sketch [66]. Matching is then performed by searching the feature space for the best-matching sets of attributes.

There are many possible variations on methods for using the binary representations of edge locations for correlation. Novak [67] lists several different similarity measures and discusses their relative merits. Wong [68] describes different processing techniques that can be applied to the edges to yield more reliable matches. Nishihara [69] developed a binary correlation method based on the sign representation of the image convolved with a Laplacian of Gaussian operator, $\nabla^2 G$, where G, given by

$$G = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.15}$$

is the Gaussian smoothing filter of bandwidth proportional to $1/\sigma$. The zero-crossings of the $\nabla^2 G$-filtered image, which occur at local maxima in the smoothed brightness gradient, were first proposed by Marr and Hildreth [70] as markers for edges. In Nishihara's method, edges are implicitly represented by encoding the sign bit of $\nabla^2 G * I$ rather than by explicitly locating its zero-crossings. He showed that this representation, whose auto-correlation function is sharply peaked at the origin, permits higher resolution disparity measurements than correlation using the values of $\nabla^2 G * I$ itself. This algorithm has been implemented as a stand-alone system designed on a VME bus with a video rate Laplacian-of-Gaussian convolver. The present version of the system allows disparity measurements at an arbitrary image location in approximately 400 microseconds. The sign-correlator algorithm operates on a single 6U (233.4mm $\times$ 160mm) VME bus board and implements 36 parallel correlators that run at a 10MHz pixel rate. The Laplacian-of-Gaussian convolution is performed by a second 6U VME bus board that takes 10MHz digital video input from a Datacube maxbus from the two cameras to produce two 16-bit digital video raster signals. These two boards fit in a VME bus box along with a video digitizer board, a single board computer, and a motor controller board for the camera head.[2]

From the viewpoint of detection theory, binary correlation methods based on the edge maps offers the same advantages as gray-level correlation without several of the disadvantages. As mentioned, edges are less sensitive than surfaces to foreshortening. In addition, it is much easier to test the reliability of matches from edge-based correlation. As will be shown in Chapter 6, reliability is directly related to edge density, which can be determined by simply counting the number of edge pixels in the blocks being compared.

The accuracy of block-correlation techniques, gray-level or binary, is inherently limited,

[2] H. K. Nishihara, personal communication, September 1992

however, by the implicit assumption that the entire block is at the same disparity. The advantage of symbolic matching techniques is that they can provide true point-to-point matching, since corners are matched to corners and line segments to line segments. There are many algorithms which have been developed to perform search on the set of extracted image features. They differ primarily in their choice of attributes and in the constraints which are imposed to reduce both the search space and the number of false matches. Grimson [71] developed a hierarchical method based on a coarse-to-fine matching of zero-crossing contours of the same sign from the image convolved with Laplacian-of-Gaussian operators, $\nabla^2 G$, with different values of σ. He imposed both consistency and figural continuity constraints to limit false matches and to effectively map complete contours. Ayache and Faverjon [72] proposed creating neighborhood graph descriptions from the extracted line segments in each image and then determining the largest connected components of a disparity graph built from the two descriptions. Matches are validated by imposing global continuity constraints and rejecting any connected components with too few members. Fleck [73] recently proposed another variation on these methods by introducing a topological pre-matching filter which provides a stronger test than allowed by consistency and figural continuity constraints.

The primary disadvantage of symbolic matching techniques with respect to the design goals of the 3-D motion system is that they cannot both be easily implemented in simple hardware and be expected to operate at video rates. Reducing the edges to primary features and building the symbolic descriptions requires processing and memory resources that are beyond the capabilities of single-chip systems. Binary correlation, on the other hand, can be implemented relatively cheaply. The primary expense is not in computing the correlation measure, which requires much less hardware than if gray levels are used, but in initially computing the edge maps. In the sign-correlation system developed by Nishihara, for example, only one 6U VME board is devoted to the actual correlation operation, while two boards and a Datacube image processor are required for capturing, digitizing, and filtering the images.

As the overview in this chapter has shown, there is no simple method that can provide accurate and reliable point correspondences in all situations. The procedure which is best suited to the present system is the one that provides the best tradeoff between the requirements for accuracy and simplicity. Among the different approaches for determining point correspondences, the block matching procedures based on computing similarity measures between edges are the simplest to implement in hardware. Since edges can be represented by binary values, computations can be performed as boolean operations. Furthermore, the number of edges in each block provides an easily computed measure of the reliability of the match. The primary drawback of block matching procedures is their limited accuracy due to the assumption that the entire block is at the same depth. We will have to ensure that enough matches are found so that on average the error will go to zero in order for the motion algorithm to compute accurate estimates of the camera motion.

Chapter 4

System Architecture

We can now formulate a plan for the architecture of a system to compute 3-D motion with specialized analog and digital VLSI processors based on the diagram of Figure 4-1. The two input images acquired at the different camera positions will be referred to as *left* and *right*, regardless of whether this terminology reflects their true spatial disposition. If the images are acquired by the same camera, the two input blocks should be considered as memory buffers.

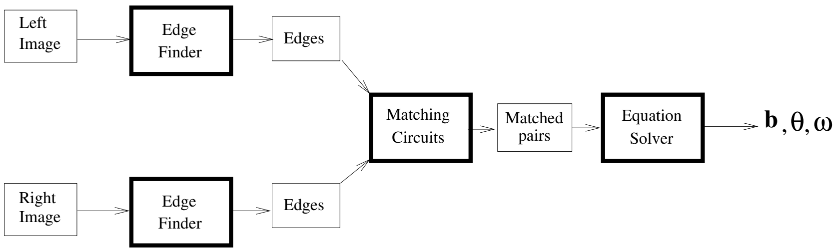


FIGURE 4-1: System block diagram

Two major trends can be discerned in observing the processing and data flow shown in this diagram. The first is that the amount of data decreases significantly from left to right, from the thousands of pixels in the input images, to the reduced binary edge maps, and then to the set of point correspondences that are used to compute the few numbers which characterize the motion. The second trend, however, is that computational complexity increases just as significantly in the same direction. The mix of analog and digital processing which is the most power- and area-efficient for a given task is largely determined by the ratio of data to the complexity of the operations performed on it.

Working backwards from right to left, we see that the ultimate goal of the processing performed on the two images is to extract a set of point correpondences which can be used in the final stage to determine the motion. Given the complexity of solving the motion equations, it is clear that standard digital processing techniques are required. Any one of the many currently available powerful microprocessors, such as the TI TMS320C40, or the Motorola 68040, can certainly do the job. However, since the ultimate goal is to build a low-power system, we need to use the simplest processor that is adequate for the task. In this thesis, I did not attempt to design a minimal custom digital circuit to solve the motion equations. However, I will show, in Chapter 7, how the complexity of the motion algorithm can be reduced so that it can be implemented on a low-end processor or microcontroller.

The set of point correspondences which are fed into the motion algorithm are best found by matching edges using binary correlation, as was concluded at the end of Chapter 3. In the second processing stage we can build a pipelined array of matching circuits, each of which computes for a specific patch in one edge map the translational offset which brings it into alignment with the most similar patch in the second edge map. The search is restricted to a predefined area whose dimensions should be user-controllable according to the application. Since the search window may need to be quite large if the baseline between the two camera positions is long or if any amount of rotation is involved, it is necessary to use a scoring method which has a very low false-alarm rate. Given that repeating patterns frequently occur in real scenes, a similarity measure alone is not sufficient to achieve an acceptable error rate. In Chapter 6, I will present the scoring method to be implemented by the matching circuits as well as discuss the tests which are included in the decision rule to minimize the number of false matches.

Because the edge signals are binary, computing the scores for each offset requires relatively little circuitry and can be easily done in digital logic. Tallying the scores and determining the best match, however, are considerably more complex. In Part III of this thesis, I will describe the design of a mixed analog and digital circuit for finding the best match and compare it to a purely digital implementation.

Edge detection is performed in the first stage of the motion system by operating directly on the signals acquired by the photosensors. Here, there is a tremendous amount of data to be processed, but the operations involved, which are computing local averages and differencing neighboring pixel values, are relatively simple. We thus have a situation where analog processing can offer significant advantages over standard digital methods. Detecting edges efficiently in analog VLSI, however, requires an algorithm which is adapted to the technology, unlike most standard methods which were designed for digital computers. The multi-scale veto (MSV) algorithm, presented in the next chapter, was thus developed to take advantage of operations which are easily performed in analog, while avoiding those that are not. Part II of this thesis is devoted to describing the design, fabrication and testing of a CCD-CMOS edge detector implementing the multi-scale veto algorithm, which is a prototype of one that could be used in the 3-D motion system presented above.

Chapter 5

An Algorithm for an Analog Edge Detector

The bulk of hardware resources in almost all image processing systems is dedicated to data storage and to the initial operations performed on the brightness values acquired by the sensors. One image typically contains several tens to hundreds of thousands of pixels, each usually digitized to 8 bits. Even simple computations, such as adding and subtracting pixel values require substantial processing due to the large amount of data involved. Given that relatively low precision is required, however, there is a clear opportunity for analog circuits to perform many of the initial processing tasks on the image data. Analog circuits which are specifically designed for a task can perform arithmetic and logic operations with 6–8 bits precision in much less area than an equivalent digital implementation. Furthermore, by remaining in the analog domain, there is no need to digitize the signal from the sensor before it can be processed.

The multi-scale veto (MSV) algorithm described in this chapter was developed to solve two problems. The first was that we needed an edge detection algorithm which could be efficiently implemented on a fully parallel analog processor. The second was that we also needed an algorithm to accurately localize edges without being overly sensitive to noise. In this chapter I will discuss how both problems were addressed, presenting first some background on classical edge detection methods to explain why it was decided to develop a new method rather than to encode an existing one into a circuit design. I will also introduce, at a conceptual level, circuit models for implementing the MSV algorithm. The more detailed design description will be saved, however, for part II where the actual prototype processor which was built based on these models is presented. In the final section, I will present results from simulating the algorithm on a pair of image sequences which will be seen again in Chapters 6 and 7 to demonstrate the matching procedure and the motion algorithm.

It should be noted that most of this chapter is derived from a previously published paper [74] in which the multi-scale veto algorithm was first described.

5.1 The Multi-Scale Veto Rule

The problem of edge detection is to find and mark locations of significant change in the image brightness function that are due to changes in the reflectance of objects in the scene, while ignoring any changes caused by high spatial frequency features attributable to noise. 'Noise' is not a well-defined term, however, as it is used to refer both to random fluctuations in the number of photons collected as well as to small-scale 'unimportant' features in the image. Noise can be removed by applying a linear low-pass smoothing filter. However, this has the effect of attenuating all high frequency components indiscriminately and introducing uncertainty in the edge locations. Nonlinear methods, such as median filtering, which preserve important edges and remove noise are also possible. However, these require more computation than linear filtering and generally cannot be implemented by convolution. The MSV algorithm was designed to overcome the problems associated with standard linear filtering methods. Its circuit implementation is conceptually straightforward, and it incorporates a simple procedure for the user to select the types of features which are to be defined as noise.

In the MSV algorithm, edges are defined as sharp changes in brightness which are significant over a range of spatial scales. In order to test for the presence of an edge, a sequence of low-pass filters of decreasing bandwidth is applied to the image, and the differences between the smoothed brightness values of neighboring pixels are computed. An edge exists between two pixels if the difference in their values is above a threshold, which is specified for each filter, at all levels of smoothing. If the threshold test is failed for any filter, the edge is vetoed.

The rationale behind the multi-scale veto method can be explained by observing how it treats different types of features. Let $x_k[m,n]$ denote an array of sampled brightnesses which has been convolved with the kth low-pass filter. Let $y_k[m,n] = x_k[m,n] - x_k[m,n-1]$ denote the differences in the smoothed brightnesses in the direction of the second coordinate, and $G_k[m,n]$ the attenuation of the difference signal, such that $y_k[m,n] = G_k[m,n]y_0[m,n]$. The filters are ordered in decreasing bandwidth so that $G_k[m,n] > G_{k+1}[m,n]$. Let τ_k denote the threshold for the kth filter, and suppose that there is an abrupt change in brightness between $n = 0$ and $n = -1$ such that $y_0[m,0] = A$, with A a positive constant.

Formally stated, an edge will be marked at $n = 0$ only if

$$A > \frac{\tau_k}{G_k[m,0]} \tag{5.1}$$

for $k = 0, \ldots, N-1$, where N is the number of filters applied.

An example is illustrated in Figure 5-1 for the cases of an ideal step edge and an isolated noise spike. The step edge is marked at $n = 0$ because the differences $y_0[m,0]$ and $y_k[m,0]$ are both above threshold. No other locations are marked, although $y_k[m,n] \neq 0$ in general, and for some n may even be greater than τ_k, because $y_0[m,n] = 0$ for all $n \neq 0$. Hence the unsmoothed differences will veto the marking of an edge everywhere except at $n = 0$. In the case of the isolated noise spike, the difference at $n = 0$, which is of the same magnitude as for the step, passes the threshold test for the unsmoothed data. However, it fails for the smoothed data since the isolated spike is attenuated more strongly than the step edge, and hence no edge is marked. In general, it may be observed that while the bandwidth and threshold of the narrowest-band, or largest scale, filter determines the effectiveness with which noise and small features are removed, the widest-band, or smallest scale, filter determines the accuracy with which edges are localized.

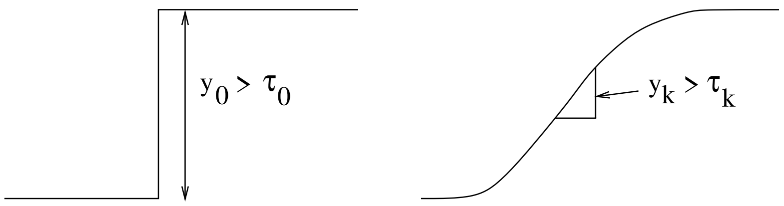
The idea of using multiple scales in edge detection is not new. It is the following features which distinguish the MSV algorithm from conventional methods.

- Edges are *not* defined as local maxima in the magnitude of the gradient, or equivalently, as zero-crossings of the second derivative. Hence computation of second differences is unnecessary.

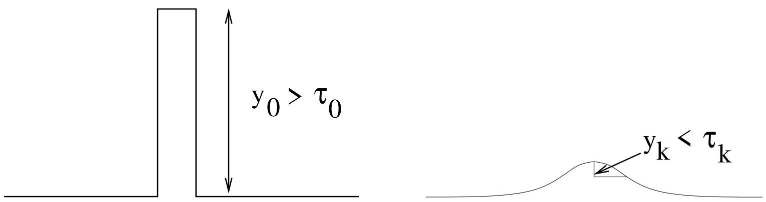
- All of the difference operations and threshold tests at different scales can be performed on the same physical network.

Both features represent a considerable savings in circuitry, which is crucial if the network is to be designed for large image arrays.

By definition, edges exist *between* two pixels on a discrete two-dimensional array. However, to avoid redefining the image grid, their locations are indicated in the output of the edge detection network by setting a binary flag at the locations of the pixels between which



(a) Response to an ideal step. Differences y_0 and y_k are both above threshold at $n = 0$. The edge is marked at the point of change in the unsmoothed data.



(b) Response to ideal point noise. The difference y_0 in the unsmoothed image is above threshold τ_0 at $n = 0$; but the difference y_k in the smoothed image is below the threshold τ_k. Hence no edge is marked.

FIGURE 5-1: Results of applying the multi-scale veto rule to an ideal step edge and to an impulse.

they occur. To simplify the discussion these marked pixels will be referred to as *edge pixels*; although a more exact term would be *edge-adjacent* pixels.

It should be noted that a significant consequence of defining edges by the multi-scale veto rule is that a very good visual approximation to the original image can be reconstructed from the (possibly smoothed) brightnesses at the edge pixels. This operation can be performed by a second processor which recomputes brightness values for non-edge locations by interpolation from the values at the edge pixels. It is thus possible to recover a smoothed version of the original image from noise-corrupted input while maintaining important high frequency information. A more complete analysis of this aspect of the MSV algorithm, along with examples of reconstructed noisy images can be found in the original paper [74].

5.2 Other Methods and the Use of Multiple Scales

In most work in computer vision, edges are defined as the loci of maxima in the first derivative of brightness, and as such can be detected from zero-crossings in the second derivative. This is the basis on which many edge and line detectors, such as the Marr-Hildreth Laplacian-of-Gaussian (LOG) filter [70], the Canny edge detector [75], and the Binford-Horn line finder [76], have been designed. The problem of finding edges by the numerical differentiation of images, however, is ill-posed [77]. Small amounts of noise, which are amplified by differentiation, can displace the zero-crossings or introduce spurious ones. A low-pass smoothing, or regularization, filter must be applied to stabilize the solution.

The issue of scale arises because features in the image generally occur over a range of spatial scales. By varying the passband of the smoothing filter, one can select the size of the features which give rise to edges. Unfortunately, the information which permits the edge to be accurately localized to the feature which produced it is thrown out with the high frequency components. Marr and Hildreth first proposed finding edges from the coincident zero-crossings of different sized LOG filters. Witken [78] introduced the notion of scale-space filtering, in which the zero-crossings of the LOG are tracked as they move with scale changes. These methods are a form of multi-scale veto, but the complexity of tracking the zero-crossings makes them ill-suited for implementation in specialized VLSI.

An alternative solution to removing noise while retaining the high frequency information associated with large scale features is to apply nonlinear filtering. The median filter [79], for example, has long been used in image processing because it is particularly effective in removing impulse, or 'salt-and-pepper', noise. An approach put forward in recent years is the idea of edge detection, or more precisely image segmentation, as a problem in minimizing energy functionals. The first proposal of this nature was the Markov Random Field (MRF) model of Geman and Geman [35]. In an MRF the minimum energy state is the maximum *a posteriori* (MAP) estimate of the energies at each node of a discrete lattice. The MAP estimate corresponds to a given configuration of neighborhoods of interaction. 'Line processes' are introduced on the lattice to inhibit interaction between nodes which have significantly different prior energies, thereby maintaining these differences in the final solution.

Mumford and Shah [80] studied the energy minimization problem reformulated in terms of deterministic functionals to be minimized by a variational approach. Specifically, they proposed finding optimal approximations of a general function $d(x, y)$, representing the data, by differentiable functions $u(x, y)$ that minimize

$$J(u, \Gamma) = \mu^2 \int\int_R (u - d)^2 dx dy + \int\int_{R-\Gamma} |\nabla u|^2 dx dy + \nu|\Gamma| \tag{5.2}$$

where $u(x, y)$ is a piecewise smooth approximation to the original image and Γ is a closed set of singular points, in effect the edges, at which u is allowed to be discontinuous. The coefficients μ^2 and ν are the weights on the different penalty terms.

Blake and Zisserman [81] referred to (5.2) as the 'weak membrane' model, since $J(u, \Gamma)$ resembles the potential energy function of an elastic membrane which is allowed to break in some places in order to achieve a lower energy state. If Γ is known, the solution to the minimization problem can be found directly from the calculus of variatons. However, the problem is to find both Γ and $u(x, y)$ by trading off the closeness of the approximation and the number of discontinuities in the set. As a result the energy function $J(u, \Gamma)$ is nonconvex and possesses many local stationary states that do not correspond to the global minimum. Blake and Zisserman were able to circumvent the problem of multiple local minima by developing a continuation method to solve the minimization problem iteratively.

The weak membrane model was one of the first methods to be implemented in analog VLSI. Digital circuits for performing Gaussian convolution and edge detection began appearing in the early 1980's [82], [83]. The possibility of performing segmentation and smoothing with analog circuitry, however, did not seem practical until the problem had been posed in terms of a physical model. Harris [84] developed the first CMOS *resistive* *fuse* circuit, which is a two-terminal nonlinear element that for small voltages behaves as a linear resistor, but 'breaks' if the voltage across its terminals becomes too large. Several implementations of resistive fuse networks have since been built to compute the minimization of the discrete form of equation (5.2) [57], [85]. Keast [86] developed a discrete-time version of the weak membrane model using CCDs to perform smoothing.

Circuit implementations of the weak membrane model cannot escape the non-convexity problem, however, and some effort is required to push them to the globally optimum solution. The MSV model is similar to the weak membrane in that it also assumes an image can be approximated by a collection of piecewise smooth functions. It is different, however, in that it does not formulate edge detection as an energy minimization problem. The operations of edge detection and image reconstruction are completely separate and independent functions, so that there is no feedback coupling to generate alternate local minima. Hence for any image and given set of parameters, there is a unique set of edges which will be found.

It should be noted that the edges produced by the MSV network are not as 'refined' as those produced by more complex methods such as Canny's edge detector [75]. This is in part due to the way edges are defined. Since many feature boundaries are more like ramps than step edges, the MSV edges are often several pixels thick. It is also due to the need to make the circuitry as simple as possible in order to minimize silicon area. It is not easy to implement contour filling or thinning algorithms with simple circuits. The edges produced by the MSV algorithm are nonetheless functionally useful for many early vision tasks, and in particular, they will be shown to be useful for feature matching.

5.3 Circuit Models

It is not necessary to build a multi-layered processor in order to implement the multi-scale veto rule. By including time as a dimension, a single smoothing network with a controllable space constant can be used. It is well known that resistive grids, such as the one shown in Figure 5-2, can compute an analog smoothing function. The network shown is one-dimensional; however, it can be easily extended to two dimensions by connecting it via transverse resistors to parallel 1-D networks. By equating the current through the vertical resistors connected to the node voltage sources d_i to the sum of the currents leaving the

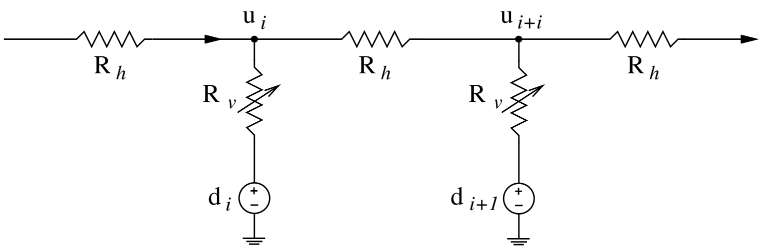


FIGURE 5-2: 1-D resistive smoothing network with controllable space constant.

node through the horizontal resistors, one arrives at the resistive grid equation:

$$u_i - \frac{R_v}{R_h}\sum_k (u_k - u_i) = d_i \tag{5.3}$$

where the subscript k is an index over the nearest neighbors of node i. The continuous 2-D approximation to this circuit is the diffusion equation

$$u - \lambda^2 \nabla^2 u = d \tag{5.4}$$

with

$$\lambda = \sqrt{\frac{R_v}{R_h}} \tag{5.5}$$

which is the space constant, or characteristic length, over which a point source input will be smoothed. By varying the values of R_v, it is therefore possible to control the bandwidth of the effective low-pass filter applied to the data.

A practical way to build a controllable smoothing network is to simulate the resistors with charge-coupled devices (CCDs). CCDs are best known for their role as image sensors, but they are also capable of performing more advanced signal processing. CCDs operate in the manner of a bucket brigade, where the 'buckets' are potential wells under polysilicon gates, and the depths of the wells are determined by the voltages applied to the gates. The 'water' in the buckets is the signal charge which can be transferred, mixed, and separated between the potential wells by varying the sequences of the clock phases which drive the gates. CCDs are built by juxtaposing gates of alternating layers of polysilicon. When used as image sensors, the gates are held at a high potential to collect the charge that naturally

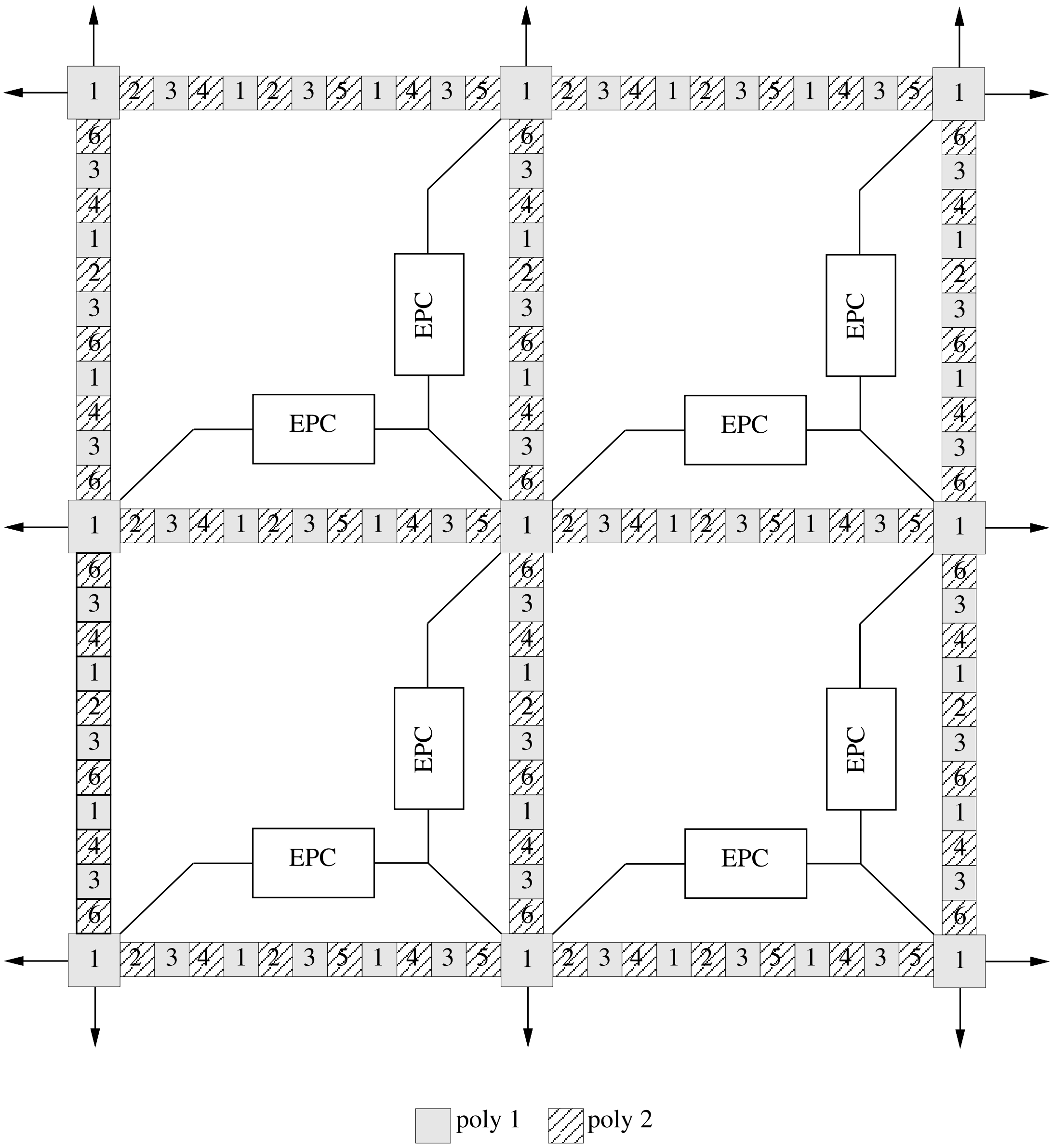


FIGURE 5-3: 2-D MSV edge detection array using charge-coupled devices (CCDs)

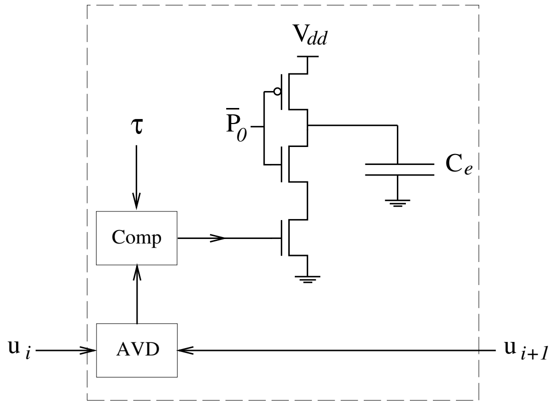


FIGURE 5-4: Conceptual model of the 'Edge precharge circuit'.

occurs when photons of energy greater than the bandgap of silicon hit the device and create electron-hole pairs. After a suitable integration time, generally a millisecond or so, the CCDs then function as analog shift registers to move the signal charges out of the camera.

The basic layout of the 2-D network required to use CCDs for the MSV edge detection algorithm is shown in Figure 5-3. It consists of a grid of orthogonal horizontal and vertical transfer channels with circuitry placed between the nodes to compute differences and perform the threshold tests. The numbers on the gates signify the different clock phases which are used to move signal charges in the array. The structure of this network is the same as that developed by Keast to implement a CCD 'resistive fuse' network [86], [57]. By appropriately sequencing the clock phases, this array can perform smoothing by averaging the signal charge held under each node with each of its neighbors. Specifically, it applies the convolution kernel

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.6}$$

to the image signal with each smoothing cycle. After two cycles the image has been effectively convolved with

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{5.7}$$

and so on. The bandwidth of the smoothing filter is thus controlled by the number of cycles performed.

The primary difference between Keast's design and the MSV network is in the functions performed by the circuits placed between the nodes. The multi-scale veto rule is implemented by the *edge precharge circuits*, shown in Figure 5-4 and indicated by the boxes labeled EPC in Figure 5-3. In each of these, a capacitor is initially charged with an 'edge' signal. At each smoothing cycle, the absolute value of the difference between the node voltages is compared to a threshold; and if the threshold is greater, the capacitor is discharged.

The complete execution of the multi-scale veto algorithm consists of the following steps: The array is initialized by transferring signal charge proportional to image brightness under each node gate (pixel) and by charging the edge capacitors. The signal charge is formed either by direct acquisition using the CCD array, or by loading the pixel values from an off-chip sensor. Several smoothing cycles, ∼5–10, with the accompanying threshold tests, are then performed.

When these are completed, the edge charges from the four precharge circuits connected to each node are tested; and, if any of them is non-zero, *i.e.,* if an edge was detected between the node and one of its four neighbors, a binary value is set at the node to indicate that it is an edge pixel.

5.4 Choosing the Parameters

It might seem that the number of free parameters—the different thresholds for each smoothing filter, as well as the number of smoothing cycles—that need to be specified in order to apply the multi-scale veto rule would make the method impractical or even arbitrary. However, there are simple ways to choose the parameters based on the types of features which one wishes to retain.

The edges which are marked by the edge detection network are those which pass the

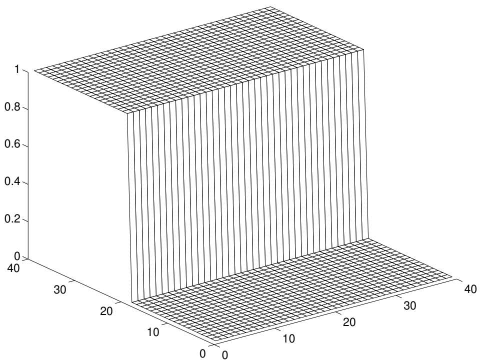
Attenuation Factors					
Smoothing Cycle:	1	2	3	4	5
Horizontal step edge	0.500	0.375	0.313	0.273	0.246
Diagonal step edge	0.375	0.273	0.226	0.196	0.176
Horizontal 1-pixel line	0.250	0.125	0.078	0.055	0.041
Diagonal 1-pixel line	0.125	0.055	0.032	0.022	0.016
Horizontal 2-pixel line	0.500	0.313	0.219	0.164	0.129
Diagonal 2-pixel line	0.313	0.164	0.105	0.074	0.056
1-pixel impulse noise	0.125	0.047	0.024	0.015	0.010
4-pixel square impulse	0.375	0.195	0.120	0.081	0.058
Horizontal 3-pixel ramp	0.750	0.688	0.641	0.602	0.568

Table 5.1: Attenuation factors for different types of features as a function of smoothing

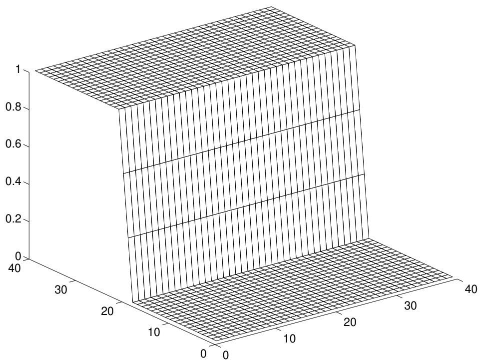
threshold test at all smoothing cycles. Following the same notation used in the example given earlier, let k denote the number of smoothing cycles performed, and let τ_k denote the threshold for the kth cycle. Given the convolution kernel (5.6) which is implemented by the smoothing network, at each cycle, the attenuation factors, G_k for the difference signals corresponding to several idealized features are computed as a function of smoothing and given in Table 5.1.

The ideal step edge refers to a two-dimensional feature which is infinite in one dimension but has an abrupt change from one pixel to the next in the other dimension. The ideal line corresponds to back-to-back step edges facing in opposite directions so that its 1-D cross-section resembles that of the impulse in Figure 5-1. The labels '1-pixel line' and '2-pixel line' in Table 5.1 refer to the width of the 1-D impulse. Impulse noise is a local abrupt change in brightness which is finite in both dimensions. Here, the labels '1-pixel impulse' and '4-pixel square impulse' refer to the area of the local discontinuity. Finally, the ideal ramp is an feature similar to a step edge, but for which the change in brightness occurs over several pixels (in this case 3) rather than abruptly. Some graphic examples of these features are shown in Figure 5-5.

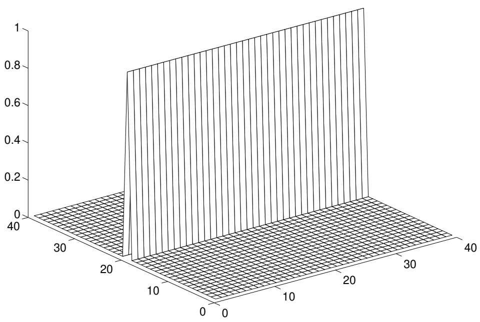
We also distinguish between horizontal features, which are those that are aligned with the rectangular pixel grid, while diagonal ones are oriented at 45^{o} with respect to the grid. It can be seen from the values in the table that diagonal features are attenuated somewhat more than horizontal ones due to the nature of the smoothing operator, and consequently, edges aligned with the grid are favored over skewed edges. An isotropic operator could



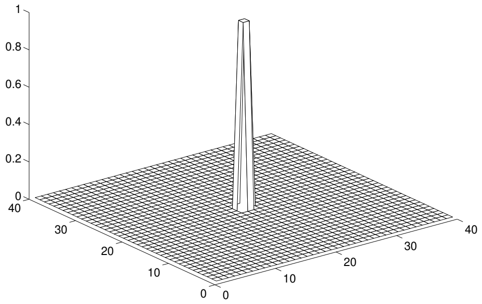
a.) Horizontal step edge



b.) Horizontal 3-pixel ramp



c.) Horizontal 1-pixel line



d.) 4-pixel square impulse

FIGURE 5-5: Ideal 2-D image features.

be implemented with a hexagonally connected network. However, based on the numerous simulations performed to produce edges for the matching algorithm, the added complexity in the design does not seem to be warranted by the slight improvement in the results that an isotropic operator would provide.

The values in Table 5.1 can be used as a guide for setting the parameters for more general types of features. As a specific example, suppose we want to retain only the boundaries from large objects in the image and remove all small scale features. The threshold τ_0 can be set as a function of the contrast in the image. We can perform 5 smoothing cycles and set $\tau_5 = .246\tau_0$. Features resembling step changes in brightness which passed the threshold at $k = 0$ will have little trouble passing the test at $k = 5$, while features resembling 4-pixel square impulses will need to have an original difference greater than $4.2\tau_0$ in order to pass. A simpler method to generate all the thresholds is to choose one idealized feature as a model and to compute

$$\tau_k = G_{k,f}\tau_0 \tag{5.8}$$

where $G_{k,f}$ is the attenuation factor for the model feature at the kth smoothing cycle. For the previous example, the model used was the horizontal step edge. In another case in which we only want to eliminate impulse noise while retaining thin lines, we might choose the diagonal 2-pixel line as a model. In an actual implementation, the values in Table 5.1 can be held in a ROM and supplied to the MSV processor at each smoothing cycle.

5.5 Simulations

The results of simulating the MSV algorithm on four images are shown in Figures 5-6 and 5-7. These images are from two motion sequences, one simulated and one real, which will be used again in the following chapters to demonstrate the matching procedure and the results of the motion algorithm.

In the first sequence, Figure 5-6, the left image is a picture of a poster (of Neil Armstrong) taken by a Panasonic CCD camera, while the right image was generated by a computer-simulated motion applied to the first. The reason for generating simulated motion is to be able to test the results of the motion computation against known values. The motion simulation program assumes an image of a planar surface at a user-supplied depth and orientation. The focal length, principal point, and x, y pixel spacing are input to the program to compute ray directions using the pinhole camera model. For the astronaut images, which are 400×400 pixels, the focal length and pixel spacing were such that the effective field of view, measured from the optical axis, was 55^o. In the right image, the surface is modeled as a frontal plane (parallel to the image plane) at a depth of 10 (baseline) units, and the motion was a translation of 1 unit in the positive $\hat{x}$ direction with a 5^o rotation about the $\hat{y}$ axis.

In the second sequence, Figure 5-7, both images were taken by a Cohu digital CCD camera rigidly mounted on a movable carriage that could be translated along a fixed rail. The carriage assembly could be rotated on both the vertical and horizontal axes so that the camera could be oriented in any direction, with positional accuracy of better than $.1^o$, on each axis. The advantage of the digital camera is that each pixel corresponds exactly to one sensor location, and there is no frame grabber in the path to resample and resize the data. The internal calibration matrix is thus very close to that given by the geometry of the image sensor, which is a 6.4mm×4.8mm CCD array with 756 (horizontal) and 484 (vertical) pixels. A 4.8mm lens was used to give an approximately 40^o field of view, measured from the optical axis. The motion for the pair of images shown was a translation in the $\hat{x}$ direction followed by a rotation of 5^o about the $\hat{z}$ axis. It should be emphasized, however, that this corresponds to the motion of the camera with respect to the motion stage coordinate system and not with respect to its own coordinate system.

In applying the edge detection algorithm, step edge models were used for both sets of images, and 7 smoothing cycles were applied. The results are shown as the binary images below the originals. It should be noted that the apparent thickness of the edges is due in part to the method of marking both pixels on either side of the change in brightness, and in part to the presence of many brightness gradations (ramp-like edges) in the scenes. We will continue with these same image sequences in the following chapters for testing the matching procedure and the motion algorithm.

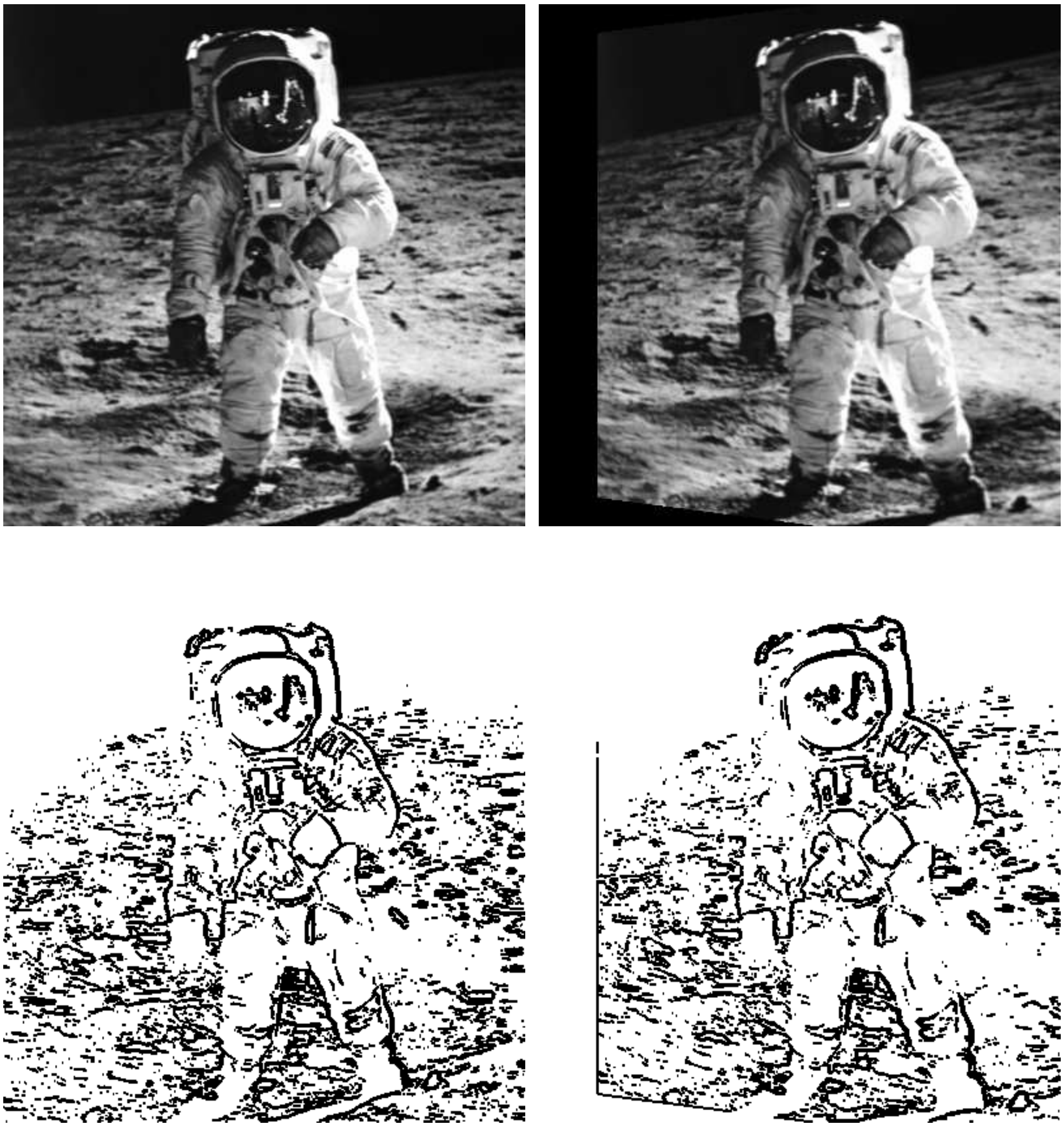


FIGURE 5-6: Simulated motion sequence: Frontal plane at 10 (baseline) units. Motion is given by: $\mathbf{b} = (1, 0, 0)$, $\theta = 5^o$, $\hat{\omega} = (0, 1, 0)$.

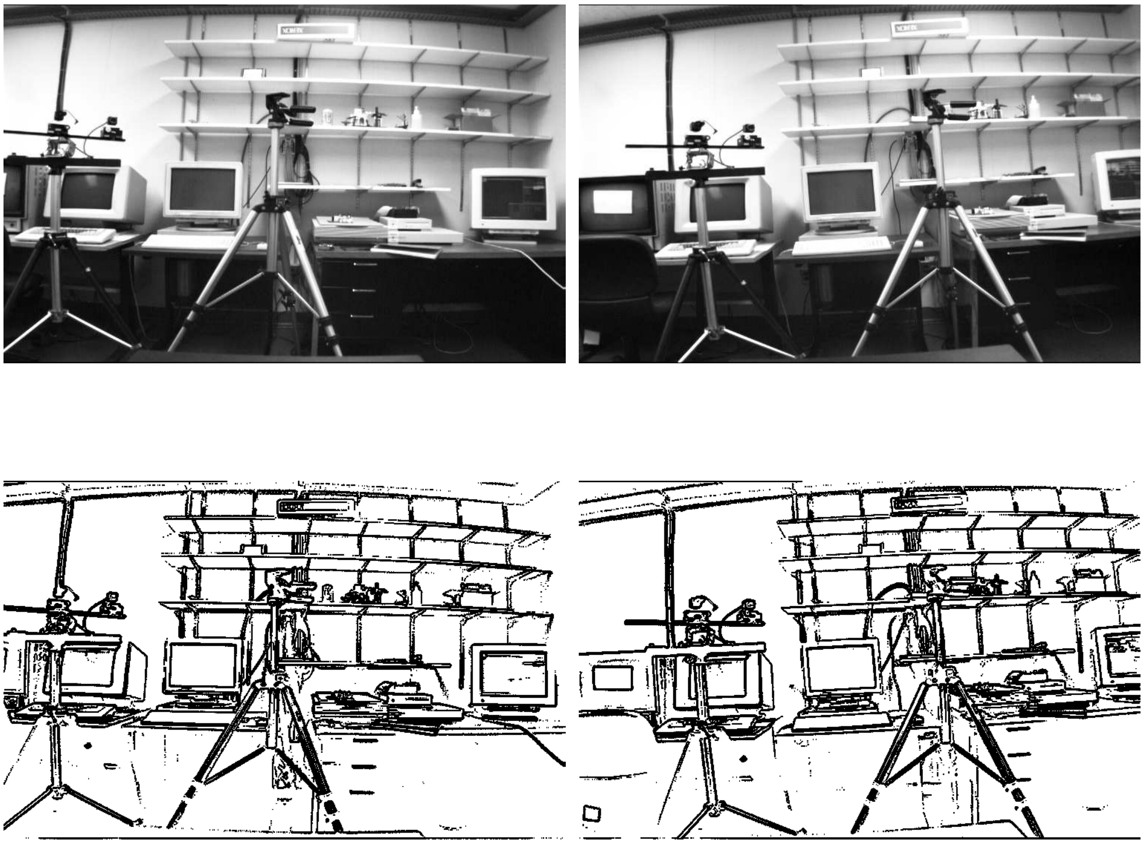


FIGURE 5-7: Real motion sequence: Motion with respect to motion stage coordinate system (not camera system): $\mathbf{b} = (1, 0, 0)$, $\theta = 5^o$, $\hat{\omega} = (0, 0, 1)$.

Chapter 6

The Matching Procedure

There are two important aspects to the problem of finding the set of point correspondences needed for computing motion. The first is that only a sparse set of very reliable matches distributed across the field of view is needed. The second is that, since there is no fixed relation between the relative positions of the corresponding points in the left and right images, as in the case when the epipolar geometry is known, the search must usually be conducted over a large area. Consequently, the matching procedure must have both a good detection rate as well as a very low false alarm rate to minimize the number of incorrect matches.

In the first part of this chapter, I will derive the basic procedure which will be used and show how thresholds can be set to ensure adequate detection and false alarm rates. I will then present the results of applying the procedure to the edge maps from the sequences shown in the previous chapter.

6.1 Finding the Best Match

Following the usual procedure for block matching, we divide one image into N, possibly overlapping, $M \times M$ blocks. Since the edge maps are binary, we can simplify the equations by adopting the following notation. Let i, $1 \le i \le N$ denote the ith block and define

$P \equiv$ total number of pixels in each block $= M^2$

$B_i \equiv$ set of pixels corresponding to an edge in the ith block of the left, or *base*, edge map.

$\overline{B}_i \equiv$ the set complement of B_i within the ith block.

$S_{jk} \equiv$ set of edge pixels in the $M \times M$ block centered at coordinate location (j, k) in the right, or *second*, edge map.

$\overline{S}_{jk} \equiv$ the set complement of S_{jk}.

When it is convenient, the lowercase variables s, $\overline{s}$, b, $\overline{b}$ will be used to refer to individual pixels within the four sets.

Using this notation, similarity measures can now be expressed by ordinary set operations. The normalized binary correlation function is given by

$$C_{ijk} = \frac{\|S_{jk} \cap B_i\|}{\sqrt{\|S_{jk}\| \cdot \|B_i\|}} \tag{6.1}$$

and the absolute value of difference function by

$$V_{ijk} = \frac{1}{P}\left(\|S_{jk} \cap \overline{B}_i\| + \|\overline{S}_{jk} \cap B_i\|\right) \tag{6.2}$$

which has been normalized so that $0 \leq V_{ijk} \leq 1$. Unlike the case of gray level images, these measures are always equivalent for binary data since they are related by the equality

$$PV_{ijk} = \|S_{jk}\| + \|B_i\| - 2(\|S_{jk}\| \cdot \|B_i\|)^{\frac{1}{2}} C_{ijk} \tag{6.3}$$

The absolute value of difference, V_{ijk}, however, is simpler to compute and so is preferred over the correlation function.

The most likely position of the match occurs at the minimum value of V_{ijk}, V_i^*. The decision to accept or reject the best match is based on the result of a comparison

$$V_i^* < \tau \tag{6.4}$$

Equation (6.4) is in the form of a binary hypothesis test that selects between the hypotheses H_0, that the match is false, and H_1, that the match is correct. The decision threshold τ is chosen to achieve a given detection or false alarm rate. Although only V_i^* must satisfy (6.4), any offset at which $V_{ijk} < \tau$ should be considered as a potential match.

Formally, the detection rate is defined as the probability that V_{ijk} will be below the threshold given that H_1 is true.

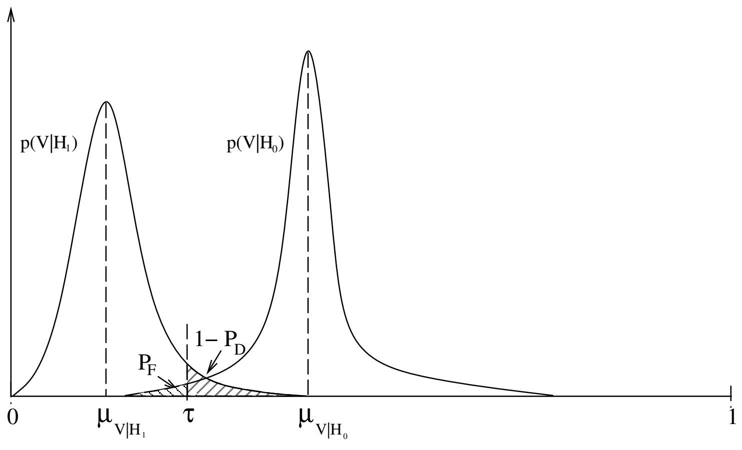


FIGURE 6-1: Threshold selection for a decision rule that chooses between two hypotheses H_0 and H_1.

$$P_D = \Pr(V_{ijk} < \tau | H_1) \tag{6.5}$$

while the false alarm rate is the probability that the test will be passed given that the match is incorrect [61].

$$P_F = \Pr(V_{ijk} < \tau | H_0) \tag{6.6}$$

As indicated by the diagram in Figure 6-1, it is not necessary to explicitly compute P_D and P_F to determine τ, but only to find the mean and variance of V_{ijk} under the hypotheses H_0 and H_1. The Chebyshev inequality [87] ensures that

$$P_D = 1 - \Pr(V_{ijk} > \tau | H_1) \geq 1 - \frac{\mathrm{Var}(V_{ijk}|H_1)}{(\tau - \mu_{H_1})^2} \tag{6.7}$$

and

$$P_F = \Pr(V_{ijk} < \tau | H_0) \leq \frac{\mathrm{Var}(V_{ijk}|H_0)}{(\tau - \mu_{H_0})^2} \tag{6.8}$$

To compute μ_{H_0} we assume that the distribution of edge pixels within the two blocks are independent since they correspond to different features. We further assume that the values of each pixel within the same block are independent and identically distributed (i.i.d.)

Bernoulli random variables. Although the values of neighboring edge pixels are certainly correlated, the assumption of independence is reasonable over the entire block if it is large enough. Let p_b denote the probability that any pixel in the block from the base image is a 1, and let p_s denote the same probability for any pixel in the block from the second image. Then

$$\mu_{H_0} \equiv \mathrm{E}[V_{ijk}|H_0] = p_s(1-p_b) + p_b(1-p_s) \tag{6.9}$$

The variance $\sigma^2_{H_0}$ is found by rewriting (6.2) as

$$V_{ijk} = \frac{1}{P}\left(\|S_{jk}\| + \|B_i\| - 2\|S_{jk}\| \cap \|B_i\|\right) \tag{6.10}$$

Then

$$\begin{aligned}
\sigma^2_{H_0} &\equiv \mathrm{Var}[V_{ijk}|H_0] = \\
&\frac{1}{P^2}\left[\mathrm{Var}\left(\|S_{jk}\|\right) + \mathrm{Var}\left(\|B_i\|\right) + 4\mathrm{Var}\left(\|S_{jk}\| \cap \|B_i\|\right) - \right. \\
&\qquad \left. 4\mathrm{Cov}\left(\|S_{jk}\|, \|S_{jk}\| \cap \|B_i\|\right) - 4\mathrm{Cov}\left(\|B_{jk}\|, \|S_{jk}\| \cap \|B_i\|\right)\right] \\
&= \frac{1}{P^2}\left[Pp_s(1-p_s) + Pp_b(1-p_b) + 4Pp_sp_b(1-p_sp_b) - \right. \\
&\qquad \left. 4Pp_bp_s(1-p_s) - 4Pp_sp_b(1-p_b)\right] \\
&= \frac{1}{P}\left[p_s(1-p_s) + p_b(1-p_b) - 4p_sp_b(1-p_s)(1-p_b)\right]
\end{aligned} \tag{6.11}$$

When H_1 is true, the distributions of edge pixels in the two blocks are no longer independent. Ideally, they should be identical, but due to the presence of noise, μ_{H_1} will not be zero. If we assume a Bernoulli noise process, n, with probability p_n such that

$$s = b\overline{n} + \overline{b}n \tag{6.12}$$

Then

$$b\overline{s} + \overline{b}s = n \tag{6.13}$$

and hence

$$\mu_{H_1} \equiv \mathrm{E}[V_{ijk}|H_1] \quad = \quad p_n \tag{6.14}$$

$$\sigma^2_{H_1} \equiv \mathrm{Var}(V_{ijk}|H_1) \quad = \quad \frac{p_n(1-p_n)}{P} \tag{6.15}$$

One of the difficulties with using equation (6.9) to determine τ is that it requires knowledge of both p_s and p_b. These can be estimated from a local analysis of the two edge maps, however, doing so greatly complicates the procedure. A simpler test can be formulated by counting only the edge pixels in each of the N blocks from the base edge map. Since these blocks do not change, this needs to be done only once.

Given $\|B_i\|$, the distribution of V_{ijk} under the null hypothesis changes slightly. We find that

$$\mu_{H_0,\|B\|} \equiv \mathrm{E}[V_{ijk}|H_0, \|B\|] = p_s\left(1 - \frac{\|B\|}{P}\right) + \frac{\|B\|}{P}(1-p_s) \tag{6.16}$$

and

$$\sigma^2_{H_0,\|B\|} \equiv \mathrm{Var}[V_{ijk}|H_0, \|B\|] = \frac{p_s(1-p_s)}{P} \tag{6.17}$$

If we consider only blocks for which $\|B_i\|/P \le 1/2$ then, since $p_s \ge 0$, it is always the case that

$$\mathrm{E}[V_{ijk}|H_0, \|B\|] \ge \frac{\|B\|}{P} \tag{6.18}$$

and hence a reasonable test can be formulated as

$$V_{ijk} \le \alpha\frac{\|B\|}{P}, \qquad (0 < \alpha < 1) \tag{6.19}$$

subject to

$$\beta p_n \le \frac{\|B\|}{P} \le \frac{1}{2}, \qquad (\beta > 1) \tag{6.20}$$

where α is chosen to ensure a low false alarm rate and β determines the detection rate given α and assuming a value for p_n. Equations (6.14)–(6.17), can be used in conjunction with equations (6.7) and (6.8) to set values for α and β. In the simulations of the matching procedure which have been performed, typical values are $\alpha = 0.5$ and $\beta p_n = 0.15$.

Although this procedure categorically rejects any block in the base map which does not first satisfy (6.20), the gain in simplicity, which directly impacts circuit complexity, is worth the loss of a few correspondence points. The loss due to the upper bound of 1/2 in (6.20) should not be too great since, on average, edges cover much less than half of the image. The lower bound would be necessary under any circumstances to avoid trying to match blocks containing very few features.

6.2 Other Tests

The previous analysis was based on the assumption that the pattern of edges in the block being matched was unique. When this assumption is true, the absolute value of difference function has a very sharp minimum at the location of the correct match. The variance of the function, as given by equation (6.17), will be very small if M is large enough since, as is easily shown,

$$\sigma_{H_0,||B||} \leq \frac{.5}{M} \tag{6.21}$$

If $M = 24$, for example, then $\sigma_{H_0,||B||} \leq .021$, It is not difficult to choose τ so that $(\mu_{H_0} - \tau)$ is many times larger than $\sigma_{H_0,||B||}$.

It is often the case, however, that the edge patterns are not unique. Repeating patterns occur frequently in natural scenes. The most common example is when the block contains linear segments that are part of larger entities, for instance the side of a door or of a table. Other examples occur with regular structures such as a set of drawers, or bookshelves.

The only practical way to deal with the problem of repeating patterns without greatly increasing the complexity of the procedure is to simply throw out any matches which do not have a single well-localized minimum. Several of the matching procedures discussed in Chapter 3 impose figural or continuity constraints to disambiguate multiple responses. However, these methods operate in software and can therefore consider global information. A procedure implemented by specialized VLSI circuits can use only local information.

Combining the test for a localized minimum with the restrictions (6.20) on the fraction of allowable base edge pixels and with the threshold test (6.19), only a relatively small percentage of the N blocks actually generate acceptable matches. It is therefore important to make N as large as is reasonably possible in order to ensure enough correspondence points are found to obtain a good estimate of the motion. It should be noted, however, that in spite of all these restrictions, there will still be errors that cannot be avoided. The purpose of the tests is to minimize the probability of these errors so that their effect on the motion estimates is minor. Additional steps can be taken, once a good estimate of the epipolar geometry has been obtained, to remove the remaining erroneous matches by identifying points which are significantly off the epipolar lines.

6.3 Simulations

The matching procedure has been tested on dozens of different image sequences. The astronaut and lab image sequences, shown previously in Figures 5-6 and 5-7, were chosen to illustrate several of the issues which have been raised in this chapter.

In applying the matching procedure to the astronaut sequence, the edge map from the left image was divided into 400 blocks whose centers were regularly spaced in a 20×20 array on the pixel grid. Each block measured 24×24 pixels, and the search in the right image was conducted over an area of 120×120 pixels surrounding the coordinates of the center of each block. The correspondences which were found are numbered sequentially and their locations are shown superimposed on the edge maps in Figure 6-2. The numbered locations of the correspondences are also displayed by themselves directly below the edge maps to aid the reader in finding them.

Of the 400 blocks, 135 passed all the tests and generated acceptable matches. The quality of the matches which did pass the tests can be seen to be quite good. They are all correct to within possible offset error caused by approximating the correspondence at the center of the area covered by the block in the second image. A distinguishing feature of the astronaut images is the absence of repeating patterns. In fact the edge maps have almost the appearance of random dot images, and as a result, few blocks were rejected for not producing a well localized minimum. The vast majority of those which were rejected failed either the threshold test (6.19) or did not have the edge density required to pass the test of (6.20).

The second sequence, composed of images taken in our laboratory, is a very different situation. Many of the objects in the scene, *i.e.,* the bookshelves, workstation monitors, and tripods, have long linear features for which it is impossible to find the correct match with any certainty using a windowing method. There are also regular repeating patterns, such as the supports on the bookshelves and the drawer handles, which generate multiple candidate matches when more than one instance is included in the search window. In addition, the motion, which includes a $\hat{z}$ axis rotation, complicates things even more by introducing a relative tilt in the edge patterns.

The matching procedure was executed on these images by dividing the left image into 900 blocks (in a 30×30 array), each measuring 24×24 pixels. The search was conducted over an area of 200 (horizontal) × 60 (vertical) pixels. Of the 900 blocks, 49 produced

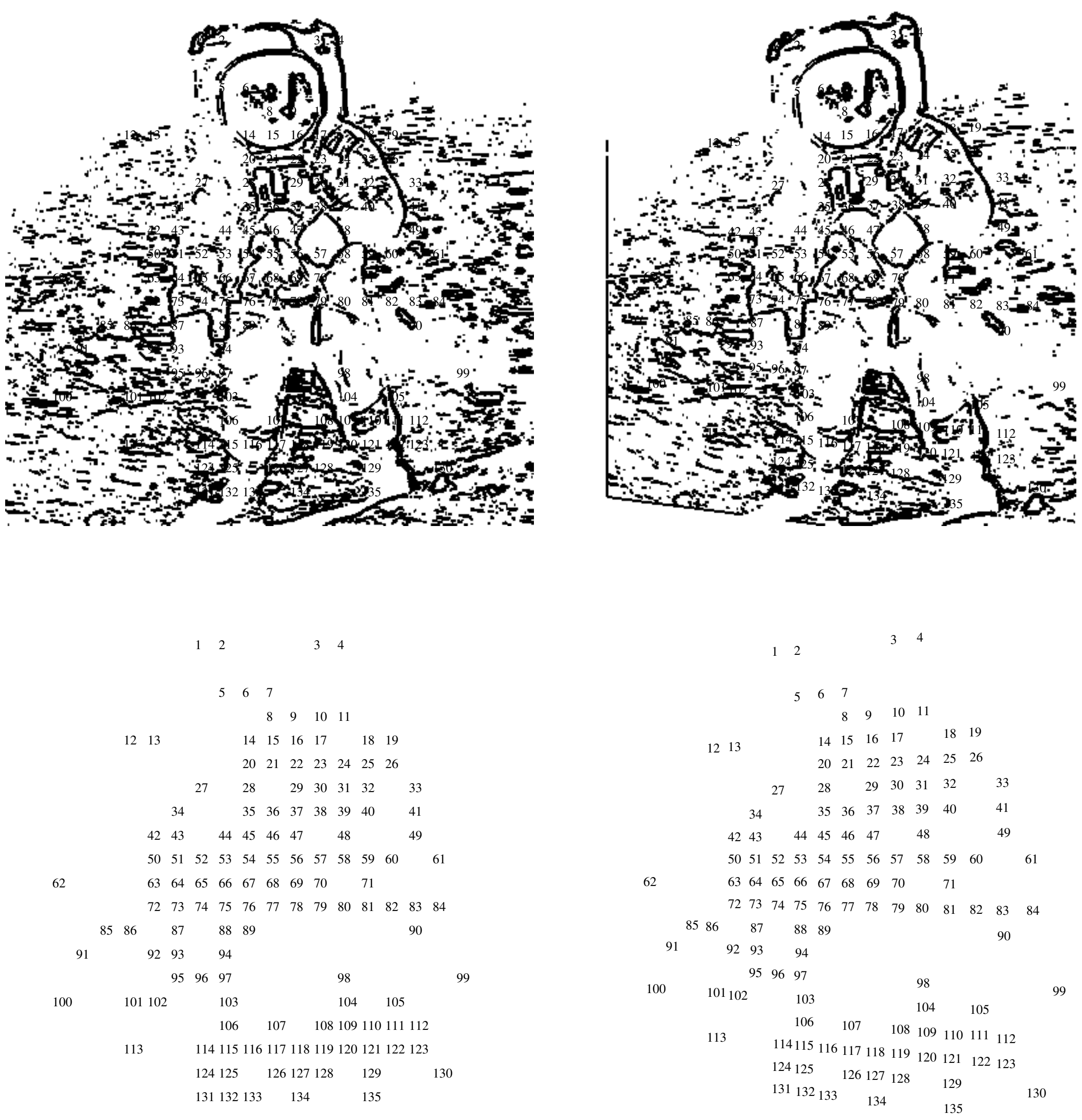


FIGURE 6-2: Binary edge maps of astronaut sequence with correspondence points found by the matching procedure.

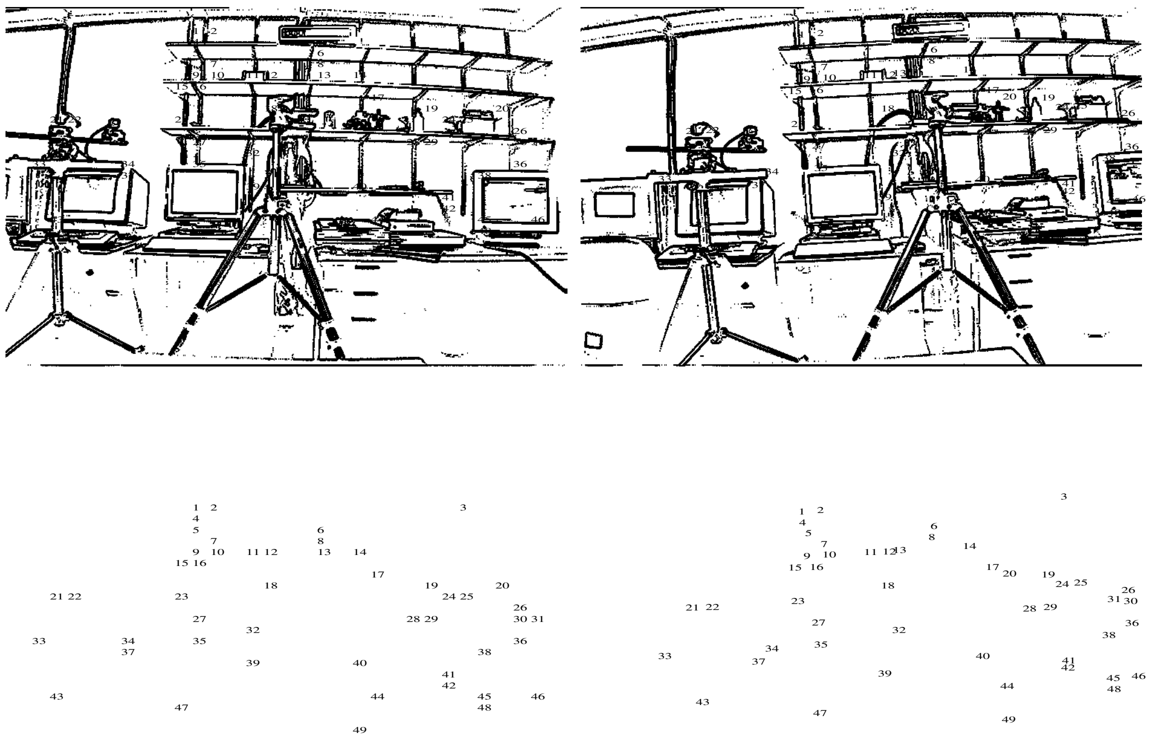


FIGURE 6-3: Binary edge maps of real motion sequence (lab scene) with correspondence points found by the matching procedure.

acceptable matches according to the different tests in the procedure. These are shown in Figure 6-3.

As in the astronaut sequence, most of these matches are very good. However, the proportion of blocks generating acceptable matches is much lower, and there are some obvious errors, such as points 13, 20, 32, and 46. The first three of these points are simply weak matches that passed the localization test only because they had a single minimum marginally below threshold, while the other minima were marginally above threshold. The last mismatched point, #46, demonstrates a different, but frequently encountered problem. This point lies near the border of the left image on the lower right corner of the workstation monitor which is not in the field of view in the second image. If the full monitor had been visible in the second image, the localization test would have rejected the match since there are several positions where a low score could be obtained. Instead, however, the wrong match was accepted.

Lowering the detection threshold is not a good solution for removing the marginal cases which slip past the localization test. This has the effect only of reducing the total number of matches, without changing the fact that the threshold can still fall in between the minima as in the cases above. In fact, there is not a simple solution at this level for removing the bad matches which escape detection without compromising the generality of the procedure. In the next chapter we will see how these false matches affect the computed motion estimates.

Chapter 7

Solving the Motion Equations

As previously discussed in Section 2.2, the basic procedure for computing general camera motion, or relative orientation, given a set of point correspondences $\{(\mathbf{r}_i, \boldsymbol{\ell}_i)\}$, $i = 1, \ldots, N$, is to find the rotation and baseline direction which minimize the sum of squared errors

$$S = \sum_{i=1}^{N} \lambda_i^2 \tag{7.1}$$

where λ_i is the triple product given in equation (2.34) as

$$\lambda_i = \mathbf{R}\boldsymbol{\ell}_i \cdot (\mathbf{r}_i \times \mathbf{b}) \tag{7.2}$$

Since the measurements of the locations of the correspondence points are not always equally reliable, it is often appropriate to define S as the weighted sum

$$S = \sum_{i=1}^{N} w_i \lambda_i^2 \tag{7.3}$$

where the weights $\{w_i\}$, $0 \leq w_i \leq 1$, reflect the relative confidences in the data.

The methods presented in Section 2.2 for minimizing (7.3) were developed to be executed on powerful digital computers where memory and power consumption limitations are not significant constraints. In this chapter a simplified algorithm which is much more suitable for implementation on low-level hardware, such as a programmable microcontroller, is presented. The algorithm, which is based on an adaptation of Horn's second method [4], is also an iterative nonlinear constrained minimization procedure. However, it breaks up the problem by alternating between updating the rotation and baseline, and in doing so, considerably reduces the size and complexity of the operations. The largest matrix which must be handled is 4×4, and the most complex operation at each iteration is solving a 3×3 eigenvalue-eigenvector problem.

It is not sufficient to present a method for computing camera motion without discussing problems of stability. There are several well known and analyzed cases in which the minimization problem is numerically unstable and which allow multiple solutions for the motion parameters. If we are going to build a robust system, we must be able to recognize and avoid these cases. In the next chapter, I will derive analytically the conditions for the function S to have more than one local minimum—even with (almost) perfect data—and will develop a test for determining when the solution found by the algorithm is indeed a reliable estimate of the true motion. In this chapter, I will merely introduce the subject of instability and multiple solutions and will present without proof the test for determining reliability. In the last section, I will present results which demonstrate both correct and incorrect convergence of the algorithm using the data from the astronaut and lab image sequences given in the preceeding chapters.

7.1 The Simplified Algorithm

In Horn's method, which was briefly discussed in Section 2.2, the rotation, represented by the unit quaternion $\mathring{q}$, and the baseline, represented indirectly by the quaternion $\mathring{d} = \mathring{b}\mathring{q}$, were updated simultaneously. This resulted in an 11×11 system of linear equations to be solved at each iteration in which three of the unknowns were the Lagrange multipliers from the constraint terms.

If, however, the motion is a pure rotation, or if either the baseline or the rotation is known, the problem becomes much easier. The simplified algorithm is based on the fact that by alternately solving these easier subproblems, assuming the values for $\mathring{q}$ and $\mathring{b}$ from the previous iteration, the estimates of the motion parameters will converge to those obtained from the more complex method in which $\mathring{q}$ and $\mathring{b}$ are updated simultaneously.

In this section, I will first present the procedures for solving the special cases and then combine these into a complete algorithm. Note that in the following derivations, the weights w_i are assumed to be constant.

7.1.1 Pure translation or known rotation

The triple product λ_i may be written as

$$\lambda_i = \mathbf{b} \cdot (\boldsymbol{\ell}'_i \times \mathbf{r}_i) \tag{7.4}$$

where $\boldsymbol{\ell}'_i$ denotes the ith left ray rotated into the right coordinate system. Let

$$\mathbf{c}_i \equiv \boldsymbol{\ell}'_i \times \mathbf{r}_i \tag{7.5}$$

The weighted sum of squares can then be expressed as

$$\begin{aligned} S &= \sum_{i=1}^{N} w_i \lambda_i^2 = \sum_{i=1}^{N} w_i (\mathbf{b} \cdot \mathbf{c}_i)^2 \\ &= \mathbf{b}^{\mathrm{T}} \left(\sum_{i=1}^{N} w_i \mathbf{c}_i \mathbf{c}_i^{\mathrm{T}} \right) \mathbf{b} \\ &= \mathbf{b}^{\mathrm{T}} \mathbf{C} \mathbf{b} \end{aligned} \tag{7.6}$$

If the rotation is given, or assumed, $\mathbf{C}$ may be treated as a constant matrix. It is a straightforward result from linear algebra that $\mathbf{C}$, being the sum of the dyadic products $\mathbf{c}_i \mathbf{c}_i^{\mathrm{T}}$, is symmetric and either positive definite, or at least, positive semi-definite. The unit vector $\mathbf{b}$ which minimizes S is the eigenvector of $\mathbf{C}$ corresponding to its smallest eigenvalue [3].

7.1.2 Rotation with known translation

There is not an equivalent closed form solution, such as the one above, for finding the rotation with $\mathbf{b}$ given, or assumed. It is possible, nonetheless, to solve the minimization problem by means of a simple iterative procedure starting from an initial guess, $\mathbf{R}_0$, for the rotation. At each iteration, k, we compute the incremental adjustment to the rotation, $\delta\mathbf{R}$, such that

$$\mathbf{R}_{k+1} = \delta\mathbf{R} \cdot \mathbf{R}_k \tag{7.7}$$

and

$$S(\mathbf{R}_{k+1}) \leq S(\mathbf{R}_k). \tag{7.8}$$

The procedure stops when the relative decrease in S is smaller than a given tolerance.

The incremental adjustment rotates each of the rays $\boldsymbol{\ell}'_i$ through an additional angle $\delta\theta$ about an axis $\widehat{\boldsymbol{\eta}}$. From Rodrigues' formula, equation (2.19), the new ray directions are given by

$$\delta\mathbf{R}\,\boldsymbol{\ell}'_i = \boldsymbol{\ell}'_i + \sin\delta\theta\,(\widehat{\boldsymbol{\eta}}\times\boldsymbol{\ell}'_i) + (1-\cos\delta\theta)\,\widehat{\boldsymbol{\eta}}\times(\widehat{\boldsymbol{\eta}}\times\boldsymbol{\ell}'_i) \tag{7.9}$$

which can be approximated to first order in $\delta\theta$ by

$$\delta\mathbf{R}\,\boldsymbol{\ell}'_i \approx \boldsymbol{\ell}'_i + \delta\theta\,(\widehat{\boldsymbol{\eta}}\times\boldsymbol{\ell}'_i) \tag{7.10}$$

The error at the end of iteration k is therefore

$$\begin{aligned}\lambda_{i,k+1} &= \mathbf{b}\cdot(\delta\mathbf{R}\,\boldsymbol{\ell}'_i\times\mathbf{r}_i)\\ &= \lambda_{i,k} + \delta\theta\,((\mathbf{b}\times\mathbf{r}_i)\times\boldsymbol{\ell}'_i)\cdot\widehat{\boldsymbol{\eta}}\end{aligned} \tag{7.11}$$

Let

$$\mathbf{a}_i \equiv (\mathbf{b}\times\mathbf{r}_i)\times\boldsymbol{\ell}'_i \quad\text{and}\quad \mathbf{m}\equiv -\delta\theta\,\widehat{\boldsymbol{\eta}} \tag{7.12}$$

so that

$$\lambda_{i,k+1} = \lambda_{i,k} - \mathbf{a}_i^{\mathrm{T}}\mathbf{m} \tag{7.13}$$

The total error is then

$$\begin{aligned}S_{k+1} &= \sum_{i=1}^{N} w_i\left(\lambda_{i,k}^2 - 2\lambda_{i,k}\,\mathbf{a}_i^{\mathrm{T}}\mathbf{m} + (\mathbf{a}_i^{\mathrm{T}}\mathbf{m})^2\right)\\ &= S_k - 2\mathbf{h}^{\mathrm{T}}\mathbf{m} + \mathbf{m}^{\mathrm{T}}\mathbf{A}\mathbf{m}\end{aligned} \tag{7.14}$$

where we have defined

$$\mathbf{h}\equiv\sum_{i=1}^{N} w_i\lambda_{i,k}\,\mathbf{a}_i \tag{7.15}$$

and

$$\mathbf{A}\equiv\sum_{i=1}^{N} w_i\,\mathbf{a}_i\mathbf{a}_i^{\mathrm{T}} \tag{7.16}$$

Except for pathological cases, *i.e.,* when the field of view is zero, or $N<3$, $\mathbf{A}$ will be invertible and positive definite. Accordingly, equation (7.14) posesses a unique minimum when

$$\mathbf{m} = \mathbf{A}^{-1}\mathbf{h} \tag{7.17}$$

and hence the $\delta\theta$ and $\widehat{\boldsymbol{\eta}}$ which minimize S to first order are given by

$$\delta\theta = \|\mathbf{m}\| \tag{7.18}$$

and

$$\widehat{\boldsymbol{\eta}} = -\widehat{\mathbf{m}} \tag{7.19}$$

In order to preserve orthonormality, $\delta\mathbf{R}$ should be computed exactly from $\delta\theta$ and $\widehat{\boldsymbol{\eta}}$ using Rodrigues' formula, equation (2.17), without approximation. Alternatively, one can maintain the rotation in unit quaternion form by computing

$$\delta\mathring{\mathrm{q}} = \left(\cos\frac{\delta\theta}{2}\,,\,\widehat{\boldsymbol{\eta}}\,\sin\frac{\delta\theta}{2}\right) \tag{7.20}$$

and

$$\mathring{\mathrm{q}}_{k+1} = \delta\mathring{\mathrm{q}}\,\mathring{\mathrm{q}}_k \tag{7.21}$$

The rules for transforming between unit quaternions and orthonormal matrices are given in Appendix A.

7.1.3 Pure rotation ($|\mathbf{b}| = 0$)

If the motion is a pure rotation, the procedure just described will still work given an arbitrary value for $\mathbf{b}$, but there is a simpler closed form method which can be applied. When $|\mathbf{b}| = 0$ we have, going back to the notation of equation (2.4) in Chapter 2,

$$\mathbf{p}_{ri} = \mathbf{R}\mathbf{p}_{li} \tag{7.22}$$

By the length preserving property of rotations,

$$|\mathbf{p}_{ri}| = |\mathbf{p}_{li}| \tag{7.23}$$

If spherical projection (2.15) is used so that

$$\mathbf{r}_i = \frac{\widetilde{\mathbf{p}}_{ri}}{|\widetilde{\mathbf{p}}_{ri}|}, \text{ and, } \boldsymbol{\ell}_i = \frac{\widetilde{\mathbf{p}}_{\ell i}}{|\widetilde{\mathbf{p}}_{\ell i}|} \tag{7.24}$$

it will also be true that

$$\mathbf{r}_i = \mathbf{R}\boldsymbol{\ell}_i = \boldsymbol{\ell}'_i \tag{7.25}$$

There are two possibilities for finding the rotation that best satisfies (7.25) in a least squares sense for the N correspondence points. The first is to define the error

$$\epsilon_i = \mathbf{r}_i \times \boldsymbol{\ell}'_i \tag{7.26}$$

and minimize

$$S = \sum_{i=1}^{N} w_i \epsilon_i^2 \tag{7.27}$$

This formulation leads directly to a procedure similar to that defined previously for the case of known translation.

The second method is to note that ideally

$$\mathbf{r}_i \cdot \boldsymbol{\ell}'_i = 1 \tag{7.28}$$

so that we can also solve for the rotation which *maximizes*

$$S' = \sum_{i=1}^{N} w_i (\mathbf{r}_i \cdot \boldsymbol{\ell}'_i) \tag{7.29}$$

This formulation was previously used by Horn in an algorithm to compute absolute orientation[1] [88]. Writing (7.29) with the rotation expressed by unit quaternions we have

$$\begin{aligned} S' &= \sum_{i=1}^{N} w_i (\mathring{\mathrm{r}}_i \cdot \mathring{\mathrm{q}} \mathring{\ell}_i \mathring{\mathrm{q}}^*) \\ &= \sum_{i=1}^{N} w_i (\mathring{\mathrm{r}}_i \mathring{\mathrm{q}} \cdot \mathring{\mathrm{q}} \mathring{\ell}_i) \end{aligned} \tag{7.30}$$

This expression can be cast into a more convenient form by introducing quaternion matrices.

[1]The difference between absolute and relative orientation is that in the former the distances to objects in the scene are known. Consequently one can vectorially subtract the translation once it has been computed to arrive at the pure rotation case. Note that this cannot be done for relative orientation since absolute distances are not known. Hence this method is only applicable if in fact the motion is a pure rotation.

As shown in Appendix A, if $\mathring{a}$ and $\mathring{b}$ are two quaternions then

$$\mathring{a}\mathring{b} = \mathcal{A}\mathring{b} = \underline{\mathcal{B}}\mathring{a} \tag{7.31}$$

$\mathcal{A}$ is referred to as the *left* quaternion matrix associated with $\mathring{a}$, and $\underline{\mathcal{B}}$ is referred to as the *right* quaternion matrix associated with $\mathring{b}$. Using (7.31) S' can be rewritten as

$$\begin{aligned} S' &= \sum_{i=1}^{N} w_i(\boldsymbol{\mathcal{R}}_i\mathring{q})^{\mathrm{T}}\underline{\boldsymbol{\mathcal{L}}}_i\mathring{q} \\ &= -\sum_{i=1}^{N} w_i\mathring{q}^{\mathrm{T}}\mathbf{M}_i\mathring{q} \end{aligned} \tag{7.32}$$

where $\mathbf{M}_i = \boldsymbol{\mathcal{R}}_i\underline{\boldsymbol{\mathcal{L}}}_i$. The minus sign arises from the fact that $\boldsymbol{\mathcal{R}}_i^{\mathrm{T}} = -\boldsymbol{\mathcal{R}}_i$. We can remove $\mathring{q}$ from the summation to obtain

$$\begin{aligned} S' &= -\mathring{q}^{\mathrm{T}}\left(\sum_{i=1}^{N} w_i\mathbf{M}_i\right)\mathring{q} \\ &= -\mathring{q}^{\mathrm{T}}\mathbf{M}\mathring{q} \end{aligned} \tag{7.33}$$

using $\mathbf{M}$ to represent the sum in parentheses.

S' is thus maximized by identifying $\mathring{q}$ with the eigenvector of $\mathbf{M}$ corresponding to its most negative eigenvalue. Since $\mathbf{M}$ is a 4×4 matrix, there is in principle a closed form solution for $\mathring{q}$, although it may be simpler to obtain the result by a standard iterative procedure.

7.1.4 The complete algorithm

Combining the procedures for the special cases we can formulate an algorithm to solve for the general case of unknown translation and rotation as follows:

> **Input:** $\mathring{q}^{(0)}$, $\mathbf{b}^{(0)}$, data
> if $\mathbf{b} = 0$
> $\mathring{q} = \textsc{Pure\_Rotate}(\text{data})$
> else {
> $k = 0$
> $S^{(0)} = \sum_{i=1}^{N} w_i\lambda_i^{2(0)}$

$change = 1$
while $(change > \epsilon)$ {
$\mathring{q}^{(k+1)} = \text{Update\_q}(\mathring{q}^{(k)}, \mathbf{b}^{(k)}, \text{data})$
$(\mathbf{b}^{(k+1)}, S^{(k+1)}) = \text{Update\_b}(\mathring{q}^{(k+1)}, \text{data})$
$change = (S^{(k)} - S^{(k+1)})/S^{(k)}$
$k++$
}
}

Pure_Rotate(), Update_b(), and Update_q() correspond to the procedures described in 7.1.1, 7.1.3, and 7.1.2, respectively. It is necessary to start the algorithm with initial values for $\mathbf{b}$ and $\mathring{q}$. These can be provided either externally or by obtaining $\mathring{q}^{(0)}$ from Pure_Rotate() and $\mathbf{b}^{(0)}$ from Update_b() with $\mathring{q}$ set to $(1,0,0,0)$. It is easily seen that the weighted sum of squares, $S^{(k)}$, monotonically decreases with each iteration since it decreases at each step. Since S is bounded below by zero, the algorithm will converge to some local minimum or stationary point.

7.2 Ambiguities and Multiple Solutions

There are four fundamental ambiguities associated with any pair $(\mathring{q}, \mathbf{b})$ which minimize the weighted sum of squares S. Since the equations involve only quadratic forms, S is unchanged by multiplying either $\mathring{q}$ or $\mathbf{b}$ by -1. The solution $-\mathring{q}$ is trivial since it corresponds to the same rotation as $\mathring{q}$. Changing the sign of $\mathbf{b}$, however, reverses the direction of the baseline which also affects the sign of the Z coordinates computed for objects in the scene.

A more subtle ambiguity occurs by imposing an additional rotation of π radians about the baseline which is equivalent to replacing $\mathring{q}$ by $\mathring{d} = \mathring{b}\mathring{q}$. We previously derived in equation (2.39) that

$$\lambda_i = \mathring{r}_i \mathring{b}\mathring{q} \cdot \mathring{q}\,\mathring{\ell}_i \tag{7.34}$$

and it is easily verified from the identities of Appendix A that replacing $\mathring{q}$ by $\mathring{b}\mathring{q}$ results in

$$\begin{aligned} \mathring{r}_i \mathring{b}\mathring{b}\mathring{q} \cdot \mathring{b}\mathring{q}\,\mathring{\ell}_i &= -\mathring{r}_i \mathring{q} \cdot \mathring{b}\mathring{q}\,\mathring{\ell}_i \\ &= -\mathring{r}_i \cdot \mathring{b}\,\mathring{\ell}'_i \end{aligned}$$

$$= \quad -\lambda_i \tag{7.35}$$

and hence will give the same total squared error.

For any solution, $(\mathring{q}, \mathbf{b})$, therefore, $(\mathring{q}, -\mathbf{b})$, $(\mathring{d}, \mathbf{b})$, $(\mathring{d}, -\mathbf{b})$, are also solutions. Each is derivable from the others, however, and only one should be feasible given the constraint that the imaged points are visible to both cameras. It is conventional therefore to count these four solutions as one [7].

If the data are error-free, a finite number of solutions exist to the *non* least-squares problem of solving $\lambda_i = 0$ for all N if there are at least five distinct ray pairs. Faugeras and Maybank [7] showed that in general there are 10 solutions to the five-point problem. If at least eight pairs are available, the solution is unique for most configurations of points. However, as first shown by Tsai and Huang [28] and Longuet-Higgins [8], there are configurations for which multiple solutions exist. Horn showed that only hyperboloids of one sheet and their degenerate forms viewed from a point on their surface could allow multiple interpretations [89], while Negahdaripour further demonstrated that only certain types of hyperboloids of one sheet and their degeneracies can result in an ambiguity, and in these cases, there are at most three possible solutions [40].

A more important concern for the present system is the fact that the function S may contain multiple local minima into which the algorithm can be trapped. The surfaces which give rise to multiple solutions of the equation $S = 0$ are rarely encountered in practice, and are even less likely to arise by chance due to errors in the matching process. However, as will be demonstrated analytically in the next chapter, many environments can, in a statistical sense, have a depth distribution which mimics the effects of those of the special surfaces, and can thus generate multiple local minima.

The conditions under which multiple solutions to the minimization problem most frequently arise are well known to be a function of the type of motion. Daniilidis and Nagel [6] derived analytically the conditions for instability in the case of pure translational motion or of translation with known rotation. They found that the extreme case occurs when the translation vector is parallel to the image plane and is accentuated as the field of view narrows. They as well as others (Spetsakis and Aloimonos [12], Horn [3], Weng *et al.* [90]) have proposed changing the error norm that is minimized to weight only the perpendicular distance of a point from its epipolar line in order to reduce the chance of convergence to an alternate minimum. Horn derived the symmetric weighting term in [3] as

$$w_i = \frac{|\mathbf{r}_i \times \boldsymbol{\ell}'_i|^2 \sigma_0^2}{((\mathbf{b} \times \mathbf{r}_i) \cdot (\mathbf{r}_i \times \boldsymbol{\ell}'_i))^2 |\mathbf{r}_i|^2 \sigma_{r_i}^2 + ((\mathbf{b} \times \boldsymbol{\ell}'_i) \cdot (\mathbf{r}_i \times \boldsymbol{\ell}'_i))^2 |\boldsymbol{\ell}'_i|^2 \sigma_{\ell_i}^2} \tag{7.36}$$

where σ_{r_i} and σ_{ℓ_i} represent the variances of the right and left rays in the ith measurement, and σ_0^2 is an arbitrary constant which is included to maintain consistency in the units.

One can gain a better understanding of equation (7.36) by going back to the rigid body motion equation which, even with imperfect data, must be approximately satisfied

$$\mathbf{p}_{ri} \approx \mathbf{R}\mathbf{p}_{li} + \mathbf{b} \tag{7.37}$$

Writing $\mathbf{r}_i = Z_{r_i}\mathbf{p}_{ri}$ and $\boldsymbol{\ell}_i = Z_{\ell_i}\mathbf{p}_{li}$, we can see that

$$\mathbf{b} \times \mathbf{r}_i \approx Z_{\ell_i}(\mathbf{r}_i \times \boldsymbol{\ell}'_i), \text{ and, } \mathbf{b} \times \boldsymbol{\ell}'_i \approx Z_{r_i}(\mathbf{r}_i \times \boldsymbol{\ell}'_i) \tag{7.38}$$

and hence equation (7.36) can be written as

$$\begin{aligned} w_i &= \frac{\sigma_0^2}{|\mathbf{r}_i \times \boldsymbol{\ell}'_i|^2 \left(Z_{\ell_i}^2 |\mathbf{r}_i|^2 \sigma_{r_i}^2 + Z_{r_i}^2 |\boldsymbol{\ell}'_i|^2 \sigma_{\ell_i}^2 \right)} \\ &= \frac{\sigma_0^2}{|\mathbf{r}_i|^2 |\boldsymbol{\ell}_i|^2 \sin^2 \alpha_i \left(Z_{\ell_i}^2 |\mathbf{r}_i|^2 \sigma_{r_i}^2 + Z_{r_i}^2 |\boldsymbol{\ell}_i|^2 \sigma_{\ell_i}^2 \right)} \end{aligned} \tag{7.39}$$

where α_i is the angle between $\mathbf{r}_i$ and $\boldsymbol{\ell}'_i$.

For rays corresponding to points approaching infinity, $Z_{r_i} \approx Z_{\ell_i}$ and $\alpha_i \rightarrow 0$ as $1/Z$, resulting in

$$w_i \longrightarrow \frac{\sigma_0^2}{|\mathbf{r}_i|^2 |\boldsymbol{\ell}_i|^2 \left(|\mathbf{r}_i|^2 \sigma_{r_i}^2 + |\boldsymbol{\ell}_i|^2 \sigma_{\ell_i}^2 \right)} \tag{7.40}$$

However, for points near the cameras, assuming their relative angle of rotation is $< 90^0$, α_i becomes larger as Z_{r_i} and Z_{ℓ_i} go to zero, so that in the limit, $w_i \rightarrow \infty$.

Equation (7.36) thus correctly weights rays corresponding to points closer to the cameras more strongly than those corresponding to far away points. Unfortunately, it is necessary to know either the epipolar geometry or the Z coordinates of the matched points in advance in order to use this equation. If the w_i are computed from the current estimate of the baseline and rotation, then it is not possible to prove that the algorithm will converge to an unbiased estimate of the correct solution.

In an automated system for computing motion, it is more important to be able to identify when the algorithm has converged to the wrong stationary point than it is to try to ensure that it never does. The numerical instability of the algorithm in the case of translation parallel to the image plane is inherent and cannot be removed without prior knowledge of the motion. Nonetheless, we can often obtain useful estimates of the motion, even under these conditions, when it can be determined that the algorithm has converged to the local minimum closest to the true solution. In the next chapter, I will show that the most reliable indicator of correct convergence is the ratio μ_2/μ_3, where μ_2 and μ_3 are the middle and largest eigenvalues of the matrix $\mathbf{C}$ defined in equation (7.6). This ratio theoretically depends on the orientation of $\mathbf{b}$ with respect to the vector $\hat{v}_3 = \mathbf{R}\hat{z}$. For translation parallel to the image plane, and therefore approximately perpendicular to $\hat{v}_3$, μ_2/μ_3 is an increasing function of the field of view and will be $\ll 1$ for most practical imaging systems. When the translation is parallel to $\hat{v}_3$, however, $\mu_2/\mu_3 \approx 1$. We can thus determine if the algorithm has converged to the correct estimate by comparing the actual ratio to the one predicted from the values of $\mathbf{b}$ and $\mathbf{R}$ retuned by the algorithm. If the actual ratio is small compared to its predicted value, we can reject the solution as unreliable and proceed to the next set of images.

Once it has been determined that the computed motion is reliable, a variety of techniques may be used to improve the estimates. For example, we can execute the algorithm several times to remove outliers, *i.e.,* correspondence pairs for which $|\lambda_i|$ is much greater than the average, or to apply weighting factors such as given by equation (7.36) using the previous estimates of $\mathbf{b}$ and $\mathring{q}$.

7.3 Simulations

The results of applying the simplified algorithm to the astronaut and lab sequences previously seen in Figures 5-6 and 5-7 are given in Tables 7.1–7.3. In the astronaut sequence, which was generated by software, the motion with respect to the origin of the camera coordinate system is known exactly, while for the lab sequence, it is only known approximately. Both sequences, however, correspond approximately to the classically unstable case of translation perpendicular to $\mathbf{R}\hat{z}$.

The correspondence points used to compute the motion for the astronaut images are

	$\mathbf{b}$	θ	$\widehat{\omega}$	$\overline{S}$	μ_2/μ_3 *actual*	μ_2/μ_3 *pred.*
True motion	$(1.0, 0.0, 0.0)$	5°	$(0.0, 1.0, 0.0)$			
First Test. Initial values from estimates of pure translation and rotation						
Initial values:	$(.9982, -.0129, -.0591)$	9.72°	$(-.0045, .9998, -.0200)$	7.7e-4		
First pass	$(.9998, -.0164, -.0120)$	4.95°	$(-.0132, .9999, -.0011)$	6.0e-6	.418	.475
Second pass	$(.9999, -.0133, -.0083)$	4.99°	$(-.0106, .9999, 0.00)$	3.4e-6	.441	.475
Second Test. Initial values chosen to result in alternate local minimum						
Initial values:	$(0.0, 0.0, 1.0)$	5°	$(0.0, 1.0, 0.0)$	4.9e-3		
First pass	$(.0481, .0093, .9988)$	10.66°	$(-.0016, 1.0000, -.0036)$	7.1e-6	.586	.979
Second pass	$(.0403, -.0040, .9992)$	10.63°	$(-.0012, 1.0000, -.0004)$	3.1e-7	.471	.977

Table 7.1: Simulation results on the astronaut sequence.

those shown in Figure 6-2 which were generated by the matching procedure. As was previously noted, the quality of these matches is very good, and this is reflected in the closeness of the calculated and the true motion. Table 7.1 shows the results of two tests. In the first, initial values were chosen by computing the best pure translation and best pure rotation which fit the data. In order to improve the estimates, two passes of the algorithm were performed, with the first to obtain an initial estimate and identify outliers to be removed. As can be seen, there is very little difference in the solutions computed in the two passes for the astronaut sequence. Of the 135 correspondence points, only 19 were found to have an error greater than one standard deviation above the mean. The reliability of the results is also evidenced by the closeness of the actual and predicted values for the ratio μ_2/μ_3, given that the field of view for this simulated sequence was set at $\sim 55^\circ$.

TDespite the fact that the field of view is relatively large and there are many correspondence points, it is nonetheless possible to make the algorithm converge to another local minimum. Starting the algorithm with an initial value of $\mathbf{b} = \hat{z}$, shown as the second test in Table 7.1, the result is quite far from the known solution. Removing outliers from the data does not prevent the algorithm from converging to this incorrect result, and nor does any other weighting scheme. It is interesting to note that the value of $\overline{S} \equiv S/N$ is of no use in discriminating between the correct and incorrect solutions since it is very small in both cases. The actual and predicted values of μ_2/μ_3, however, do show the difference. In the second test they are .586 and .979 in the first pass and .471 and .977 in the second. The fact that the predicted ratio is significantly different from the actual value indicates that these results are unreliable.

	$\mathbf{b}$	θ	$\widehat{\omega}$	$\overline{S}$	μ_2/μ_3 *actual*	μ_2/μ_3 *pred.*
Motion of stage	$(1.0, 0.0, 0.0)$	5°	$(0.0, 0.0, -1.0)$			
First Test. Initial values from estimates of pure translation and rotation						
Initial values:	$(.5867, -.0878, .8050)$	7.43°	$(.0278, .8102, -.5855)$	7.06e-5		
First pass	$(.9937, .0007, .1120)$	5.04°	$(.1286, -.3815, -.9154)$	4.09e-5	.027	.084
Second pass, (rejects: 20, 39, 43)	$(.0247, -.1316, .9910)$	7.86°	$(.0387, .8559, -.5156)$	1.84e-5	.045	.776
Second Test. Initial values chosen to converge to estimate of correct motion						
Initial values:	$(.9990, .0316, .0316)$	7.43°	$(.0278, .8102, -.5855)$	1.21e-4		
First pass	$(.9961, -.0543, .0703)$	5.12°	$(.0207, -.3750, -.9268)$	3.41e-5	.027	.083
Second pass, (rejects: 39, 43)	$(.9972, -.0513, .0552)$	5.88°	$(.0137, -.5309, -.8474)$	3.91e-6	.026	.083
Third Test. Same initial values as 2nd test, but points 13, 20, 32, 46 removed by hand.						
Initial values:	$(.9990, .0316, .0316)$	7.49°	$(.0274, .8184, -.5740)$	1.02e-4		
First pass	$(.9940, -.0085, .1095)$	4.84°	$(.1126, -.2912, -.9500)$	3.67e-5	.028	.084
Second pass, (rejects: 39, 43)	$(.9961, -.0056, .0885)$	5.74°	$(.1026, -.5122, -.8527)$	4.09e-6	.026	.083

Table 7.2: Simulation results on the lab sequence with points from automatic matching procedure.

	$\mathbf{b}$	θ	$\widehat{\omega}$	$\overline{S}$	μ_2/μ_3 *actual*	μ_2/μ_3 *pred.*
Motion of stage	$(1.0, 0.0, 0.0)$	5°	$(0.0, 0.0, -1.0)$			
First Test. Initial values from estimates of pure translation and rotation						
Initial values:	$(.9915, -.0656, -.1126)$	7.69°	$(-.0019, .8193, -.5733)$	1.29e-4		
First pass	$(.9964, -.0808, .0262)$	6.07°	$(-.0484, -.5737, -.8176)$	2.64e-6	.067	.087
Second pass, (rejects: 16, 17, 20)	$(.9964, -.0802, .0287)$	6.0°	$(-.0453, -.5591, -.8278)$	1.55e-6	.073	.083

Table 7.3: Simulation results on the lab sequence with hand-picked correspondence points.

In the lab sequence, the exact motion with respect to the camera coordinate system is unknown because the system was not accurately calibrated. In order to evaluate the quality of the correspondences obtained by the matching procedure against a known standard, we compare the results for the automatic data with those from a second set of correspondence points chosen by hand. To estimate the motion for both sets of data, we use the following approximate internal calibration matrix, derived from the manufacturer's data on the camera and lens used with these images,

$$\mathbf{K}_c = \begin{pmatrix} 567 & 0.0 & 378 \\ 0.0 & 484 & 242 \\ 0.0 & 0.0 & 1.0 \end{pmatrix} \tag{7.41}$$

to transform between image plane and world coordinates, as described in Section 2.1.1.

The correspondences obtained by the matching procedure are shown in Figure 6-3. It was previously observed that four of the points, specifically 13, 20, 32, and 46, were clearly wrong, while it was less obvious if other points were also in error. Table 7.2 lists the results of three tests conducted on the data from the matching procedure. In the first, the estimates of pure translation and pure rotation were used as starting values for the algorithm. Although the initial value for **b**: $(.5867, -.0878, .8050)$ is quite far from the approximate translation direction, the algorithm does converge to a reasonably close solution on the first pass. On the second pass, however, it falls into an alternate stationary point—indicating that removing outliers on a heuristic basis does not always give a better estimate. The actual and predicted values of μ_2/μ_3 once again point out the difference in the two results. In the first we have .027 and .084 for the actual and predicted ratios, computed for an effective field of view of 30^o, while in the second we have .045 and .776, indicating an unreliable result.

In the second test, a starting value of $\mathbf{b} = (.9990, .0316, .0316)$ was used, and this time the algorithm converged to a reliable solution on both passes. Interestingly, none of the four clearly incorrect matches was rejected after the first pass, although points 39 and 43, which are not so obviously wrong, were rejected. The actual and predicted ratios of μ_2/μ_3 of $(.027, .083)$ on the first pass, and $(.026, .083)$ on the second, are very close and indicate that the solutions are reliable. Although the errors for points 13, 20, 32, and 46 are quite noticeable, careful examination reveals that they do not have a large component in the direction perpendicular to the correct epipolar line, while points 39 and 43, which were rejected, do. In the third test, we verify directly that these four points have little effect on the computed motion by manually removing them from the data set. As seen in Table 7.2, the results of this test are almost identical to those of the previous one.

The manually chosen correspondence points for this sequence are shown in Figure 7-1. There are 28 points in all which were selected using a high resolution display and a

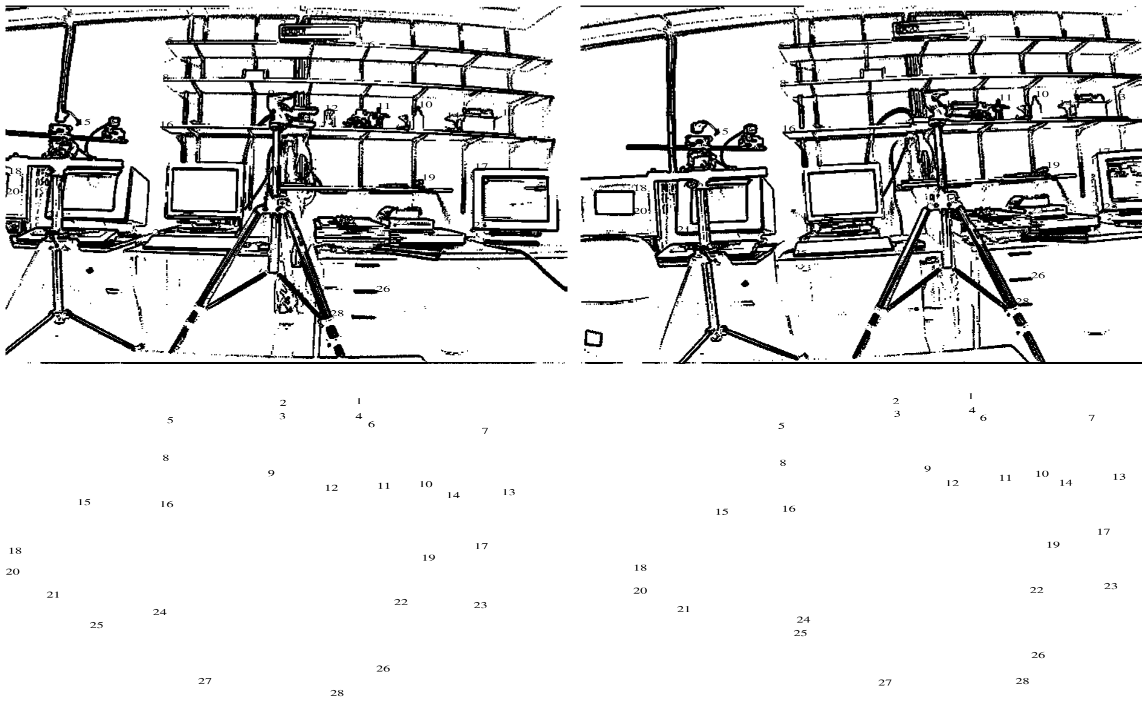


FIGURE 7-1: Binary edge maps of real motion sequence with *hand picked* point matches.

mouse-driven pointer. The same test, in which the motion was computed using the initial values derived from the estimates of pure translation and rotation, that was performed on the points found by the matching procedure was performed on these data with the results given in Table 7.3. This time the initial estimate of the translation was much closer to the actual value and the algorithm converged to the correct estimate on both passes. As can be seen by comparing Tables 7.3 and 7.2, the estimates computed for the manually and the automatically chosen points are very close: $\mathbf{b} = (.9964, -.0802, .0287)$, $\theta = 6.0^{o}, \hat{\omega} = (-.0453, -.5591, -.8278)$ for the manual data vs. $\mathbf{b} = (.9972, -.0513, .0552)$, $\theta = 5.88^{o}, \hat{\omega} = (.0137, -.5309, -.8474)$ and $\mathbf{b} = (.9961, -.0056, .0885)$, $\theta = 5.74^{o}, \hat{\omega} = (.1026, -.5122, -.8527)$ in the second and third tests with the automatic data.

Based on the results from both the astronaut and lab sequences, we can thus conclude that the data obtained by the matching procedure are of comparable quality, at least with respect to estimating motion, to those obtained by more elaborate methods.

Chapter 8

The Effects of Measurement Errors

Understanding the effects of errors in the data on the estimated motion is critical in designing an automated system for tasks that demand high reliability, such as navigating an autonomous vehicle. Previous studies on the effects of error, however, have been incomplete and would lead one to believe that any system built from current technologies would at best give poor results.

In this chapter, I will analyze in detail the numerical stability of the motion algorithm, which affects its sensitivity to error, and derive analytic expressions for the expected estimation error in the case of both random and systematic errors in the data. I will also derive the conditions under which the algorithm will converge to an alternate local minimum and develop the theoretical basis of the ratio test to determine if the solution reported by the algorithm is reliable. In the last section, I will compare the theoretical predictions of the first part of the chapter with the results from simulations on data from artificially generated motion sequences with varying amounts of added error.

Several important results are obtained in this analysis. The first, of course, is the development of the ratio test to determine reliability. Just as important, however, is the fact that the error analysis provides us with guidelines for designing a system with the required sensor resolution and field of view, as well as with the required number and size of matching circuits, to obtain a given maximum expected error in the estimated motion. Finally, I also derive an interesting practical result which is that precise internal camera calibration is not necessary in order to obtain accurate estimates of the translation direction, as long as the rotation is estimated as well.

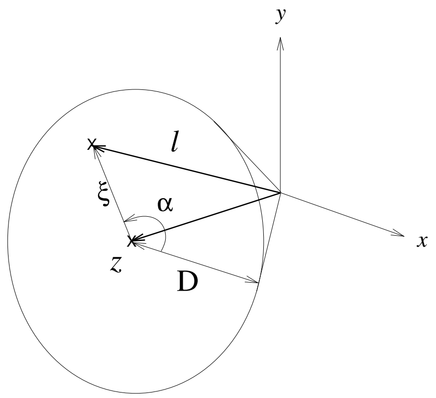


FIGURE 8-1: Geometry of the position vector for a point on the image plane and cone of all vectors for a given field of view.

8.1 Numerical Stability

In order to understand the problems of numerical instability, we must have analytical forms for the matrices **C** and **A** which are used in the procedures to update **b** and $\mathring{q}$. Assuming the correspondence points are distributed uniformly over the left and right images, we can approximate the summation of equation (7.3) by an integration over the field of view.

$$S = \sum_{i=1}^{N} w_i \lambda_i^2 \longrightarrow \frac{N}{\pi D^2} \int_0^D \int_0^{2\pi} w(\xi, \alpha)\, \lambda^2\, \xi\, d\xi\, d\alpha \tag{8.1}$$

We assume that the correspondence points in the left image are contained within a circle of radius D centered about the point $(0, 0)$, as shown in Figure 8-1 and use the homogeneous form (2.14) of representing ray directions. The vector $\boldsymbol{\ell}$ from the center of projection to a point $(\xi \cos\alpha,\, \xi \sin\alpha)$ on the image plane is thus

$$\boldsymbol{\ell} = \begin{pmatrix} \xi \cos\alpha \\ \xi \sin\alpha \\ 1 \end{pmatrix} \tag{8.2}$$

where $\xi \in [\,0, D]$ and $\alpha \in [\,0, 2\pi)$. Since we have implicitly set $f = 1$, the viewing angle ϕ is computed as

$$\phi = \tan^{-1} D \tag{8.3}$$

The factor $N/\pi D^2$ in equation (8.1) takes into account the number of correspondence points by representing them as a uniform density over the viewing field. It is not appropriate to make the assumption that N scales proportionally to the viewing area, that is as D^2, because the number of pixels on the sensor, which ultimately limits N, is constant. If we change the optics on the imaging system to give a wider field of view, the number of correspondences will not increase significantly, the points will simply be distributed over a larger area.

In this section I will derive the analytical forms for $\mathbf{C}$ and $\mathbf{A}$ in the case where the data are error-free, as well as the conditions for S to have multiple local minima corresponding to feasible solutions. In the next section I will analyze the case of imperfect data and study the effects of error on the reliability of the motion estimates.

8.1.1 Eigenvalues and eigenvectors of C

In section 7.1.1, we defined the matrix $\mathbf{C}$ as

$$\mathbf{C} = \sum_{i=1}^{N} w_i \, \mathbf{c}_i \mathbf{c}_i^{\mathrm{T}} \tag{8.4}$$

$$= \sum_{i=1}^{N} w_i \, (\boldsymbol{\ell}'_i \times \mathbf{r}_i)(\boldsymbol{\ell}'_i \times \mathbf{r}_i)^{\mathrm{T}} \tag{8.5}$$

where $\boldsymbol{\ell}'_i = \mathbf{R}\boldsymbol{\ell}_i$.

In this section, we assume that the data are error-free so that the rigid body motion equation

$$\mathbf{p}_r = \mathbf{R}\mathbf{p}_l + \mathbf{b} \tag{8.6}$$

previously seen as equation (2.4) in Chapter 2, holds exactly.

We define $\mathbf{r}_i$ and $\boldsymbol{\ell}_i$ as the homogeneous vectors

$$\mathbf{r}_i = \tilde{\mathbf{p}}_{r\,i} = \frac{1}{Z_{r_i}} \mathbf{p}_{r\,i} \text{ and, } \boldsymbol{\ell}_i = \tilde{\mathbf{p}}_{\ell i} = \frac{1}{Z_{\ell_i}} \mathbf{p}_{l\,i} \tag{8.7}$$

so that

$$Z_{r_i} \mathbf{r}_i = Z_{\ell_i} \boldsymbol{\ell}'_i + \mathbf{b} \tag{8.8}$$

Taking the cross product of both sides with $\boldsymbol{\ell}'_i$ we thus have

$$\boldsymbol{\ell}'_i \times \mathbf{r}_i = \frac{1}{Z_{r_i}} (\boldsymbol{\ell}'_i \times \mathbf{b}) \tag{8.9}$$

Since there is no error in the data, we set $w_i = 1$ and write $\mathbf{C}$ as

$$\begin{aligned}
\mathbf{C} &= \sum_{i=1}^{N} (\boldsymbol{\ell}'_i \times \mathbf{r}_i)(\boldsymbol{\ell}'_i \times \mathbf{r}_i)^{\mathrm{T}} && (8.10) \\
&= \sum_{i=1}^{N} \frac{1}{Z_{r_i}^2} (\boldsymbol{\ell}'_i \times \mathbf{b})(\boldsymbol{\ell}'_i \times \mathbf{b})^{\mathrm{T}} && (8.11) \\
&= -\mathbf{B}_\times \left(\sum_{i=1}^{N} \frac{1}{Z_{r_i}^2} \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \right) \mathbf{B}_\times && (8.12)
\end{aligned}$$

where

$$\mathbf{B}_\times = \begin{pmatrix} 0 & -b_z & b_y \\ b_z & 0 & -b_x \\ -b_y & b_x & 0 \end{pmatrix} \tag{8.13}$$

We now make the approximation that the correspondences are distributed uniformly over the image and replace the summation of equation (8.12) by the integral

$$\mathbf{C} = -\mathbf{B}_\times \left(\frac{N}{\pi D^2} \int_0^D \int_0^{2\pi} \frac{1}{Z_r^2} \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha \right) \mathbf{B}_\times \tag{8.14}$$

The distribution of depths, Z_r, of points in the scene is of course unknown. However, given that we are interested only in analyzing the general structure of the matrix $\mathbf{C}$ for different types of motion, we can consider Z_r as a random variable whose probability distribution is independent of ξ and α and can therefore replace the $1/Z_r^2$ term by its expected value and take it outside the integral. We then have

$$\mathbf{C} = -\kappa \mathbf{B}_\times \left(\frac{N}{\pi D^2} \int_0^D \int_0^{2\pi} \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha \right) \mathbf{B}_\times \tag{8.15}$$

where κ has been defined as

$$\kappa \equiv \left(\overline{\frac{1}{Z_r^2}} \right) \tag{8.16}$$

This integral is now in the form of equation (B.3) whose solution is given in Appendix B, equation (B.20). The result is

$$\mathbf{C} = -\kappa N \mathbf{B}_\times \left(\frac{D^2}{4} \left(\mathbf{I} - \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} \right) + \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} \right) \mathbf{B}_\times \tag{8.17}$$

$$\begin{aligned} &= \kappa N \left(\frac{D^2}{4} \left(\mathbf{I} - \mathbf{b}\mathbf{b}^{\mathrm{T}} - (\mathbf{b} \times \hat{v}_3)(\mathbf{b} \times \hat{v}_3)^{\mathrm{T}} \right) \right. \\ &+ \left. (\mathbf{b} \times \hat{v}_3)(\mathbf{b} \times \hat{v}_3)^{\mathrm{T}} \right) \end{aligned} \tag{8.18}$$

where $\hat{v}_3 = \mathbf{R}\hat{z}$ represents the rotation of the optical axis in the left camera system.

By inspection, we can see that both $\mathbf{b}$ and $\mathbf{b} \times \hat{v}_3$ are eigenvectors of (8.18) with eigenvalues $\mu_{\mathbf{b}} = 0$ and

$$\mu_{\mathbf{b}\times\hat{v}_3} = \kappa N \left(\frac{D^2}{4} \left(1 - |\mathbf{b} \times \hat{v}_3|^2 \right) + |\mathbf{b} \times \hat{v}_3|^2 \right) \tag{8.19}$$

$$= \kappa N \left(\frac{D^2}{4} (\mathbf{b} \cdot \hat{v}_3)^2 + |\mathbf{b} \times \hat{v}_3|^2 \right) \tag{8.20}$$

Consequently $(\mathbf{b} \times \hat{v}_3) \times \mathbf{b}$ is also an eigenvector with eigenvalue

$$\mu_{(\mathbf{b}\times\hat{v}_3)\times\mathbf{b}} = \frac{\kappa N D^2}{4} \tag{8.21}$$

Only the eigenvalue of $\mathbf{b} \times \hat{v}_3$ depends on the motion. Its extreme values are obtained when $\mathbf{b} \perp \hat{v}_3$ and $\mathbf{b} \parallel \hat{v}_3$. If $\mathbf{b} \perp \hat{v}_3$ we have

$$\mu_{\mathbf{b}\times\hat{v}_3} = \kappa N \tag{8.22}$$

For $D < 2$, or a viewing angle $\phi < 63.4^{\mathrm{o}}$, this will be the largest eigenvalue, and the ratio of the second largest to the largest eigenvalues will be

$$\frac{\mu_2}{\mu_3} = \frac{D^2}{4} \tag{8.23}$$

The numerical stability of determining the translation with known rotation is related to this ratio. If it is small compared to zero—which is the ratio of the smallest and largest eigenvalues—adding error to the data can cause the two smallest eigenvalues to switch places. In most practical situations $\hat{v}_3$ will be very close to $\hat{z}$. From equation (2.16), $\hat{v}_3$, which is the third column of the rotation matrix $\mathbf{R}$, is given by

$$\hat{v}_3 = \begin{pmatrix} \omega_x\omega_z(1-\cos\theta)+\omega_y\sin\theta \\ \omega_y\omega_z(1-\cos\theta)-\omega_x\sin\theta \\ \cos\theta+\omega_z^2(1-\cos\theta) \end{pmatrix} \tag{8.24}$$

If $\theta = 0$ or $\hat{\boldsymbol{\omega}} = \hat{z}$, $\hat{v}_3$ is identically equal to $\hat{z}$. If $\hat{\boldsymbol{\omega}} \neq \hat{z}$ and θ is so large that the approximation $\cos\theta \sim 1$ is not valid, the two cameras will not image the same scene. The unstable case thus usually occurs when the motion is (nearly) parallel to the image plane and the field of view is small, as reported by Daniilidis and Nagel [6]. The eigenvector corresponding to the second largest eigenvalue is $(\mathbf{b}\times\hat{v}_3)\times\mathbf{b} = \hat{v}_3$. If the standard deviation of the error in the data is greater than the difference between the two smallest eigenvalues, we may find that the procedure UPDATE_B() reports a translation direction which is close to $\hat{z}$ instead of to $\hat{x}$. This is the instablility which is most commonly observed in practice.

The other extreme case corresponds to $\mathbf{b} = \hat{v}_3$, or motion (nearly) parallel to the optical axis. In this case, $\hat{v}_3$ is an eigenvector and so is any vector perpendicular to $\hat{v}_3$, since the direction of $\mathbf{b}\times\hat{v}_3$ is undefined. The two largest eigenvalues are equal

$$\mu_2 = \mu_3 = \frac{\kappa N D^2}{4} \tag{8.25}$$

and thus the estimation of the translation direction is numerically stable, independently of the field of view.

8.1.2 Condition of the matrix A

The condition number of the matrix $\mathbf{A}$, K_A, defined as the ratio of its largest and smallest eigenvalues, is a measure of its nearness to singularity. Since the incremental update to the rotation requires inverting $\mathbf{A}$, K_A is the critical parameter in determining the numerical stability of this procedure.

In equation (7.16) we defined $\mathbf{A}$ as

$$\mathbf{A} \equiv \sum_{i=1}^{N} w_i\,\mathbf{a}_i\mathbf{a}_i^{\mathrm{T}} \tag{8.26}$$

where

$$\mathbf{a}_i = (\mathbf{b} \times \mathbf{r}_i) \times \boldsymbol{\ell}'_i \tag{8.27}$$

As before, we assume that the data are error-free so that $w_i = 1$ and we can write

$$Z_{r_i}\mathbf{r}_i = Z_{\ell_i}\boldsymbol{\ell}'_i + \mathbf{b} \tag{8.28}$$

Taking the cross product of both sides with $\mathbf{b}$ we have

$$\mathbf{b} \times \mathbf{r}_i = \frac{Z_{\ell_i}}{Z_{r_i}}(\mathbf{b} \times \boldsymbol{\ell}'_i) \tag{8.29}$$

and hence

$$\begin{aligned} \mathbf{a}_i &= \frac{Z_{\ell_i}}{Z_{r_i}}(\mathbf{b} \times \boldsymbol{\ell}'_i) \times \boldsymbol{\ell}'_i && (8.30) \\ &= \frac{Z_{\ell_i}}{Z_{r_i}}\left((\mathbf{b} \cdot \boldsymbol{\ell}'_i)\boldsymbol{\ell}'_i - |\boldsymbol{\ell}'_i|^2\mathbf{b}\right) && (8.31) \end{aligned}$$

Substituting the above expression for $\mathbf{a}_i$ into equation (8.26), we obtain

$$\mathbf{A} = \sum_{i=1}^{N}\left(\frac{Z_{\ell_i}}{Z_{r_i}}\right)^2 \left[(\mathbf{b} \cdot \boldsymbol{\ell}'_i)^2\boldsymbol{\ell}'_i\boldsymbol{\ell}'^{\mathrm{T}}_i - |\boldsymbol{\ell}'_i|^2(\mathbf{b} \cdot \boldsymbol{\ell}'_i)\left(\mathbf{b}\boldsymbol{\ell}'^{\mathrm{T}}_i + \boldsymbol{\ell}'_i\mathbf{b}^{\mathrm{T}}\right) + |\boldsymbol{\ell}'_i|^4\mathbf{b}\mathbf{b}^{\mathrm{T}}\right] \tag{8.32}$$

If the distances to objects in the scene are large compared to the baseline length, the terms $(Z_{\ell_i}/Z_{r_i})^2$ should be close to 1. In any case, we may assume they are random variables which are independent of position and which may therefore be replaced by their expected value. Let

$$\gamma \equiv E\left[\left(\frac{Z_{\ell_i}}{Z_{r_i}}\right)^2\right] \tag{8.33}$$

Then, making the approximation that the correspondences are distributed uniformly over the image, equation (8.32) becomes

$$\mathbf{A} = \frac{\gamma N}{\pi D^2}\int_0^D\int_0^{2\pi}\left[(\mathbf{b} \cdot \boldsymbol{\ell}')^2\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}} - |\boldsymbol{\ell}'|^2(\mathbf{b} \cdot \boldsymbol{\ell}')\left(\mathbf{b}\boldsymbol{\ell}'^{\mathrm{T}} + \boldsymbol{\ell}'\mathbf{b}^{\mathrm{T}}\right) + |\boldsymbol{\ell}'|^4\mathbf{b}\mathbf{b}^{\mathrm{T}}\right]\xi\, d\xi\, d\alpha \tag{8.34}$$

Each term on the right-hand side of this expression corresponds to one of the special integrals computed in Appendix B. The solutions are as follows:

$$\int_0^D \int_0^{2\pi} (\mathbf{b}\cdot\boldsymbol{\ell}')^2 \boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\,\xi\,d\xi\,d\alpha = \frac{\pi D^4}{4}\left[\left(\frac{D^2}{6}-1\right)\left(2\mathbf{w}\mathbf{w}^{\mathrm{T}} + |\mathbf{w}|^2\left(\mathbf{I} - \hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)\right) + 2\mathbf{b}\mathbf{b}^{\mathrm{T}} + \mathbf{I}\right] + \pi D^2(\mathbf{b}\cdot\hat{v}_3)^2\,\hat{v}_3\hat{v}_3{}^{\mathrm{T}}\left(1-\frac{3D^2}{4}\right) \tag{8.35}$$

where

$$\mathbf{w} \equiv (\hat{v}_3\times\mathbf{b})\times\hat{v}_3 = \left(\mathbf{I} - \hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)\,\mathbf{b} \tag{8.36}$$

For the second term:

$$\int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^2(\mathbf{b}\cdot\boldsymbol{\ell}')\left(\mathbf{b}\boldsymbol{\ell}'^{\mathrm{T}} + \boldsymbol{\ell}'\mathbf{b}^{\mathrm{T}}\right)\,\xi\,d\xi\,d\alpha = \int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^2\left(\mathbf{b}\mathbf{b}^{\mathrm{T}}\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}} + \boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\mathbf{b}\mathbf{b}^{\mathrm{T}}\right)\,\xi\,d\xi\,d\alpha$$

$$= \pi D^2\left[\left(\frac{D^2}{2}+\frac{D^4}{3}\right)\mathbf{b}\mathbf{b}^{\mathrm{T}} + \left(1+\frac{D^2}{4}-\frac{D^4}{6}\right)(\mathbf{b}\cdot\hat{v}_3)\left(\mathbf{b}\hat{v}_3{}^{\mathrm{T}} + \hat{v}_3\mathbf{b}^{\mathrm{T}}\right)\right] \tag{8.37}$$

and for the last term:

$$\int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^4\mathbf{b}\mathbf{b}^{\mathrm{T}}\,\xi\,d\xi\,d\alpha = \pi D^2\left(1 + D^2 + \frac{D^4}{3}\right)\mathbf{b}\mathbf{b}^{\mathrm{T}} \tag{8.38}$$

Combining (8.35), (8.37), and (8.38) and skipping much of the messy algebra, we obtain

$$\begin{aligned}\mathbf{A} \;=\; \gamma N\Bigg[&\left(1-\frac{D^2}{4}-\frac{D^4}{12}\right)\mathbf{w}\mathbf{w}^{\mathrm{T}} + \frac{D^2}{4}\left(\frac{D^2}{6} + \left(1-\frac{D^2}{6}\right)(\mathbf{b}\cdot\hat{v}_3)^2\right)\mathbf{I} \\ &+ \frac{D^2}{4}\left(1-\frac{D^2}{6}\right)\left(1-5(\mathbf{b}\cdot\hat{v}_3)^2\right)\hat{v}_3\hat{v}_3{}^{\mathrm{T}} + \left(\frac{3D^2}{4}+\frac{D^4}{6}\right)\mathbf{b}\mathbf{b}^{\mathrm{T}}\Bigg]\end{aligned} \tag{8.39}$$

Since

$$\mathbf{w}\cdot(\mathbf{b}\times\hat{v}_3) = \hat{v}_3\cdot(\mathbf{b}\times\hat{v}_3) = \mathbf{b}\cdot(\mathbf{b}\times\hat{v}_3) = 0 \tag{8.40}$$

$(\mathbf{b}\times\hat{v}_3)$ is clearly an eigenvector of $\mathbf{A}$ with eigenvalue

$$\mu_{\mathbf{b}\times\hat{v}_3} = \frac{\gamma N D^2}{4}\left(\frac{D^2}{6} + \left(1-\frac{D^2}{6}\right)(\mathbf{b}\cdot\hat{v}_3)^2\right) \tag{8.41}$$

The other two eigenvectors are therefore linear combinations of $\mathbf{b}$ and $\hat{v}_3$ but do not coincide exactly with either $\mathbf{b}$ or $\hat{v}_3$ unless $\mathbf{b} \perp \hat{v}_3$ or $\mathbf{b} \parallel \hat{v}_3$.

If $\mathbf{b} \perp \hat{v}_3$, then $\mathbf{w} = \mathbf{b}$ and $\mathbf{A}$ becomes

$$\mathbf{A} = \gamma N \left[\left(1 + \frac{D^2}{2} + \frac{D^4}{12}\right) \mathbf{b}\mathbf{b}^{\mathrm{T}} + \frac{D^4}{24}\mathbf{I} + \frac{D^2}{4}\left(1 - \frac{D^2}{6}\right) \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} \right] \tag{8.42}$$

The eigenvectors and eigenvalues are then, in increasing order,

$$\mathbf{b} \times \hat{v}_3, \text{ with } \mu_{\mathbf{b}\times\hat{v}_3} = \frac{\gamma N D^4}{24} \tag{8.43}$$

$$\hat{v}_3, \text{ with } \mu_{\hat{v}_3} = \frac{\gamma N D^2}{4} \tag{8.44}$$

and

$$\mathbf{b}, \text{ with } \mu_{\mathbf{b}} = \gamma N \left(1 + \frac{D^2}{2} + \frac{D^4}{8}\right) \tag{8.45}$$

Note that this order holds for $D < \sqrt{6}$, or for a viewing angle $\phi < 67.8^{\circ}$, at which point $\mu_{\mathbf{b}\times\hat{v}_3} = \mu_{\hat{v}_3}$. The condition number is given by

$$K_{A_\perp} = \frac{\mu_{\mathbf{b}}}{\mu_{\mathbf{b}\times\hat{v}_3}} = \frac{24}{D^4}\left(1 + \frac{D^2}{2} + \frac{D^4}{8}\right) \tag{8.46}$$

As $D \to 0$, $K_{A_\perp} \to \infty$, as we should expect. This reflects the well-known phenomenon, which occurs with small fields of view, of interference between the displacement patterns caused by a rotational motion and those caused by a translation parallel to the image plane. As $D \to \infty$, $K_{A_\perp}$ decreases and eventually goes to zero, however for values of D which exist in real imaging systems, it remains quite large. For example, if $\phi < 60^{\circ}$ ($D^2 < 3$), $K_{A_\perp} > 8.67$. Hence the estimate of the rotation will always be very sensitive to error when $\mathbf{b} \cdot \hat{v}_3 = 0$.

If $\mathbf{b} = \hat{v}_3$, $\mathbf{w} = \mathbf{0}$, and $\mathbf{A}$ becomes

$$\mathbf{A} = \frac{\gamma N D^2}{4}\left[\mathbf{I} - \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} + \frac{4D^2}{3}\hat{v}_3 \hat{v}_3{}^{\mathrm{T}}\right] \tag{8.47}$$

$\hat{v}_3$ is an eigenvector and so is any vector perpendicular to $\hat{v}_3$. The eigenvalues are

$$\mu_{\hat{v}_3} = \frac{\gamma N D^4}{3} \tag{8.48}$$

and

$$\mu_{\perp} = \frac{\gamma N D^2}{4} \tag{8.49}$$

When $D^2 = 3/4$, or $\phi = 36.9^o$, the eigenvalues are equal. For $D < \sqrt{3/4}$

$$K_{A_{\|}} = \frac{\mu_{\perp}}{\mu_{\hat{v}_3}} = \frac{3}{4D^2} \tag{8.50}$$

while for$D > \sqrt{3/4}$

$$K_{A_{\|}} = \frac{\mu_{\hat{v}_3}}{\mu_{\perp}} = \frac{4D^2}{3} \tag{8.51}$$

$K_{A_{\|}}$ thus has a minimum at $D = \sqrt{3/4}$ and goes to infinity both as $D \to 0$ and as $D \to \infty$. For $60^o \geq \phi \geq 23.4^o$, however, $K_{A_{\|}} \leq 4$, and so for viewing fields used in most real imaging systems, the estimation of the rotation will be robust.

8.1.3 Minimizers of S

Using the results derived in this section, we can now determine the conditions for S to have more than one stationary point corresponding to a feasible, non-trivial solution. As long as $D \neq 0$ and the motion is not a pure rotation, in which case $\mathbf{C} = \mathbf{0}$, the smallest eigenvalue of $\mathbf{C}$ given the true rotation is unique, and hence so is the solution for the baseline. Let $\mathbf{b}_0$ and $\mathbf{R}_0$ denote the baseline and rotation corresponding to the actual motion. If an alternate minimum of S exists corresponding to the solutions $\mathbf{b}'$ and $\mathbf{R}'$, it must be the case that

$$\mathbf{R}' \neq \mathbf{R}_0 \tag{8.52}$$

$$\mathbf{C}'\mathbf{b}' = \mu'\mathbf{b}' \tag{8.53}$$

and

$$\mathbf{h}' = \mathbf{0} \tag{8.54}$$

where μ' is the smallest eigenvalue of $\mathbf{C}'$ and

$$\mathbf{C}' = \sum_{i=1}^{N} (\boldsymbol{\ell}''_i \times \mathbf{r}_i)(\boldsymbol{\ell}''_i \times \mathbf{r}_i)^{\mathrm{T}} \tag{8.55}$$

with $\boldsymbol{\ell}''_i = \mathbf{R}'\boldsymbol{\ell}_i$.

We can always write $\mathbf{R}'$ as $\mathbf{R}' = \delta\mathbf{R} \cdot \mathbf{R}_0$ so that $\boldsymbol{\ell}''_i = \delta\mathbf{R}\,\boldsymbol{\ell}'_i$. If $\delta\mathbf{R}$ corresponds to an incremental rotation of $\delta\theta$ about an axis $\widehat{\boldsymbol{\eta}}$, then we can use the small angle approximation to Rodrigues' formula to obtain

$$\begin{aligned} \boldsymbol{\ell}''_i &\approx \boldsymbol{\ell}'_i + \delta\theta\,(\widehat{\boldsymbol{\eta}} \times \boldsymbol{\ell}'_i) \\ &= \boldsymbol{\ell}'_i + \boldsymbol{\ell}'_i \times \mathbf{m} \end{aligned} \tag{8.56}$$

where $\mathbf{m} = -\delta\theta\,\widehat{\boldsymbol{\eta}}$. We can thus write

$$\lambda'_i = \lambda'_{i0} + \delta\lambda'_i, \text{ and } \mathbf{a}'_i = \mathbf{a}'_{i0} + \delta\mathbf{a}'_i \tag{8.57}$$

where

$$\lambda'_{i0} \equiv \mathbf{b}' \cdot (\boldsymbol{\ell}'_i \times \mathbf{r}_i) \tag{8.58}$$

$$\mathbf{a}'_{i0} \equiv (\mathbf{b}' \times \mathbf{r}_i) \times \boldsymbol{\ell}'_i \tag{8.59}$$

$$\begin{aligned} \delta\lambda'_i &\equiv \mathbf{b}' \cdot ((\boldsymbol{\ell}'_i \times \mathbf{m}) \times \mathbf{r}_i) \\ \delta\mathbf{a}'_i &\equiv (\mathbf{b}' \times \mathbf{r}_i) \times (\boldsymbol{\ell}'_i \times \mathbf{m}) \end{aligned} \tag{8.60}$$

Noting that

$$\delta\lambda'_i = -((\mathbf{b}' \times \mathbf{r}_i) \times \boldsymbol{\ell}'_i) \cdot \mathbf{m} = -{\mathbf{a}'_{i0}}^{\mathrm{T}}\mathbf{m} \tag{8.61}$$

the vector $\mathbf{h}'$ thus becomes

$$\begin{aligned} \mathbf{h}' &= \sum_{i=1}^{N} \lambda'_i \mathbf{a}'_i \\ &= \sum_{i=1}^{N} \left(\lambda'_{i0} - {\mathbf{a}'_{i0}}^{\mathrm{T}}\mathbf{m}\right)(\mathbf{a}'_{i0} + \delta\mathbf{a}'_i) \\ &= \sum_{i=1}^{N} \lambda'_{i0}\mathbf{a}'_{i0} - \sum_{i=1}^{N} \mathbf{a}'_{i0}{\mathbf{a}'_{i0}}^{\mathrm{T}}\mathbf{m} + \sum_{i=1}^{N} \left(\lambda'_{i0} - {\mathbf{a}'_{i0}}^{\mathrm{T}}\mathbf{m}\right)\delta\mathbf{a}'_i \end{aligned} \tag{8.62}$$

Let $S^* = \mu'$ denote the value of S given $\mathbf{b}'$ and $\mathbf{R}'$, and let $\mathbf{C}_0$ denote the matrix $\mathbf{C}$ computed using $\mathbf{R}_0$. From equation (7.14) we then have

$$\begin{aligned} S^* &= \sum_{i=1}^{N} \left({\lambda'_{i0}}^2 - 2\lambda'_{i0}{\mathbf{a}'_{i0}}^{\mathrm{T}}\mathbf{m} + \mathbf{m}^{\mathrm{T}}\mathbf{a}'_{i0}{\mathbf{a}'_{i0}}^{\mathrm{T}}\mathbf{m} \right) \\ &= \mathbf{b}'^{\mathrm{T}}\mathbf{C}_0\mathbf{b}' - 2{\mathbf{h}'_0}^{\mathrm{T}}\mathbf{m} + \mathbf{m}^{\mathrm{T}}\mathbf{A}'_0\mathbf{m} \end{aligned} \tag{8.63}$$

where

$$\mathbf{h}'_0 \equiv \sum_{i=1}^{N} \lambda'_{i0}\mathbf{a}'_{i0}, \text{ and, } \mathbf{A}'_0 \equiv \sum_{i=1}^{N} \mathbf{a}'_{i0}{\mathbf{a}'_{i0}}^{\mathrm{T}} \tag{8.64}$$

If S^* is indeed a local minimum, then from equation (7.17), it must also be the case that

$${\mathbf{h}'_0}^{\mathrm{T}} = \mathbf{A}'_0\mathbf{m} \tag{8.65}$$

giving

$$S^* = \mathbf{b}'^{\mathrm{T}}\mathbf{C}_0\mathbf{b}' - \mathbf{m}^{\mathrm{T}}\mathbf{A}'_0\mathbf{m} \tag{8.66}$$

as well as

$$\mathbf{h}' = \sum_{i=1}^{N} \left(\lambda'_{i0} - {\mathbf{a}'_{i0}}^{\mathrm{T}}\mathbf{m} \right) \delta\mathbf{a}'_i \tag{8.67}$$

Defining

$$\delta\mathbf{h}' \equiv \sum_{i=1}^{N} \lambda'_{i0}\delta\mathbf{a}'_i, \text{ and, } \delta\mathbf{A}' \equiv \sum_{i=1}^{N} \delta\mathbf{a}'_i{\mathbf{a}'_{i0}}^{\mathrm{T}} \tag{8.68}$$

we see that the necessary condition, $\mathbf{h}' = \mathbf{0}$, for S to have a stationary point can thus be stated as

$$\mathbf{m} = (\delta\mathbf{A}')^{-1}\delta\mathbf{h}' \tag{8.69}$$

We can now show at least one case, which occurs when $\mathbf{b}_0 \perp \hat{v}_3$, where we know an alternate solution exists. Since

$$\mathbf{r}_i = \frac{Z_{\ell_i}}{Z_{r_i}}\boldsymbol{\ell}'_i + \frac{1}{Z_{r_i}}\mathbf{b}_0 \tag{8.70}$$

we can write $\mathbf{a}'_{i0}$ as

$$\mathbf{a}'_{i0} = \frac{Z_{\ell_i}}{Z_{r_i}}(\mathbf{b}' \times \boldsymbol{\ell}'_i) \times \boldsymbol{\ell}'_i - \frac{1}{Z_{r_i}}\mathbf{u} \times \boldsymbol{\ell}'_i \tag{8.71}$$

where $\mathbf{u} \equiv \mathbf{b}_0 \times \mathbf{b}'$. Now let $\mathbf{b}' = \hat{v}_3$ and $\mathbf{m} = \alpha\,\mathbf{u}$, where α is some constant, so that

$$S^* = \hat{v}_3{}^{\mathrm{T}}\mathbf{C}_0\hat{v}_3 - \alpha^2\mathbf{u}^{\mathrm{T}}\left(\sum_{i=1}^{N}\left(\frac{Z_{\ell_i}}{Z_{r_i}}\right)^2((\hat{v}_3 \times \boldsymbol{\ell}'_i) \times \boldsymbol{\ell}'_i)((\hat{v}_3 \times \boldsymbol{\ell}'_i) \times \boldsymbol{\ell}'_i)^{\mathrm{T}}\right)\mathbf{u} \tag{8.72}$$

The summation in the second term of this equation is identical in form to the one in equation (8.32) whose analytic expression, derived by approximating the sum as an integral, is given on the right-hand side of equation (8.39). Upon substituting $\hat{v}_3$ for $\mathbf{b}$ and using the facts that $\mathbf{w}' = (\hat{v}_3 \times \mathbf{b}') \times \hat{v}_3 = \mathbf{0}$ and $\hat{v}_3 \cdot \mathbf{b}_0 = 0$, we obtain

$$\mathbf{u}^{\mathrm{T}}\left(\sum_{i=1}^{N}\left(\frac{Z_{\ell_i}}{Z_{r_i}}\right)^2((\mathbf{b}' \times \boldsymbol{\ell}'_i) \times \boldsymbol{\ell}'_i)((\mathbf{b}' \times \boldsymbol{\ell}'_i) \times \boldsymbol{\ell}'_i)^{\mathrm{T}}\right)\mathbf{u} \longrightarrow \frac{\gamma N D^2}{4} \tag{8.73}$$

Since $\hat{v}_3$ is an eigenvector of $\mathbf{C}_0$, when $\hat{v}_3 \perp \mathbf{b}_0$, with eigenvalue

$$\mu_{\hat{v}_3} = \frac{\kappa N D^2}{4} \tag{8.74}$$

we thus have

$$S^* = \frac{ND^2}{4}\left(\kappa - \alpha^2\gamma\right) \tag{8.75}$$

If $\alpha^2 = \kappa/\gamma$, S^* will be identically zero, and thus will clearly be a minimum. However, we should keep in mind that the constants κ and γ are only approximations to values that would be obtained if the Z coordinates of the points in the scene were known exactly, and hence we cannot use equation (8.75) to find α by setting $S^* = 0$. To show that the solution $\mathbf{b}' = \hat{v}_3$, $\mathbf{m} = \alpha\,(\mathbf{b}_0 \times \hat{v}_3)$ minimizes S, we need to show that it satisfies the necessary conditions (8.53) and (8.54).

We first check that $\mathbf{h}' = \mathbf{0}$ by expanding each of the terms in equation (8.67), applying the rigid body motion equation (8.70) and the conditions $\mathbf{b}_0 \cdot \hat{v}_3 = 0$ and $\mathbf{m} = \alpha\,\mathbf{b}_0 \times \hat{v}_3$, to give

$$\lambda'_{i0} = \hat{v}_3 \cdot (\boldsymbol{\ell}'_i \times \mathbf{r}_i) = \frac{1}{Z_{r_i}}\boldsymbol{\ell}'_i \cdot (\mathbf{b}_0 \times \hat{v}_3) \tag{8.76}$$

$$\begin{aligned}
\mathbf{a}'_{i0}{}^{\mathrm{T}}\mathbf{m} &= \alpha\frac{Z_{\ell_i}}{Z_{r_i}}((\hat{v}_3 \times \boldsymbol{\ell}'_i) \times \boldsymbol{\ell}'_i) \cdot (\mathbf{b}_0 \times \hat{v}_3) \\
&= \alpha\frac{Z_{\ell_i}}{Z_{r_i}}(\hat{v}_3 \cdot \boldsymbol{\ell}'_i)\,\boldsymbol{\ell}'_i \cdot (\mathbf{b}_0 \times \hat{v}_3)
\end{aligned} \tag{8.77}$$

and

$$\begin{aligned}
\delta \mathbf{a}'_i &= \alpha\,(\hat{v}_3 \times \mathbf{r}_i) \times (\boldsymbol{\ell}'_i \times (\mathbf{b}_0 \times \hat{v}_3)) \\
&= \alpha \frac{Z_{\ell_i}}{Z_{r_i}} (\hat{v}_3 \times \boldsymbol{\ell}'_i) \times (\boldsymbol{\ell}'_i \times (\mathbf{b}_0 \times \hat{v}_3)) + \alpha \frac{1}{Z_{r_i}} (\hat{v}_3 \times \mathbf{b}_0) \times (\boldsymbol{\ell}'_i \times (\mathbf{b}_0 \times \hat{v}_3)) \\
&= -\alpha \left[\frac{Z_{\ell_i}}{Z_{r_i}} (\boldsymbol{\ell}'_i \cdot \mathbf{b}_0)\boldsymbol{\ell}'_i + \frac{1}{Z_{r_i}} (\boldsymbol{\ell}'_i \cdot (\mathbf{b}_0 \times \hat{v}_3))(\mathbf{b}_0 \times \hat{v}_3) \right]
\end{aligned} \tag{8.78}$$

We can thus write

$$\begin{aligned}
\mathbf{h}' &= -\alpha \sum_{i=1}^{N} \frac{Z_{\ell_i}}{Z_{r_i}^2} \left(1 - \alpha Z_{\ell_i} (\hat{v}_3 \cdot \boldsymbol{\ell}'_i)\right) \left((\mathbf{b}_0 \times \hat{v}_3) \cdot \boldsymbol{\ell}'_i\right) \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \mathbf{b}_0 \\
&\quad - \alpha \sum_{i=1}^{N} \frac{1}{Z_{r_i}^2} \left(1 - \alpha Z_{\ell_i} (\hat{v}_3 \cdot \boldsymbol{\ell}'_i)\right) (\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i (\mathbf{b}_0 \times \hat{v}_3)(\mathbf{b}_0 \times \hat{v}_3)
\end{aligned} \tag{8.79}$$

and then replace the sums by integrals, assuming the rays are uniformly distributed, to obtain an analytic expression. Before writing the solution, however, we first use a result shown at the ends of Sections B.5 and B.6, that substituting $(\hat{v}_3 \cdot \boldsymbol{\ell}'_i) = 1$, in the above equation does not affect the value of either integral. We also note from equation (B.28) that the integral

$$\int_0^D \int_0^{2\pi} \left((\mathbf{b}_0 \times \hat{v}_3) \cdot \boldsymbol{\ell}'\right) \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} \mathbf{b}_0 \, \xi \, d\xi \, d\alpha \longrightarrow \mathbf{0} \tag{8.80}$$

We thus have

$$\begin{aligned}
\mathbf{h}' &\longrightarrow -\alpha\kappa \left(1 - \alpha \overline{Z_\ell}\right) \frac{N}{\pi D^2} \int_0^D \int_0^{2\pi} (\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} (\mathbf{b}_0 \times \hat{v}_3)(\mathbf{b}_0 \times \hat{v}_3)\, \xi \, d\xi \, d\alpha \\
&= -\alpha\kappa \frac{N D^2}{4} \left(1 - \alpha \overline{Z_\ell}\right) (\mathbf{b}_0 \times \hat{v}_3)
\end{aligned} \tag{8.81}$$

where $\overline{Z_\ell} \equiv E[Z_\ell]$.

The condition $\mathbf{h} = \mathbf{0}$ can be thus be satisfied by setting $\alpha = 1/\overline{Z_\ell}$. Again, this is a convenient approximation, however, it does not change the fact that we can find a constant α which satisfies $\mathbf{h} = \mathbf{0}$ by setting

$$\alpha = \frac{\sum_{i=1}^N 1/Z_{r_i}^2 (\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i (\mathbf{b}_0 \times \hat{v}_3)}{\sum_{i=1}^N Z_{\ell_i}/Z_{r_i}^2 (\hat{v}_3 \cdot \boldsymbol{\ell}'_i)(\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i (\mathbf{b}_0 \times \hat{v}_3)} \tag{8.82}$$

The second necessary condition for a minimum is that $\mathbf{b}' = \hat{v}_3$ must be the eigenvector of $\mathbf{C}'$ corresponding to its smallest eigenvalue. We write $\mathbf{C}'$ as

$$\begin{aligned}
\mathbf{C}' &= \sum_{i=1}^{N} \mathbf{c}'_i \mathbf{c}'^{\mathrm{T}}_i \\
&= \sum_{i=1}^{N} (\mathbf{c}_{i0} + \delta\mathbf{c}_i)(\mathbf{c}_{i0} + \delta\mathbf{c}_i)^{\mathrm{T}} \\
&= \mathbf{C}_0 + \sum_{i=1}^{N} \left(\mathbf{c}_{i0} \delta\mathbf{c}_i{}^{\mathrm{T}} + \delta\mathbf{c}_i \mathbf{c}_{i0}^{\mathrm{T}} \right) + \sum_{i=1}^{N} \delta\mathbf{c}_i \delta\mathbf{c}_i{}^{\mathrm{T}}
\end{aligned} \tag{8.83}$$

where

$$\begin{aligned}
\mathbf{c}_{i0} &\equiv \boldsymbol{\ell}'_i \times \mathbf{r}_i \\
\delta\mathbf{c}_i &\equiv \alpha(\boldsymbol{\ell}'_i \times (\mathbf{b}_0 \times \hat{v}_3)) \times \mathbf{r}_i
\end{aligned} \tag{8.84}$$

Using the rigid body motion equation (8.70), we expand $\mathbf{c}_{i0}$ as

$$\mathbf{c}_{i0} = -\frac{1}{Z_{r_i}} \mathbf{B}_{\times 0}\, \boldsymbol{\ell}'_i \tag{8.85}$$

where $\mathbf{B}_{\times 0}$ is the cross-product matrix corresponding to the operation $\mathbf{b}_0 \times$, and in a similar manner write $\delta\mathbf{c}_i$ as

$$\delta\mathbf{c}_i = \frac{\alpha}{Z_{r_i}} \left[Z_{\ell_i} \left(|\boldsymbol{\ell}'_i|^2 \mathbf{I} - \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \right) + (\mathbf{b}_0 \cdot \boldsymbol{\ell}'_i)\, \mathbf{I} \right] (\mathbf{b}_0 \times \hat{v}_3) \tag{8.86}$$

We thus have

$$\mathbf{c}_{i0} \delta\mathbf{c}_i{}^{\mathrm{T}} = -\frac{\alpha}{Z_{r_i}^2} \mathbf{B}_{\times 0} \left[\left(Z_{\ell_i} |\boldsymbol{\ell}'_i|^2 \boldsymbol{\ell}'_i + \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \mathbf{b}_0 \right) (\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} - Z_{\ell_i} (\boldsymbol{\ell}'_i \cdot (\mathbf{b}_0 \times \hat{v}_3)) \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \right] \tag{8.87}$$

and

$$\begin{aligned}
\delta\mathbf{c}_i \delta\mathbf{c}_i{}^{\mathrm{T}} = \frac{\alpha^2}{Z_{r_i}^2} \Big[& \left(Z_{\ell_i}^2 |\boldsymbol{\ell}'_i|^4 + 2 Z_{\ell_i} |\boldsymbol{\ell}'_i|^2 \boldsymbol{\ell}'^{\mathrm{T}}_i \mathbf{b}_0 + \mathbf{b}_0^{\mathrm{T}} \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \mathbf{b}_0 \right) (\mathbf{b}_0 \times \hat{v}_3)(\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} \\
& - Z_{\ell_i}^2 |\boldsymbol{\ell}'_i|^2 \left(\boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i (\mathbf{b}_0 \times \hat{v}_3)(\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} + (\mathbf{b}_0 \times \hat{v}_3)(\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \right) \\
& - Z_{\ell_i} (\boldsymbol{\ell}'_i \cdot (\mathbf{b}_0 \times \hat{v}_3)) \left(\boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \mathbf{b}_0 (\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} + (\mathbf{b}_0 \times \hat{v}_3) \mathbf{b}_0^{\mathrm{T}} \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \right) \\
& + Z_{\ell_i}^2 (\boldsymbol{\ell}'_i \cdot (\mathbf{b}_0 \times \hat{v}_3))^2 \boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i \Big]
\end{aligned} \tag{8.88}$$

The above equations are written so that each term can be identified with one of the special integrals solved in Appendix B. Upon approximating the summations in equation (8.83) as integrals and using the fact that $\mathbf{b}_0 \cdot \hat{v}_3 = 0$, the solutions are found to be

$$\sum_{i=1}^{N} \left(\mathbf{c}_{i0} \delta \mathbf{c}_i{}^{\mathrm{T}} + \delta \mathbf{c}_i \mathbf{c}_{i0}^{\mathrm{T}} \right) \longrightarrow -2\alpha\kappa \overline{Z_\ell} N \left[\left(1 + \frac{D^2}{4} \right) (\mathbf{b}_0 \times \hat{v}_3)(\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} + \frac{D^2}{4} \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} \right] \tag{8.89}$$

and,

$$\begin{aligned} \sum_{i=1}^{N} \delta \mathbf{c}_i \delta \mathbf{c}_i{}^{\mathrm{T}} \quad \longrightarrow \quad & \alpha^2 N \left[\gamma \left(1 + \frac{D^2}{2} + \frac{D^4}{12} \right) + \frac{\kappa D^2}{4} \right] (\mathbf{b}_0 \times \hat{v}_3)(\mathbf{b}_0 \times \hat{v}_3)^{\mathrm{T}} \\ & + \alpha^2 \gamma N \frac{D^4}{24} \mathbf{I} + \alpha^2 \gamma N \frac{D^2}{4} \left(1 - \frac{D^2}{6} \right) \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} \end{aligned} \tag{8.90}$$

It is now clear that $\mathbf{C}'$ has the same eigenvectors as $\mathbf{C}_0$, namely $\mathbf{b}_0$, $\hat{v}_3$, and $\mathbf{b}_0 \times \hat{v}_3$. From equations (8.21), (8.20), (8.83), (8.89), and (8.89), the eigenvalues are given by

$$\mu'_{\mathbf{b}} = 0 \tag{8.91}$$

$$\mu'_{\hat{v}_3} = \frac{N D^2}{4} \left(\kappa - \alpha^2 \gamma \left(\frac{2\kappa \overline{Z_\ell}}{\alpha} - \gamma \right) \right) \tag{8.92}$$

and

$$\mu'_{\mathbf{b} \times \hat{v}_3} = \kappa N \left[1 - 2\alpha \overline{Z_\ell} \left(1 + \frac{D^2}{4} \right) + \alpha^2 \frac{D^2}{4} \right] + \alpha^2 \gamma N \left(1 + \frac{D^2}{2} + \frac{D^4}{8} \right) \tag{8.93}$$

The only difference between equations (8.92) and (8.75) is the factor $(2\kappa\overline{Z_\ell}/\alpha - \gamma)$ multiplying $\alpha^2\gamma$, which arises from the different manner in which intermediate terms were grouped in deriving these equations. Recalling that κ, γ, and $\overline{Z_\ell}$ are only convenient symbols representing unknown values, differences in terms involving products of these constants should not necessarily be considered significant. We note that if the distances in the image are constant and equal, *i.e.*, $Z_{\ell_i} = Z_{r_i} = Z$ for $i = 1, \ldots, N$, we can set $\alpha = 1/Z$ giving

$$\frac{2\kappa Z}{\alpha} - \gamma = 1 \tag{8.94}$$

and hence

$$\mu'_{\hat{v}_3} = S^* = 0 \tag{8.95}$$

We nonetheless expect $\mu'_{\hat{v}_3} \approx 0$ for many distributions of depths in the scene when $\mathbf{b}_0 \perp \hat{v}_3$. Given the fact that the two smallest eigenvalues of $\mathbf{C}'$ are both very close and both much less than $\mu'_{\mathbf{b}\times\hat{v}_3}$, we can see that with the inevitable errors in the data, it can easily happen that $\mu'_{\hat{v}_3} < \mu'_{\mathbf{b}_0}$.

From this derivation, we now see the basis for the test presented in the last chapter to determine if the algorithm has fallen into an alternate minimum. If it is true that $\mathbf{b}_0 \parallel \hat{v}_3$, then, as we saw previously in equation (8.25), the ratio of the middle to the largest eigenvalues is

$$\frac{\mu_2}{\mu_3} \approx 1 \tag{8.96}$$

If on the other hand $\mathbf{b}_0 \perp \hat{v}_3$ and $\mathbf{b}' = \hat{v}_3$, then

$$\frac{\mu_2}{\mu_3} \approx 0 \tag{8.97}$$

The ratio test is simple to compute and, based on numerous experiments, has proven to be an extremely reliable indicator of a false solution. It has consistently outperformed other measures which can be readily obtained from the data, such as S or K_C, the condition number of $\mathbf{C}$.

8.2 Error Analysis

Having analyzed the numerical stability of the procedures for estimating translation and rotation as a function of both the size of the viewing field and the type of motion, we can now quantify more precisely the robustness of the estimates as a function of the error in the data.

Errors in the data may arise from both random and systematic sources. Systematic error can usually be attributed to poor calibration of the imaging system while random errors result from the finite resolution of the image sensor and from approximations made in the matching procedure. Since the sensor is discretized, each pixel subtends a finite solid angle, which is approximated by a single vector from the center of projection to the center of the pixel. The block-matching procedure compounds the error due to the finite pixel size

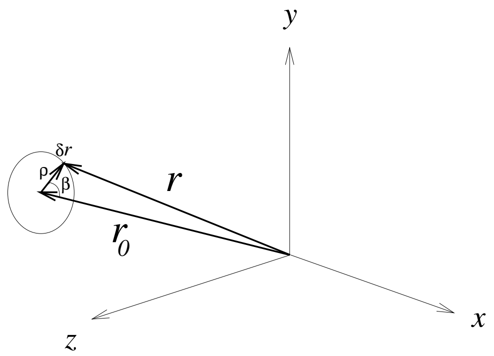


FIGURE 8-2: Change in $\mathbf{r}_i$ caused by error in determining the exact location of the correspondence with the left image.

by assigning the correspondence to the centers of the best matching blocks.

The combined result of both random and systematic errors is a displacement *in the image plane* of the estimated correpondence point location from its true position. We will assume that the position vectors $\boldsymbol{\ell}_i$ of the feature points in the left image are known exactly and model the error in the corresponding $\mathbf{r}_i$ by a vector $\delta\mathbf{r}_i$ which is added to the correct vector $\mathbf{r}_{i0}$,

$$\mathbf{r}_i = \mathbf{r}_{i0} + \delta\mathbf{r}_i \tag{8.98}$$

as shown in Figure 8-2. Since the error, whether systematic or random, is assumed to be uniform over the image, we again set the weights $w_i = 1$ in all of the following derivations.

We will consider the cases of random and systematic errors separately. For random errors we assume that the vectors $\delta\mathbf{r}_i$ are independent and identically distributed, since each block is matched independently of the others, and since neither the pixel size nor the block size used in the matching procedure is a function of position in the image plane. We write $\delta\mathbf{r}_i$ as

$$\delta\mathbf{r}_i = \begin{pmatrix} \rho_i \cos\beta_i \\ \rho_i \sin\beta_i \\ 0 \end{pmatrix} \tag{8.99}$$

where β_i is uniformly distributed over $[\,0, 2\pi)$ and ρ_i has some probability distribution over

$[\,0, R_{max}\,]$ with $E[\rho_i^2] = \sigma^2$. We thus have

$$E[\delta\mathbf{r}_i] = \mathbf{0} \tag{8.100}$$

and

$$E[\delta\mathbf{r}_i^{\mathrm{T}}\delta\mathbf{r}_j] = \begin{cases} \sigma^2, & i = j \\ 0, & i \neq j \end{cases} \tag{8.101}$$

while the covariance matrix, $\mathbf{\Lambda}$, is given by

$$\mathbf{\Lambda} = E[\delta\mathbf{r}_i\delta\mathbf{r}_j^{\mathrm{T}}] = \begin{cases} \dfrac{\sigma^2}{2}\left(\mathbf{I} - \hat{z}\hat{z}^{\mathrm{T}}\right), & i = j \\ \mathbf{0}, & i \neq j \end{cases} \tag{8.102}$$

Systematic errors are modeled by a constant vector $\delta\mathbf{r}$ added to all of the vectors $\mathbf{r}_{i0}$. In this case we have

$$E[\delta\mathbf{r}] = \rho\,\widehat{\delta\mathbf{r}}, \text{ and, } E[\delta\mathbf{r}^{\mathrm{T}}\delta\mathbf{r}] = \rho^2 \tag{8.103}$$

Since the error is constant, of course, its variance is zero.

We now examine the first-order effects of including these error terms on the estimates of the baseline and the rotation.

8.2.1 First-order error in the baseline (rotation known)

We consider first the case where the rotation is known exactly. From our definition of the error in equation (8.99), we can write the vectors $\mathbf{c}_i$, defined in (7.5), as

$$\begin{aligned} \mathbf{c}_i &= \boldsymbol{\ell}'_i \times \mathbf{r}_i \\ &= \boldsymbol{\ell}'_i \times \mathbf{r}_{i0} + \boldsymbol{\ell}'_i \times \delta\mathbf{r}_i \\ &= \mathbf{c}_{i0} + \delta\mathbf{c}_i \end{aligned} \tag{8.104}$$

We also have

$$\lambda_i = \mathbf{c}_i \cdot \mathbf{b} = \delta\mathbf{c}_i \cdot \mathbf{b} \tag{8.105}$$

since $\mathbf{c}_{i0} \cdot \mathbf{b} = 0$ by the definition of $\mathbf{c}_{i0}$ as the error-free vector $(\boldsymbol{\ell}'_i \times \mathbf{r}_{i0})$. We thus write $\mathbf{C}$ as

$$\begin{aligned}
\mathbf{C} &= \sum_{i=1}^{N} \mathbf{c}_i \mathbf{c}_i^{\mathrm{T}} \\
&= \sum_{i=1}^{N} \mathbf{c}_{i0} \mathbf{c}_{i0}^{\mathrm{T}} + \sum_{i=1}^{N} \left(\mathbf{c}_{i0} \delta \mathbf{c}_i^{\mathrm{T}} + \delta \mathbf{c}_i \mathbf{c}_{i0}^{\mathrm{T}} + \delta \mathbf{c}_i \delta \mathbf{c}_i^{\mathrm{T}} \right) \\
&= \mathbf{C}_0 + \Delta \mathbf{C} \qquad (8.106)
\end{aligned}$$

Dropping the last term, which is second-order in the error, we then write

$$\Delta \mathbf{C} \approx \sum_{i=1}^{N} \left(\mathbf{c}_{i0} \delta \mathbf{c}_i^{\mathrm{T}} + \delta \mathbf{c}_i \mathbf{c}_{i0}^{\mathrm{T}} \right) \qquad (8.107)$$

Let $\mathbf{b}_\delta$ denote the eigenvector corresponding to the smallest eigenvalue of $\mathbf{C}$, and let $\mathbf{b}_1$, $\mathbf{b}_2$, and $\mathbf{b}_3$ represent the eigenvectors of $\mathbf{C}_0$ with eigenvalues μ_1, μ_2, and μ_3, where

$$\mu_3 \geq \mu_2 \geq \mu_1 = 0 \qquad (8.108)$$

and

$$|\mathbf{b}_1| = |\mathbf{b}_2| = |\mathbf{b}_3| = 1 \qquad (8.109)$$

Using a result from matrix perturbation theory [91], we can express $\mathbf{b}_\delta$ by a first-order Taylor series expansion in terms of $\Delta\mathbf{C}$ and the unperturbed eigenvectors and eigenvalues of $\mathbf{C}_0$ as

$$\begin{aligned}
\mathbf{b}_\delta &\approx \mathbf{b}_1 - \left(\frac{\mathbf{b}_2^{\mathrm{T}} \Delta \mathbf{C} \mathbf{b}_1}{\mu_2} \right) \mathbf{b}_2 - \left(\frac{\mathbf{b}_3^{\mathrm{T}} \Delta \mathbf{C} \mathbf{b}_1}{\mu_3} \right) \mathbf{b}_3 \qquad (8.110) \\
&= \mathbf{b}_1 + \delta \mathbf{b}_1 \qquad (8.111)
\end{aligned}$$

The error vector $\delta\mathbf{b}_1$ is perpendicular to $\mathbf{b}_1$, and its magnitude, given by

$$|\delta \mathbf{b}_1| = \sqrt{ \left(\frac{\mathbf{b}_2^{\mathrm{T}} \Delta \mathbf{C} \mathbf{b}_1}{\mu_2} \right)^2 + \left(\frac{\mathbf{b}_3^{\mathrm{T}} \Delta \mathbf{C} \mathbf{b}_1}{\mu_3} \right)^2 } \qquad (8.112)$$

approximates the angle, θ_b, between $\mathbf{b}_1$ and $\mathbf{b}_\delta$.

The important quantity in determining the magnitude of the error is clearly the vector

$\Delta\mathbf{C}\mathbf{b}_1$. From (8.106) we have

$$\begin{aligned}\Delta\mathbf{C}\mathbf{b}_1 &= \sum_{i=1}^{N}\left(\mathbf{c}_{i0}\delta\mathbf{c}_i^{\mathrm{T}} + \delta\mathbf{c}_i\mathbf{c}_{i0}^{\mathrm{T}}\right)\mathbf{b}_1 \\ &= \sum_{i=1}^{N}\lambda_i\mathbf{c}_{i0} \qquad (8.113)\end{aligned}$$

Expressions for the estimation error in the cases of random and systematic measurement errors will now be derived separately.

Uncorrelated random error

When the error is uncorrelated, we have from equations (8.105) and (8.100)

$$\begin{aligned}E\left[\Delta\mathbf{C}\mathbf{b}_1\right] &= \sum_{i=1}^{N}E[\lambda_i]\mathbf{c}_{i0} \\ &= \sum_{i=1}^{N}(\mathbf{b}_1 \times \boldsymbol{\ell}'_i)^{\mathrm{T}}E\left[\delta\mathbf{r}_i\right]\mathbf{c}_{i0} = \mathbf{0} \qquad (8.114)\end{aligned}$$

giving also,

$$E\left[\delta\mathbf{b}_1\right] = \mathbf{0} \qquad (8.115)$$

The fact that the expected value of $\delta\mathbf{b}_1$ is zero only means that it has no preferred direction. The appropriate measure of the error is the *magnitude* of $\delta\mathbf{b}_1$ which is also, to first order, the angle θ_b between $\mathbf{b}_1$ and $\mathbf{b}_\delta$. We thus compute

$$\overline{\theta_b^2} \equiv E\left[\theta_b^2\right] = \frac{1}{\mu_2^2}\mathbf{b}_2^{\mathrm{T}}E\left[\Delta\mathbf{C}\mathbf{b}_1\mathbf{b}_1^{\mathrm{T}}\Delta\mathbf{C}\right]\mathbf{b}_2 + \frac{1}{\mu_3^2}\mathbf{b}_3^{\mathrm{T}}E\left[\Delta\mathbf{C}\mathbf{b}_1\mathbf{b}_1^{\mathrm{T}}\Delta\mathbf{C}\right]\mathbf{b}_3 \qquad (8.116)$$

From equation (8.113), we can write the term $\Delta\mathbf{C}\mathbf{b}_1\mathbf{b}_1^{\mathrm{T}}\Delta\mathbf{C}$ as

$$\Delta\mathbf{C}\mathbf{b}_1\mathbf{b}_1^{\mathrm{T}}\Delta\mathbf{C} = \sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j\mathbf{c}_{i0}\mathbf{c}_{j0}^{\mathrm{T}} \qquad (8.117)$$

and hence, by independence of the errors,

$$E\left[\Delta\mathbf{C}\mathbf{b}_1\mathbf{b}_1^{\mathrm{T}}\Delta\mathbf{C}\right] = \sum_{i=1}^{N}\sum_{j=1}^{N}E\left[\lambda_i\lambda_j\right]\mathbf{c}_{i0}\mathbf{c}_{j0}^{\mathrm{T}}$$

$$= \sum_{i=1}^{N} E\left[\lambda_i^2\right] \mathbf{c}_{i0}\mathbf{c}_{i0}^{\mathrm{T}} \tag{8.118}$$

We could approximate this sum as an integral, following the approach taken in Section 8.1.1, and derive an exact expression for $E\left[\Delta\mathbf{C}\mathbf{b}_1\mathbf{b}_1^{\mathrm{T}}\Delta\mathbf{C}\right]$. However, we can gain more insight into the problem by making the approximation that the term $E[\lambda_i^2]$ can be replaced by its average value $\overline{\lambda^2}$, where

$$\overline{\lambda^2} = \frac{1}{N}\sum_{i=1}^{N} E\left[\lambda_i^2\right] \tag{8.119}$$

Making the substitution in equation (8.118), we have

$$\begin{aligned} E\left[\Delta\mathbf{C}\mathbf{b}_1\mathbf{b}_1^{\mathrm{T}}\Delta\mathbf{C}\right] &= \sum_{i=1}^{N} \overline{\lambda^2}\,\mathbf{c}_{i0}\mathbf{c}_{i0}^{\mathrm{T}} \\ &= \overline{\lambda^2}\,\mathbf{C}_0 \end{aligned} \tag{8.120}$$

and hence the error $\overline{\theta_b^2}$ becomes

$$\begin{aligned} \overline{\theta_b^2} &= \frac{\overline{\lambda^2}}{\mu_2^2}\mathbf{b}_2^{\mathrm{T}}\mathbf{C}_0\mathbf{b}_2 + \frac{\overline{\lambda^2}}{\mu_3^2}\mathbf{b}_3^{\mathrm{T}}\mathbf{C}_0\mathbf{b}_3 \\ &= \overline{\lambda^2}\left(\frac{1}{\mu_2} + \frac{1}{\mu_3}\right) \end{aligned} \tag{8.121}$$

We now need only to find an expression for $\overline{\lambda^2}$. From equations (8.102) and (8.105), the expected value of λ_i^2 is

$$\begin{aligned} E\left[\lambda_i^2\right] &= E\left[(\mathbf{b}_1 \times \boldsymbol{\ell}'_i)^{\mathrm{T}}\delta\mathbf{r}_i\delta\mathbf{r}_j^{\mathrm{T}}(\mathbf{b}_1 \times \boldsymbol{\ell}'_j)\right] \\ &= \frac{\sigma^2}{2}(\mathbf{b}_1 \times \boldsymbol{\ell}'_i)^{\mathrm{T}}\left(\mathbf{I} - \hat{z}\hat{z}^{\mathrm{T}}\right)(\mathbf{b}_1 \times \boldsymbol{\ell}'_i) \\ &= \frac{\sigma^2}{2}\left[|\boldsymbol{\ell}'_i|^2 - (\boldsymbol{\ell}'_i \cdot \mathbf{b}_1)^2 - ((\mathbf{b}_1 \times \hat{z}) \cdot \boldsymbol{\ell}'_i)^2\right] \end{aligned} \tag{8.122}$$

Substituting this equation into (8.119) and approximating the sum by an integral we have

$$\overline{\lambda^2} \longrightarrow \frac{\sigma^2}{2\pi D^2} \int_0^D \int_0^{2\pi} \left[|\boldsymbol{\ell}'|^2 - (\boldsymbol{\ell}' \cdot \mathbf{b})^2 - ((\mathbf{b}_1 \times \hat{z}) \cdot \boldsymbol{\ell}')^2 \right] \xi \, d\xi \, d\alpha \tag{8.123}$$

Then, using the results of Appendix B and simplifying, we obtain

$$\overline{\lambda^2} = \frac{\sigma^2}{2} \left[|(\mathbf{b}_1 \times \hat{v}_3) \times \hat{z}|^2 + \frac{D^2}{4} \left(1 + (\mathbf{b}_1 \cdot \hat{v}_3)^2 - |(\mathbf{b}_1 \times \hat{z}) \times \hat{v}_3|^2 \right) \right] \tag{8.124}$$

Combining this expression with the formulas for the eigenvalues of $\mathbf{C}_0$ given in equations (8.20) and (8.21), we can write $\overline{\theta_b^2}$ in its most general form as

$$\begin{aligned} \overline{\theta_b^2} \quad = \quad & \frac{2\sigma^2}{\kappa N} \left[|(\mathbf{b}_1 \times \hat{v}_3) \times \hat{z}|^2 + \frac{D^2}{4} \left(1 + (\mathbf{b}_1 \cdot \hat{v}_3)^2 - |(\mathbf{b}_1 \times \hat{z}) \times \hat{v}_3|^2 \right) \right] \cdot \\ & \left(\frac{1}{D^2} + \frac{1}{D^2 (\mathbf{b}_1 \cdot \hat{v}_3)^2 + 4 |\mathbf{b}_1 \times \hat{v}_3|^2} \right) \end{aligned} \tag{8.125}$$

We observe that $\overline{\theta_b^2}$ is proportional to the error variance, σ^2, and to the squared distances of objects in the scene. (Recall that $\kappa = E[1/Z_r^2]$.) We can also see that $\overline{\theta_b^2} \to 0$ as $N \to \infty$ as should be expected for an unbiased estimator. The behavior of $\overline{\theta_b^2}$ as a function of D, however, depends on the orientation of $\mathbf{b}_1$ with respect to $\hat{v}_3$. When $\mathbf{b}_1 \perp \hat{v}_3$, equation (8.125) reduces to

$$\overline{\theta_b^2} = \frac{\sigma^2}{2\kappa N D^2} \left[|(\mathbf{b}_1 \times \hat{v}_3) \times \hat{z}|^2 + \frac{D^2}{4} |\hat{z} \times \hat{v}_3|^2 \right] \left(4 + D^2 \right) \tag{8.126}$$

As $D \to 0$ the term in $1/D^2$ dominates giving

$$\overline{\theta_b^2} \longrightarrow \frac{2\sigma^2}{\kappa N D^2} |(\mathbf{b}_1 \times \hat{v}_3) \times \hat{z}|^2 \tag{8.127}$$

so that $\overline{\theta_b^2} \to \infty$ as the field of view decreases. As D increases, however,

$$\overline{\theta_b^2} \longrightarrow \frac{\sigma^2}{2\kappa N} \left[|(\mathbf{b}_1 \times \hat{v}_3) \times \hat{z}|^2 + \left(1 + \frac{D^2}{4} \right) |\hat{z} \times \hat{v}_3|^2 \right] \tag{8.128}$$

Since $\hat{v}_3$ is usually very close to $\hat{z}$ we can neglect the term $|\hat{z} \times \hat{v}_3|^2$ for all reasonable values of D. We thus expect the error to become constant for large fields of view.

When $\mathbf{b}_1 = \hat{v}_3$, any vector perpendicular to $\hat{v}_3$ is an eigenvector of $\mathbf{C}_0$ with eigenvalue $\mu = \kappa N D^2/4$. The expected squared error thus becomes

$$\overline{\theta_b^2} = \frac{\sigma^2}{\kappa N}\left(2 - |\hat{z} \times \hat{v}_3|^2\right) \tag{8.129}$$

and is now completely independent of the field of view.

Systematic error

When the error vector $\delta\mathbf{r}$ is constant, equation (8.113) becomes

$$\begin{aligned}\Delta\mathbf{C}\mathbf{b}_1 &= \sum_{i=1}^{N} \lambda_i \mathbf{c}_{i0} \\ &= \left(\sum_{i=1}^{N} \mathbf{c}_{i0}(\mathbf{b}_1 \times \boldsymbol{\ell}'_i)^{\mathrm{T}}\right)\delta\mathbf{r}\end{aligned} \tag{8.130}$$

Using the rigid body motion equation (8.9), we can write

$$\mathbf{b} \times \boldsymbol{\ell}'_i = -Z_{r_i}\left(\boldsymbol{\ell}'_i \times \mathbf{r}_{i0}\right) \tag{8.131}$$

and hence

$$\Delta\mathbf{C}\mathbf{b}_1 = -\left(\sum_{i=1}^{N} Z_{r_i}\mathbf{c}_{i0}\mathbf{c}_{i0}^{\mathrm{T}}\right)\delta\mathbf{r} \tag{8.132}$$

We now approximate that the depths Z_{r_i} can be replaced by their average value $\overline{Z_r}$ so that

$$\begin{aligned}\Delta\mathbf{C}\mathbf{b}_1 &= -\overline{Z_r}\left(\sum_{i=1}^{N} \mathbf{c}_{i0}\mathbf{c}_{i0}^{\mathrm{T}}\right)\delta\mathbf{r} \\ &= -\overline{Z_r}\,\mathbf{C}_0\delta\mathbf{r}\end{aligned} \tag{8.133}$$

Substituting this expression into equation (8.110), we find that the estimated value of the baseline is given by

$$\begin{aligned}\mathbf{b}_\delta &\approx \mathbf{b}_1 + \overline{Z_r}\left(\frac{\mathbf{b}_2^{\mathrm{T}}\mathbf{C}_0\,\delta\mathbf{r}}{\mu_2}\right)\mathbf{b}_2 + \overline{Z_r}\left(\frac{\mathbf{b}_3^{\mathrm{T}}\mathbf{C}_0\,\delta\mathbf{r}}{\mu_3}\right)\mathbf{b}_3 \\ &= \mathbf{b}_1 + \overline{Z_r}\left((\mathbf{b}_2 \cdot \delta\mathbf{r})\mathbf{b}_2 + (\mathbf{b}_3 \cdot \delta\mathbf{r})\mathbf{b}_3\right)\end{aligned} \tag{8.134}$$

which can also be written as

$$\mathbf{b}_\delta = \mathbf{b}_1 + \overline{Z_r}\,(\delta\mathbf{r} - (\mathbf{b}_1 \cdot \delta\mathbf{r})\mathbf{b}_1) \tag{8.135}$$

This result makes perfect sense, of course, since it states that the adjustment to the baseline is along the component of $\delta\mathbf{r}$ which is perpendicular to $\mathbf{b}_1$. The angular error, θ_b, is given by

$$\begin{aligned} \theta_b &\approx |\delta\mathbf{b}| \\ &= \overline{Z_r}\,|\mathbf{b}_1 \times (\mathbf{b}_1 \times \delta\mathbf{r})| \\ &= \overline{Z_r}|\mathbf{b}_1 \times \delta\mathbf{r}| \end{aligned} \tag{8.136}$$

With systematic measurement errors, the estimation error is still a function of the distances to objects in the scene, but no longer depends on either the field of view or the number of correspondence points N.

8.2.2 First-order error in the rotation (baseline known)

We now look at the case where the translation direction is known and examine the error in the rotation estimate. To first order, this error can be associated with the incremental adjustment computed by the UPDATE_Q() procedure when the correct $\mathbf{b}$ and $\mathring{q}$ are input. Let $\mathring{q}_\delta$ denote the updated rotation quaternion and $\mathring{q}$ its true value. Using the notation of Section 7.1.2

$$\mathring{q}_\delta = \delta\mathring{q}\,\mathring{q} \tag{8.137}$$

with

$$\delta\mathring{q} = \left(\cos\frac{\delta\theta}{2}\,,\, \widehat{\boldsymbol{\eta}}\,\sin\frac{\delta\theta}{2}\right) \tag{8.138}$$

The error quaternion $\delta\mathring{q}$ corresponds to an additional rotation of $\delta\theta$ about an axis $\widehat{\boldsymbol{\eta}}$ applied to the true rotation $\mathring{q}$. From equations (7.12), (7.15), and (7.17), $\delta\theta$ and $\widehat{\boldsymbol{\eta}}$ are computed from the vector $\mathbf{m}$, given by

$$\mathbf{m} \equiv -\delta\theta\,\widehat{\boldsymbol{\eta}} = \mathbf{A}^{-1}\mathbf{h} \tag{8.139}$$

where

$$\mathbf{h} = \sum_{i=1}^{N} \lambda_i \, \mathbf{a}_i = \sum_{i=1}^{N} \lambda_i \, (\mathbf{b} \times \mathbf{r}_i) \times \boldsymbol{\ell}'_i \tag{8.140}$$

We can write $\mathbf{a}_i$ as

$$\mathbf{a}_i = \mathbf{a}_{i0} + \delta \mathbf{a}_i \tag{8.141}$$

with

$$\mathbf{a}_{i0} = (\mathbf{b} \times \mathbf{r}_{i0}) \times \boldsymbol{\ell}'_i, \text{ and } \delta \mathbf{a}_i = (\mathbf{b} \times \delta \mathbf{r}_i) \times \boldsymbol{\ell}'_i \tag{8.142}$$

The term $\lambda_i \delta \mathbf{a}_i$ is second-order in the error, however, and so is dropped, leaving

$$\mathbf{h} \approx \sum_{i=1}^{N} \lambda_i \, \mathbf{a}_{i0}, \text{ and } \mathbf{A}_0 = \sum_{i=1}^{N} \mathbf{a}_{i0} \mathbf{a}_{i0}^{\mathrm{T}} \tag{8.143}$$

Just as the magnitude of the error in the translation estimate is given by the angle between $\mathbf{b}_\delta$ and $\mathbf{b}$, the magnitude of the rotation error vector, $|\mathbf{m}| = |\delta\theta|$, is related to the angle between $\mathring{q}_\delta$ and $\mathring{q}$. Taking their inner product, we find

$$\begin{aligned} \mathring{q}_\delta \cdot \mathring{q} &= (\delta \mathring{q} \, \mathring{q}) \cdot \mathring{q} \\ &= \delta \mathring{q} \cdot (\mathring{q} \, \mathring{q}^*) \\ &= \delta \mathring{q} \cdot \mathring{e} \\ &= \cos \frac{\delta\theta}{2} \end{aligned} \tag{8.144}$$

using identities (A.10) and (A.7) from Appendix A.

The squared estimation error is thus given by

$$|\delta\theta|^2 = \mathbf{m}^{\mathrm{T}} \mathbf{m} = \mathbf{h}^{\mathrm{T}} \mathbf{A}_0^{-1} \mathbf{A}_0^{-1} \mathbf{h} \tag{8.145}$$

which can be simplified by expanding $\mathbf{A}_0$ and $\mathbf{A}_0^{-1}$ in terms of their eigenvectors. Let $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$ denote the eigenvectors of $\mathbf{A}_0$ with eigenvalues μ_1, μ_2, and μ_3, where

$$\mu_3 \geq \mu_2 \geq \mu_1 \tag{8.146}$$

Because $\mathbf{A}_0$ and $\mathbf{A}_0^{-1}$ are symmetric, they can be diagonalized as

$$\mathbf{A}_0 = \mathbf{S}\mathbf{V}\mathbf{S}^{\mathrm{T}}, \text{ and } \quad \mathbf{A}_0^{-1} = \mathbf{S}\mathbf{V}^{-1}\mathbf{S}^{\mathrm{T}} \tag{8.147}$$

where

$$\mathbf{S} = (\mathbf{w}_1\,\mathbf{w}_2\,\mathbf{w}_3)\,, \; \mathbf{V} = \mathrm{diag}\,(\mu_1, \mu_2, \mu_3)\,, \text{ and } \mathbf{S}^{\mathrm{T}}\mathbf{S} = \mathbf{I} \tag{8.148}$$

so that

$$\mathbf{h}^{\mathrm{T}}\mathbf{A}_0^{-1}\mathbf{A}_0^{-1}\mathbf{h} = \frac{1}{\mu_1^2}(\mathbf{w}_1 \cdot \mathbf{h})^2 + \frac{1}{\mu_2^2}(\mathbf{w}_2 \cdot \mathbf{h})^2 + \frac{1}{\mu_3^2}(\mathbf{w}_3 \cdot \mathbf{h})^2 \tag{8.149}$$

The expression for the estimation error will depend on whether the errors in the data are systematic or random. These cases are now examined separately.

Uncorrelated random error

With random measurement errors, the expected value of $\mathbf{m}$ is zero since

$$E[\mathbf{m}] = \mathbf{A}_0^{-1}E[\mathbf{h}] = \mathbf{A}_0^{-1}\sum_{i=1}^{N} E[\lambda_i]\,\mathbf{a}_{i0} = \mathbf{0} \tag{8.150}$$

which implies, as we should expect, that $\mathbf{m}$ has no preferred direction. The variance of the error is thus $E[\delta\theta^2]$, which can be written as

$$\overline{\delta\theta^2} = E[\delta\theta^2] = \frac{1}{\mu_1^2}\mathbf{w}_1^{\mathrm{T}}E[\mathbf{h}\mathbf{h}^{\mathrm{T}}]\mathbf{w}_1 + \frac{1}{\mu_2^2}\mathbf{w}_2^{\mathrm{T}}E[\mathbf{h}\mathbf{h}^{\mathrm{T}}]\mathbf{w}_2 + \frac{1}{\mu_3^2}\mathbf{w}_3^{\mathrm{T}}E[\mathbf{h}\mathbf{h}^{\mathrm{T}}]\mathbf{w}_3 \tag{8.151}$$

The important quantity to compute is clearly $E[\mathbf{h}\mathbf{h}^{\mathrm{T}}]$. Using (8.143), we have

$$\begin{aligned} E\left[\mathbf{h}\mathbf{h}^{\mathrm{T}}\right] &= E\left[\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j\,\mathbf{a}_{i0}\mathbf{a}_{j0}^{\mathrm{T}}\right] \\ &= \sum_{i=1}^{N} E\left[\lambda_i^2\right]\,\mathbf{a}_{i0}\mathbf{a}_{i0}^{\mathrm{T}} \end{aligned} \tag{8.152}$$

We again make the approximation that $E[\lambda_i^2]$ can be replaced by its average value $\overline{\lambda^2}$ so that

$$
\begin{aligned}
E\left[\mathbf{h}\mathbf{h}^{\mathrm{T}}\right] &= \overline{\lambda^2}\sum_{i=1}^{N}\mathbf{a}_{i0}\mathbf{a}_{i0}^{\mathrm{T}} \\
&= \overline{\lambda^2}\mathbf{A}_0
\end{aligned} \tag{8.153}
$$

giving

$$
\begin{aligned}
\overline{\delta\theta^2} &= \frac{\overline{\lambda^2}}{\mu_1^2}\mathbf{w}_1^{\mathrm{T}}\mathbf{A}_0\mathbf{w}_1 + \frac{\overline{\lambda^2}}{\mu_2^2}\mathbf{w}_2^{\mathrm{T}}\mathbf{A}_0\mathbf{w}_2 + \frac{\overline{\lambda^2}}{\mu_3^2}\mathbf{w}_3^{\mathrm{T}}\mathbf{A}_0\mathbf{w}_3 \\
&= \overline{\lambda^2}\left(\frac{1}{\mu_1} + \frac{1}{\mu_2} + \frac{1}{\mu_3}\right)
\end{aligned} \tag{8.154}
$$

which is completely analogous to equation (8.121) for the error in the baseline estimate.

We can find an expression for $\overline{\delta\theta^2}$ in the special cases $\mathbf{b} \perp \hat{v}_3$ and $\mathbf{b} = \hat{v}_3$ for which we previously derived the eigenvalues of $\mathbf{A}_0$ in Section 8.1.2. From equations (8.43)–(8.45) we have in the case $\mathbf{b} \perp \hat{v}_3$

$$\mu_1 = \frac{\gamma N D^4}{24} \tag{8.155}$$

$$\mu_2 = \frac{\gamma N D^2}{4} \tag{8.156}$$

$$\mu_3 = \gamma N\left(1 + \frac{D^2}{2} + \frac{D^4}{8}\right) \tag{8.157}$$

while $\overline{\lambda^2}$, given in (8.124), simplifies to

$$\overline{\lambda^2} = \frac{\sigma^2}{2}\left[|(\mathbf{b}\times\hat{v}_3)\times\hat{z}|^2 + \frac{D^2}{4}|\hat{v}_3\times\hat{z}|^2\right] \tag{8.158}$$

The variance of the error is therefore

$$
\begin{aligned}
\overline{\delta\theta^2} &= \frac{2\sigma^2}{\gamma N}\left[|(\mathbf{b}\times\hat{v}_3)\times\hat{z}|^2 + \frac{D^2}{4}|\hat{v}_3\times\hat{z}|^2\right]\left(\frac{6}{D^4} + \frac{1}{D^2} + \frac{2}{8+4D^2+D^4}\right) \\
&\approx \frac{2\sigma^2}{\gamma N}\left(\frac{6}{D^4} + \frac{1}{D^2} + \frac{8}{8+4D^2+D^4}\right)
\end{aligned} \tag{8.159}
$$

after dropping the term $|\hat{v}_3 \times \hat{z}|$ which is usually negligible.

We see that $\overline{\delta\theta^2}$ is proportional to the variance of the error in the data, σ^2, and inversely proportional to the number of correspondences, N. As $D \to 0$, the error in the rotation estimate increases rapidly ($\overline{\delta\theta^2} \sim 1/D^4$), while for large values of D, it eventually goes to zero.

When $\mathbf{b} = \hat{v}_3$ we have

$$\mu_1 = \mu_2 = \frac{\gamma N D^2}{4}, \text{ and } \mu_3 = \frac{\gamma N D^4}{3} \tag{8.160}$$

and

$$\overline{\lambda^2} = \frac{\sigma^2 D^2}{8}\left(2 - |\hat{z} \times \hat{v}_3|^2\right) \tag{8.161}$$

The expression for $\overline{\delta\theta^2}$ thus becomes

$$\overline{\delta\theta^2} = \frac{\sigma^2}{\gamma N}\left(2 - |\hat{z} \times \hat{v}_3|^2\right)\left(1 + \frac{3}{8D^2}\right) \tag{8.162}$$

Again, $\overline{\delta\theta^2}$ is proportional to σ^2/N, however, its behavior as a function of D is much less severe than in the case of $\mathbf{b} \perp \hat{v}_3$. As $D \to 0$, $\overline{\delta\theta^2}$ increases only as $1/D^2$, and as $D \to \infty$, it approaches a constant value.

Systematic error

In the case of systematic measurement errors, we must derive an exact expression for $\mathbf{h}$ in order to obtain a formula for $|\delta\theta|^2$. From equations (8.105), (8.142), and (8.143), we have

$$\begin{aligned} \mathbf{h} &= \sum_{i=1}^{N} \lambda_i \, \mathbf{a}_{i0} \\ &= \sum_{i=1}^{N} ((\mathbf{b} \times \mathbf{r}_{i0}) \times \boldsymbol{\ell}')\boldsymbol{\ell}' \cdot (\delta\mathbf{r} \times \mathbf{b}) \end{aligned} \tag{8.163}$$

Using (8.31), we can also write

$$\mathbf{h} = \sum_{i=1}^{N} \frac{Z_{\ell_i}}{Z_{r_i}} \left((\mathbf{b} \cdot \boldsymbol{\ell}'_i)\boldsymbol{\ell}'_i \boldsymbol{\ell}'^{\mathrm{T}}_i - |\boldsymbol{\ell}'_i|^2 \mathbf{b} \boldsymbol{\ell}'^{\mathrm{T}}_i\right)(\delta\mathbf{r} \times \mathbf{b}) \tag{8.164}$$

Approximating Z_{ℓ_i}/Z_{r_i} by its average value and assuming that the correspondences are distributed uniformly over the field of view, we replace the summation by an integral to obtain

$$\mathbf{h} \longrightarrow \left(\overline{\frac{Z_\ell}{Z_r}}\right) \frac{N}{\pi D^2} \int_0^D \int_0^{2\pi} \left((\mathbf{b} \cdot \boldsymbol{\ell}')\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}} - |\boldsymbol{\ell}'|^2 \mathbf{b}\boldsymbol{\ell}'^{\mathrm{T}}\right) (\delta\mathbf{r} \times \mathbf{b})\, \xi \, d\xi \, d\alpha \tag{8.165}$$

The solution to this integral can be obtained by combining equations (B.12) and (B.28) of Appendix B. Skipping the algebra, the result is:

$$\mathbf{h} = \left(\overline{\frac{Z_\ell}{Z_r}}\right) N \left[(\mathbf{b} \cdot \hat{v}_3) \left(\frac{D^2}{4}\mathbf{I} + \left(1 - \frac{3D^2}{4}\right) \hat{v}_3 \hat{v}_3{}^{\mathrm{T}}\right) - \left(1 + \frac{D^2}{4}\right) \mathbf{b}\hat{v}_3{}^{\mathrm{T}}\right] (\delta\mathbf{r} \times \mathbf{b}) \tag{8.166}$$

We note also that we can express the average value of λ_i as

$$\begin{aligned} \overline{\lambda} \;&\equiv\; \frac{1}{N}\sum_{i=1}^{N} \lambda_i \longrightarrow \frac{1}{\pi D^2} \int_0^D \int_0^{2\pi} (\delta\mathbf{r} \times \mathbf{b})^{\mathrm{T}} \boldsymbol{\ell}' \, \xi \, d\xi \, d\alpha \\ &=\; (\delta\mathbf{r} \times \mathbf{b}) \cdot \hat{v}_3 \end{aligned} \tag{8.167}$$

Using this expression in equation (8.166) then gives

$$\mathbf{h} = N \left(\overline{\frac{Z_\ell}{Z_r}}\right) \left[(\mathbf{b} \cdot \hat{v}_3) \left(\frac{D^2}{4}(\delta\mathbf{r} \times \mathbf{b}) + \overline{\lambda} \left(1 - \frac{3D^2}{4}\right) \hat{v}_3\right) - \overline{\lambda} \left(1 + \frac{D^2}{4}\right) \mathbf{b}\right] \tag{8.168}$$

Again, the behavior of the error as a function of D depends on the orientation of $\mathbf{b}$ and $\hat{v}_3$. When $\mathbf{b} \perp \hat{v}_3$, $\mathbf{h}$ becomes

$$\mathbf{h} = -\overline{\lambda} N \left(\overline{\frac{Z_\ell}{Z_r}}\right) \left(1 + \frac{D^2}{4}\right) \mathbf{b} \tag{8.169}$$

Because $\mathbf{b}$ is an eigenvector of $\mathbf{A}$ with eigenvalue

$$\mu_{\mathbf{b}} = \gamma N \left(1 + \frac{D^2}{2} + \frac{D^4}{8}\right) \tag{8.170}$$

we find from equation (8.149) that the error $|\delta\theta|$ becomes

$$|\delta\theta| = \frac{2|\overline{\lambda}|}{\gamma}\left(\frac{\overline{Z_\ell}}{Z_r}\right)\frac{4+D^2}{8+4D^2+D^4} \tag{8.171}$$

The error does not depend on N, but does depend on the field of view. As $D \to 0$, the error becomes infinite as $1/D^2$ while as $D \to \infty$, $|\delta\theta| \to 0$.

When $\mathbf{b} \parallel \hat{v}_3$, $\mathbf{h}$ becomes

$$\mathbf{h} = \frac{ND^2}{4}\left(\frac{\overline{Z_\ell}}{Z_r}\right)(\delta\mathbf{r}\times\mathbf{b}) \tag{8.172}$$

In this case, any vector perpendicular to $\mathbf{b}$ is an eigenvector of $\mathbf{A}$ with eigenvalue $\mu_\perp = \gamma ND^2/4$. We thus have

$$\mathbf{m} = \frac{1}{\gamma}\left(\frac{\overline{Z_\ell}}{Z_r}\right)(\delta\mathbf{r}\times\mathbf{b}) \tag{8.173}$$

and

$$|\delta\theta| = |\mathbf{m}| = \frac{1}{\gamma}\left(\frac{\overline{Z_\ell}}{Z_r}\right)|\delta\mathbf{r}\times\mathbf{b}| \tag{8.174}$$

The error is now independent of both D and N.

8.2.3 Coupling between estimation errors

In the last two sections we have analyzed the estimation errors in the cases when either the baseline or the rotation is known. We now examine the situation when neither is known. Let S^* denote the minimum value of $S = \sum_{i=1}^N \lambda_i^2$, and let S_0 denote the value when the correct $\mathbf{b}$ and $\mathring{q}$ corresponding to the actual motion are used. We can expand S in a first-order Taylor series to approximate S^* as

$$S^* \approx S_0 + \left.\frac{\partial S}{\partial \mathbf{b}}\right|_{S=S_0}^{\mathrm{T}} \delta\mathbf{b} + \left.\frac{\partial S}{\partial \mathbf{m}}\right|_{S=S_0}^{\mathrm{T}} \mathbf{m} \tag{8.175}$$

Assuming S^* corresponds to a true local minimum and is therefore unique, this expression defines a relation between $\delta\mathbf{b}$ and $\mathbf{m}$ which must be (approximately) satisfied to achieve optimality. The values of $\delta\mathbf{b}$ and $\mathbf{m}$ used to obtain S^* from S_0 cannot therefore be the same as those which minimize the error in the case of known rotation or known translation since, by definition, each of these assumes the other to be zero.

In order to determine a constraint between $\delta\mathbf{b}$ and $\mathbf{m}$, we need to use the necessary conditions for S^* to be a (local) minimum, which are

$$\mathbf{C}\mathbf{b} = \mu \mathbf{b} \tag{8.176}$$

where μ is the smallest eigenvalue of $\mathbf{C}$, and

$$\mathbf{h} = \sum_{i=1}^{N} \lambda_i \, \mathbf{a}_i = \mathbf{0} \tag{8.177}$$

Since there is always at least one solution to the eigenvector equation (8.176) for every rotation, it does not provide any new information. We thus need to find solutions to (8.176) that are also consistent with (8.177).

Defining $\mathbf{b}_0$ as the true translation vector, we have at $S = S^*$

$$\begin{aligned} \lambda_i &= \mathbf{b} \cdot (\boldsymbol{\ell}'_i \times \mathbf{r}_i) \\ &= (\mathbf{b}_0 + \delta\mathbf{b}) \cdot ((\boldsymbol{\ell}'_{i0} + \delta\boldsymbol{\ell}'_i) \times (\mathbf{r}_{i0} + \delta\mathbf{r}_i)) \\ &\approx \delta\mathbf{b} \cdot (\boldsymbol{\ell}'_{i0} \times \mathbf{r}_{i0}) + \mathbf{b}_0 \cdot (\delta\boldsymbol{\ell}'_i \times \mathbf{r}_{i0}) + \mathbf{b}_0 \cdot (\boldsymbol{\ell}'_{i0} \times \delta\mathbf{r}_i) \end{aligned} \tag{8.178}$$

dropping terms that are second-order and higher in the error. Since λ_i contains all of the first-order error, we also consider $\mathbf{a}_i \approx \mathbf{a}_{i0}$.

From equations (7.10) and (7.12) we obtain

$$\delta\boldsymbol{\ell}'_i = \boldsymbol{\ell}'_{i0} \times \mathbf{m}_0 \tag{8.179}$$

where $\mathbf{m}_0$ represents the incremental rotation vector applied to the true rotation to get to $S = S^*$, and can thus write

$$\mathbf{b}_0 \cdot (\delta\boldsymbol{\ell}'_i \times \mathbf{r}_{i0}) = \mathbf{b}_0 \cdot ((\boldsymbol{\ell}'_{i0} \times \mathbf{m}_0) \times \mathbf{r}_{i0}) = -\mathbf{a}_{i0}^{\mathrm{T}} \mathbf{m}_0 \tag{8.180}$$

Using also the fact that

$$\mathbf{b}_0 \times \boldsymbol{\ell}'_i = -Z_{r_i} \, (\boldsymbol{\ell}'_i \times \mathbf{r}_{i0}) = -Z_{r_i} \mathbf{c}_{i0} \tag{8.181}$$

equation (8.178) becomes

$$\begin{aligned}
\lambda_i &= \delta\mathbf{b}\cdot\mathbf{c}_{i0} - \mathbf{a}_{i0}^{\mathrm{T}}\mathbf{m}_0 - Z_{r_i}\mathbf{c}_{i0}\cdot\delta\mathbf{r}_i \\
&= \mathbf{c}_{i0}^{\mathrm{T}}\left(\delta\mathbf{b} - Z_{r_i}\delta\mathbf{r}_i\right) - \mathbf{a}_{i0}^{\mathrm{T}}\mathbf{m}_0 \qquad (8.182)
\end{aligned}$$

Combining these expressions, we find that the necessary condition for optimality is thus

$$\sum_{i=1}^{N}\mathbf{a}_{i0}\mathbf{c}_{i0}^{\mathrm{T}}\left(\delta\mathbf{b} - Z_{r_i}\delta\mathbf{r}_i - \mathbf{a}_{i0}^{\mathrm{T}}\mathbf{m}_0\right) = \mathbf{0} \qquad (8.183)$$

which can also be written as

$$\sum_{i=1}^{N}\mathbf{a}_{i0}\mathbf{c}_{i0}^{\mathrm{T}}\left(\delta\mathbf{b} - Z_{r_i}\delta\mathbf{r}_i\right) = \sum_{i=1}^{N}\mathbf{a}_{i0}\mathbf{a}_{i0}^{\mathrm{T}}\mathbf{m}_0 = \mathbf{A}_0\mathbf{m}_0 = \mathbf{h}_0 \qquad (8.184)$$

where $\mathbf{h}_0$ is the value of $\mathbf{h}$ at $S = S_0$.

If the measurement errors are random and uncorrelated, we have from equations (8.100), (8.115) and (8.150),

$$E[\delta\mathbf{r}_i] = E[\delta\mathbf{b}] = E[\mathbf{h}_0] = \mathbf{0} \qquad (8.185)$$

and hence equation (8.184) does not provide any new information. With random measurement errors, the expected values of $\delta\mathbf{b}$ and $\mathbf{m}$ are zero whether the rotation and translation are estimated separately or together. Consequently, equations (8.121) and (8.154) for the variances of the estimates, $\overline{\theta_b^2}$ and $\overline{\delta\theta^2}$, are also still valid.

With systematic measurement errors, however, the situation is different. We can write

$$\begin{aligned}
\sum_{i=1}^{N}\mathbf{a}_{i0}\mathbf{c}_{i0}^{\mathrm{T}} &= \sum_{i=1}^{N} -\frac{Z_{l_i}}{Z_{r_i}^2}((\mathbf{b}_0\times\boldsymbol{\ell}'_{i0})\times\boldsymbol{\ell}'_{i0})(\mathbf{b}_0\times\boldsymbol{\ell}'_{i0})^{\mathrm{T}} \\
&= \sum_{i=1}^{N}\frac{Z_{l_i}}{Z_{r_i}^2}\left((\mathbf{b}_0\cdot\boldsymbol{\ell}'_{i0})\boldsymbol{\ell}'_{i0}\boldsymbol{\ell}'^{\mathrm{T}}_{i0} - |\boldsymbol{\ell}'_{i0}|^2\mathbf{b}_1\boldsymbol{\ell}'^{\mathrm{T}}_{i0}\right)\mathbf{B}_{\times 0} \qquad (8.186)
\end{aligned}$$

where $\mathbf{B}_{\times 0}$ is the cross-product matrix corresponding to the operation $\mathbf{b}_0\times$, and combine equations (8.184) and (8.186) to give

$$\sum_{i=1}^{N} \frac{Z_{l_i}}{Z_{r_i}} \left((\mathbf{b}_0 \cdot \boldsymbol{\ell}'_{i0}) \boldsymbol{\ell}'_{i0} \boldsymbol{\ell}'^{\mathrm{T}}_{i0} - |\boldsymbol{\ell}'_{i0}|^2 \mathbf{b}_0 \boldsymbol{\ell}'^{\mathrm{T}}_{i0} \right) \left(\frac{1}{Z_{r_i}} (\mathbf{b}_0 \times \delta \mathbf{b}) + (\delta \mathbf{r} \times \mathbf{b}_0) \right) = \mathbf{h}_0 \qquad (8.187)$$

Comparing this equation with the definition of $\mathbf{h}_0$ given in equation (8.164), we see that a solution exists only if $\delta \mathbf{b} = \mathbf{0}$, which implies that the estimation error is entirely absorbed by the rotation.

This is a significant practical result as it implies that one can obtain very accurate estimates of the translation, even with poorly calibrated systems, when the full algorithm is used. We have to be careful, however, before concluding that $\delta \mathbf{b}$ is always zero when the rotation and translation are estimated jointly, because $\mathbf{h}$ is only a linear approximation to the derivative of S with respect to the rotation. We can write S_0 as

$$\begin{aligned} S_0 &= \sum_{i=1}^{N} ((\delta \mathbf{r} \times \mathbf{b}_0) \cdot \boldsymbol{\ell}'_{i0})^2 \\ &= \sum_{i=1}^{N} (\delta \mathbf{r} \times \mathbf{b}_0)^{\mathrm{T}} \boldsymbol{\ell}'_{i0} \boldsymbol{\ell}'^{\mathrm{T}}_{i0} (\delta \mathbf{r} \times \mathbf{b}_0) \\ &\longrightarrow \frac{N}{\pi D^2} \int_0^D \int_0^{2\pi} (\delta \mathbf{r} \times \mathbf{b}_0)^{\mathrm{T}} \boldsymbol{\ell}'_{i0} \boldsymbol{\ell}'^{\mathrm{T}}_{i0} (\delta \mathbf{r} \times \mathbf{b}_0)\, \xi \, d\xi \, d\alpha \\ &= N \overline{\lambda}^2 + \frac{N D^2}{4} \left(|\delta \mathbf{r} \times \mathbf{b}_0|^2 - \overline{\lambda}^2 \right) \qquad (8.188) \end{aligned}$$

using the solution for the integral given in equation (B.19) and the definition $\overline{\lambda} = (\delta \mathbf{r} \times \mathbf{b}_0) \cdot \hat{v}_3$ from equation (8.167).

When $\mathbf{b}_0 \parallel \hat{v}_3$, $\overline{\lambda} = 0$, so that

$$S_0 = \frac{N D^2}{4} \left| \delta \mathbf{r} \times \mathbf{b}_0 \right|^2 \qquad (8.189)$$

From equations (8.172) and (8.173) we have

$$\mathbf{h}_0^{\mathrm{T}} \mathbf{m}_0 = \frac{1}{\gamma} \left(\frac{\overline{Z_\ell}}{Z_r} \right)^2 \frac{N D^2}{4} |\delta \mathbf{r} \times \mathbf{b}|^2 \approx \frac{N D^2}{4} |\delta \mathbf{r} \times \mathbf{b}|^2 \qquad (8.190)$$

after setting

$$\frac{1}{\gamma}\left(\frac{\overline{Z_\ell}}{Z_r}\right)^2 \approx 1 \tag{8.191}$$

and thus when $\mathbf{b}_0$ and $\hat{v}_3$ are aligned, the quantity $-\mathbf{h}_0^{\mathrm{T}}\mathbf{m}_0$ exactly cancels S_0.

When $\mathbf{b}_0 \perp \hat{v}_3$, however, the term $|\delta\mathbf{r} \times \mathbf{b}_0|^2 - \overline{\lambda}^2$ in the equation for S_0 becomes

$$\begin{aligned} |\delta\mathbf{r} \times \mathbf{b}_0|^2 - \overline{\lambda}^2 &= |\delta\mathbf{r} \times \mathbf{b}_0|^2 - ((\delta\mathbf{r} \times \mathbf{b}_0)\cdot \hat{v}_3)^2 \\ &= |(\delta\mathbf{r} \times \mathbf{b}_0) \times \hat{v}_3|^2 \\ &= (\delta\mathbf{r} \cdot \hat{v}_3)^2 \\ &\simeq 0 \end{aligned} \tag{8.192}$$

since $\delta\mathbf{r}$ is parallel to the image plane and $\hat{v}_3 \approx \hat{z}$. We thus have

$$S_0 \simeq N\overline{\lambda}^2 \tag{8.193}$$

while equations (8.169) and (8.171) combine to give

$$\begin{aligned} \mathbf{h}_0^{\mathrm{T}}\mathbf{m}_0 &= \frac{1}{2\gamma}\left(\frac{\overline{Z_\ell}}{Z_r}\right)^2 N\overline{\lambda}^2 \frac{(4+D^2)^2}{8+4D^2+D^4} \\ &\approx N\overline{\lambda}^2\left(1 - \frac{D^4}{16+8D^2+2D^4}\right) \end{aligned} \tag{8.194}$$

The quantity $-\mathbf{h}_0^{\mathrm{T}}\mathbf{m}_0$ now only cancels S_0 in the limit as $D \to 0$. We thus conclude that the constraint equation (8.187) is only a loose approximation in the case of $\mathbf{b}_0 \perp \hat{v}_3$ since the nonlinear terms which were neglected must become significant for large values of D in order for the conditions of optimality to be obtained.

8.3 Experimental Verification

In order to test the results derived in this chapter, it was necessary to generate a random data set of 3-D coordinates corresponding to world points in the scene. This was done by first generating the Z values according to a specified probability distribution and then selecting the X and Y values so that the position vectors would be uniformly distributed

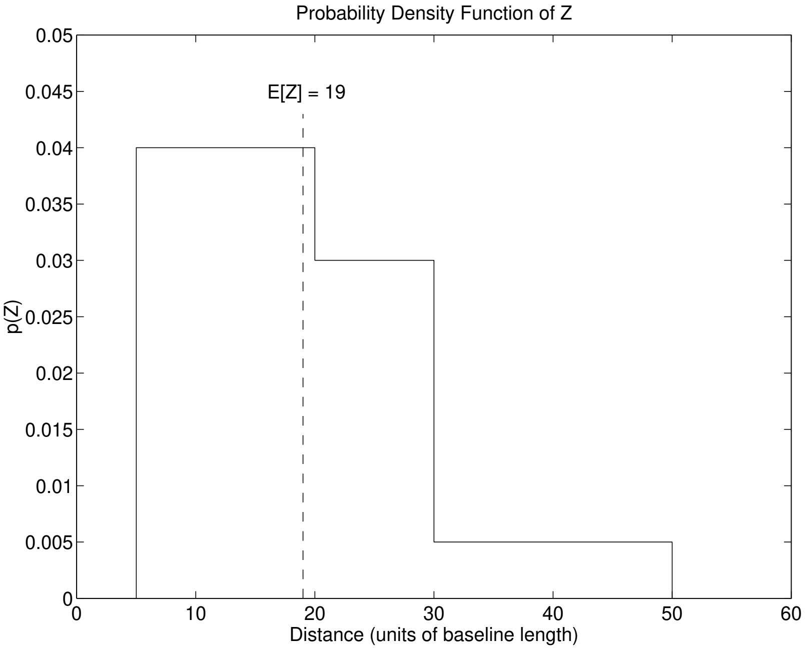


FIGURE 8-3: Probability density function used to obtain random Z values.

over the specified field of view.

The probability density function used to assign Z values for the tests is shown in Figure 8-3 and was determined based on a rough estimate of the distribution of depths that are typically encountered in practice. The mean value of Z was set at 19 baseline units, which means that if the camera moves 20 cm between frames, the average distance to any object in the scene would be 3.8 m. Once the value of Z is selected, ξ and α are chosen uniformly over the intervals $[0, D]$ and $[0, 2\pi)$, respectively, so that X and Y are computed as

$$X = Z\xi\cos\alpha, \text{ and}, Y = Z\xi\sin\alpha \tag{8.195}$$

Given the set of 3-D points, $\mathbf{p}_l$, the set of correspondences, $\mathbf{p}_r$, is then generated by applying the rigid body motion equation

$$\mathbf{p}_r = \mathbf{R}\mathbf{p}_l + \mathbf{b} \tag{8.196}$$

to each point in $\mathbf{p}_l$, for some value of $\mathbf{R}$ and $\mathbf{b}$. We thus obtain an error-free list of N pairs

$\{(\mathbf{r}_i, \boldsymbol{\ell}_i)\}$, where

$$\mathbf{r}_i = \frac{1}{Z_{r_i}}\mathbf{p}_{r\,i}, \text{ and, } \boldsymbol{\ell}_i = \frac{1}{Z_{l_i}}\mathbf{p}_{l\,i} \tag{8.197}$$

Measurement errors are then simulated by determining a vector $\delta\mathbf{r}_i$,

$$\delta\mathbf{r}_i = \begin{pmatrix} \rho_i \cos\beta_i \\ \rho_i \sin\beta_i \\ 0 \end{pmatrix} \tag{8.198}$$

as previously defined in equation (8.99).

The magnitude of the error, ρ_i, is specified in units of the focal length, f, so that it is independent of both the field of view and the spatial discretization of the sensor. To convert from units of f to pixels, one can multiply ρ_i by $2D/n$, where n is the number of pixels along one dimension of the sensor. For example, with a 20° field of view ($D = .364$) on a 256×256 pixel array, $\rho = .0028f$, corresponds to 1 pixel.

For systematic errors, the direction $\widehat{\delta\mathbf{r}}$ and the magnitude ρ are given as input to the simulation program. For these tests, $\widehat{\delta\mathbf{r}}$ was set equal to $\hat{y}$ for simplicity and a value was selected for ρ between 0 and $0.025f$. For uncorrelated random errors, each vector $\delta\mathbf{r}_i$ was determined independently by choosing β_i uniformly over the interval $[\,0, 2\pi)$ and ρ_i over the interval $[\,0, R_{max}\,]$, where R_{max} was supplied as an input parameter. Given that the vectors $\delta\mathbf{r}_i$ are uniformly distributed over a disk of area πR_{max}^2, we thus have

$$E[\delta\mathbf{r}_i^{\mathrm{T}}\,\delta\mathbf{r}_i] = E[\rho_i^2] = \frac{R_{max}^2}{2} = \sigma^2 \tag{8.199}$$

Values of R_{max} were chosen in the simulations to give σ between 0 and $0.02f$.

In order to compare the equations determined in Sections 8.2.1 and 8.2.2 for $\overline{\theta_b^2}$ and $\overline{\delta\theta^2}$, with the estimation errors actually computed for data corrupted by random error, several tests were performed for different types of motion and different values of D and N. The results of two series of tests, one with $\mathbf{b} \perp \hat{v}_3$ and the other with $\mathbf{b} \parallel \hat{v}_3$ are shown in Figures 8-4 and 8-5. In both series the rotation was given by $\hat{\omega} = (0,0,1)$ with $\theta = 5°$, so that $\hat{v}_3 = \hat{z}$, while the translation for the first series was specified as $\mathbf{b} = (1,0,0)$ and in the second as $\mathbf{b} = (0,0,1)$. The same set of 3-D coordinates $\mathbf{p}_l$, with $N = 50$, $\overline{Z} = 18.95$, and $\kappa = E[1/Z^2] = .0062$, was used for all tests. For each motion, actual and predicted errors for the translation and the rotation were computed for viewing fields of $\phi = 20°$, $\phi = 40°$,

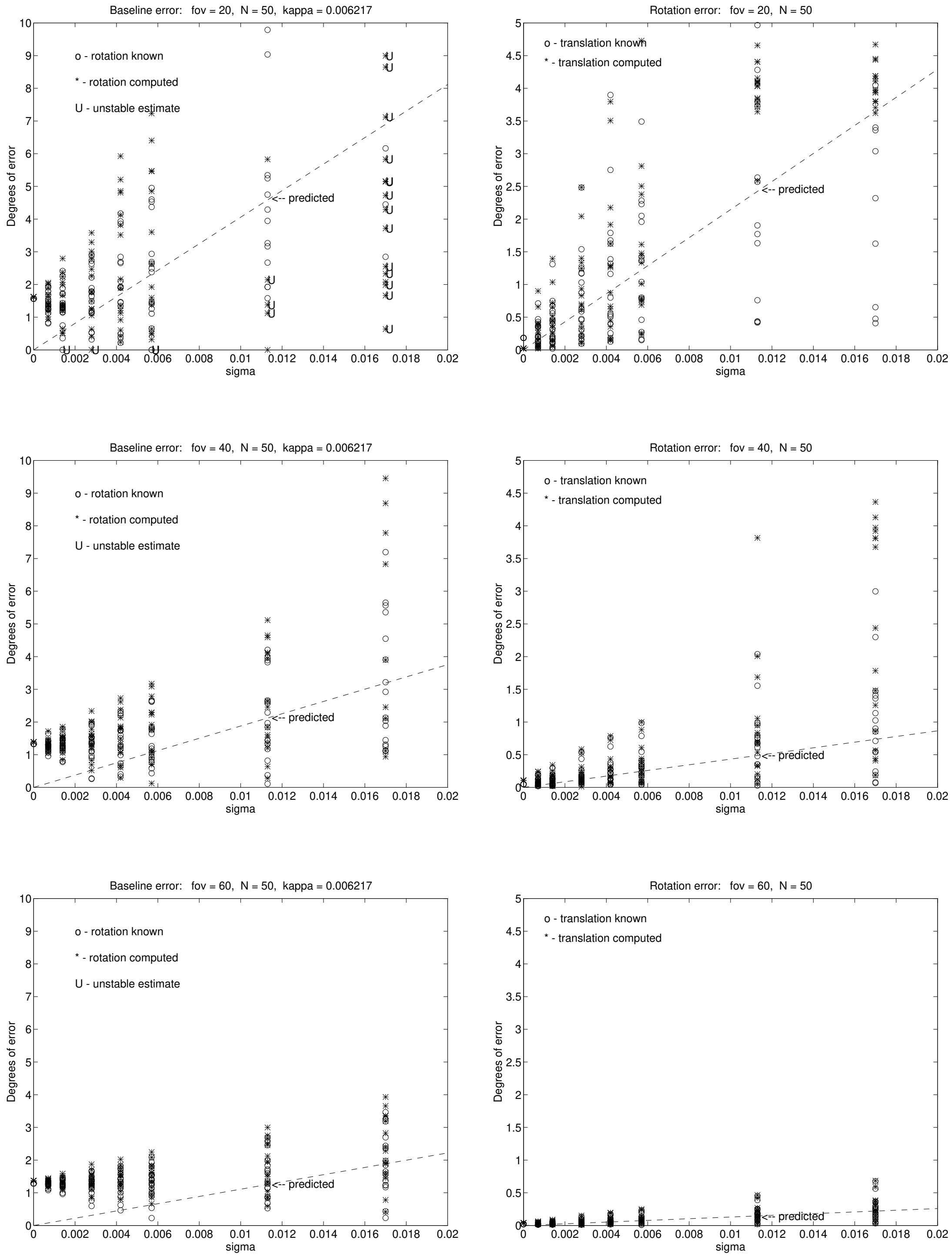


FIGURE 8-4: Estimation errors in translation and rotation with $\mathbf{b} \perp \hat{v}_3$ (uncorrelated measurement errors). $\mathbf{b} = (1, 0, 0)$, $\widehat{\omega} = (0, 0, 1)$, $\theta = 5^{\mathrm{o}}$

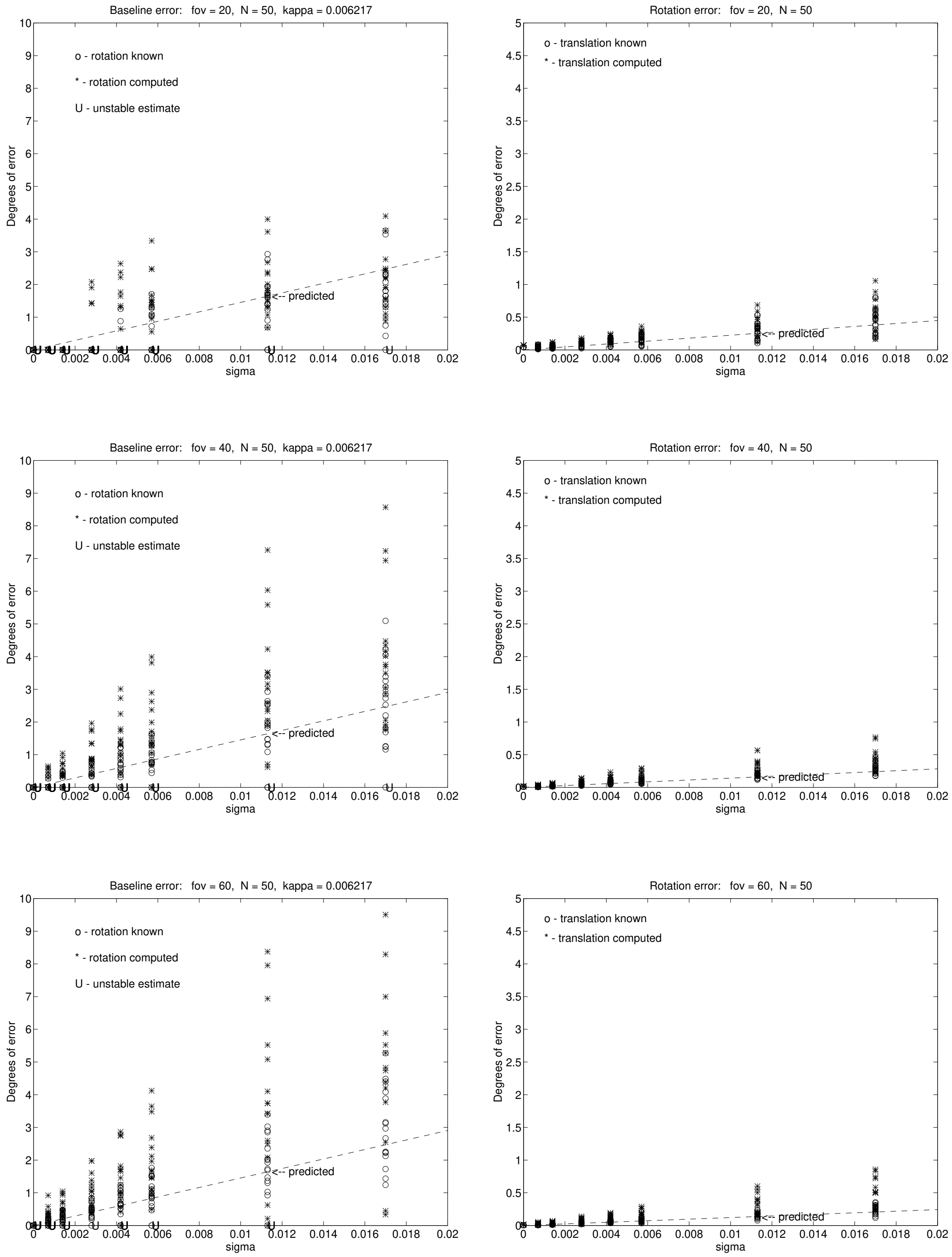


FIGURE 8-5: Estimation errors in translation and rotation with $\mathbf{b} \parallel \hat{v}_3$ (uncorrelated measurement errors). $\mathbf{b} = (0, 0, 1)$, $\widehat{\omega} = (0, 0, 1)$, $\theta = 5^o$

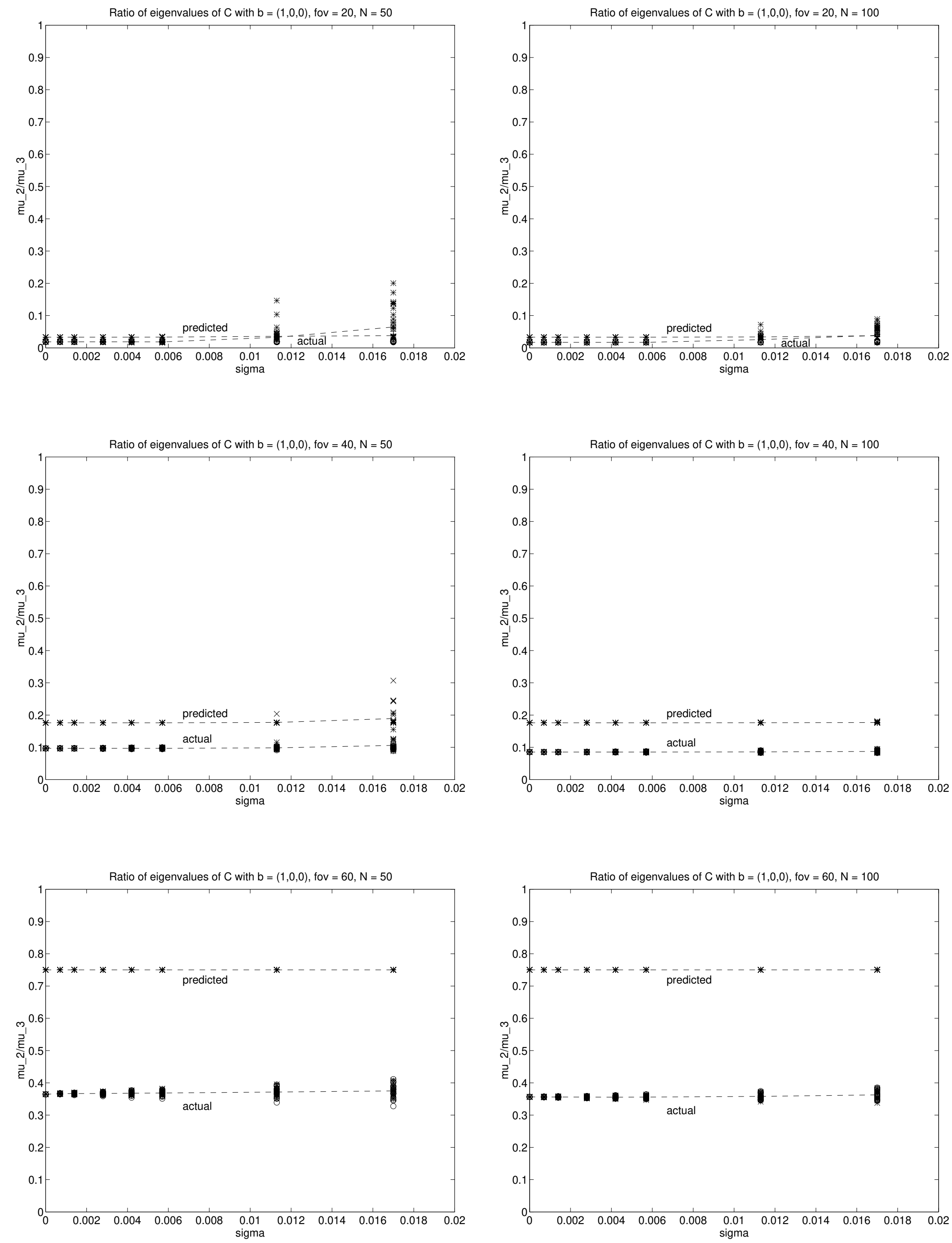


FIGURE 8-6: Actual and predicted values of μ_2/μ_3 for $\mathbf{b} \perp \hat{v}_3$, for $\phi = 20^o$, 40^o, and 60^o with $N = 50$ and $N = 100$.

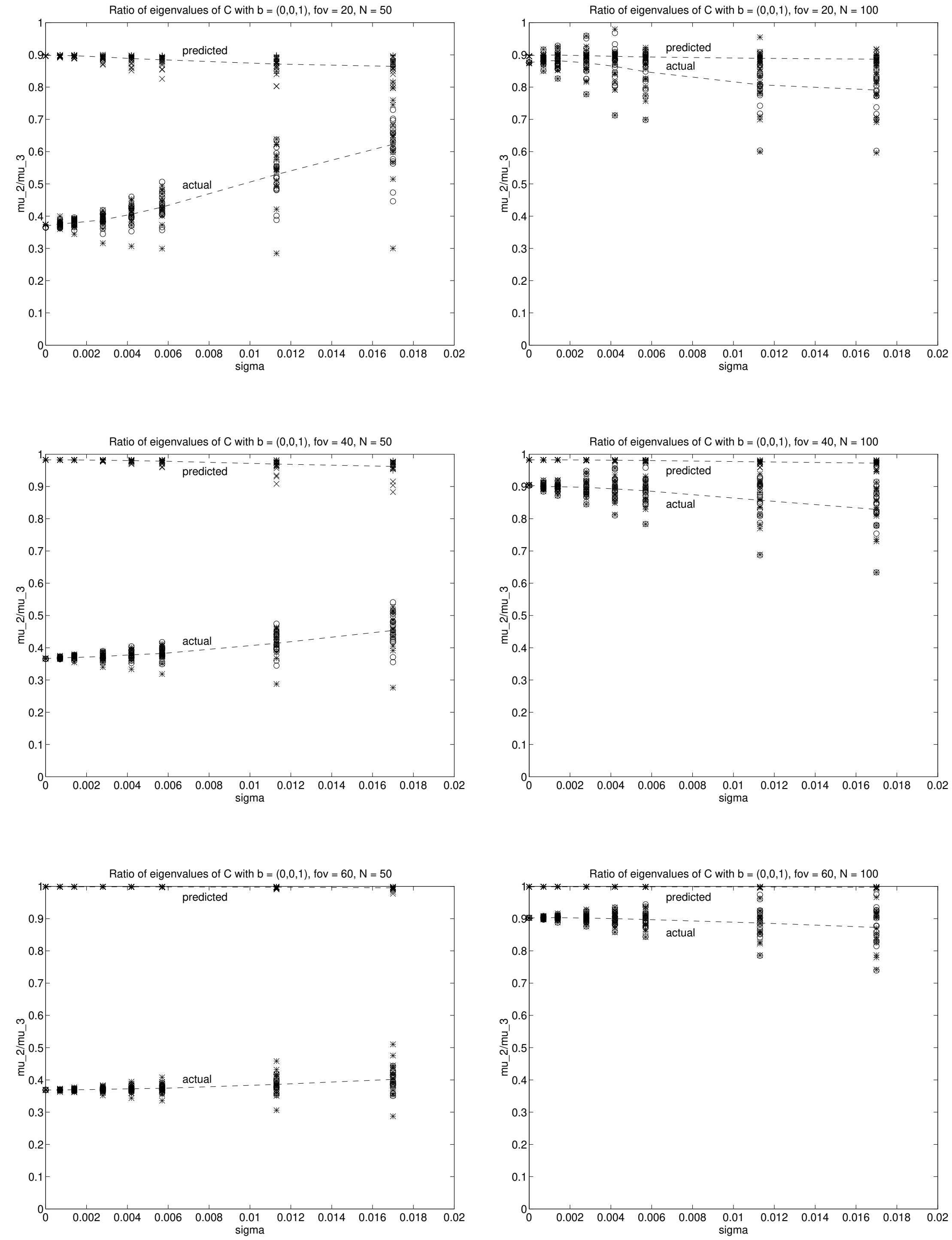


FIGURE 8-7: Actual and predicted values of μ_2/μ_3 for $\mathbf{b} \parallel \hat{v}_3$, for $\phi = 20^o$, 40^o, and 60^o with $N = 50$ and $N = 100$.

and $\phi = 60^{\circ}$ ($D = 0.364$, $D = 0.840$, and $D = 1.732$).

Since the equations for $\overline{\theta_b^2}$ and $\overline{\delta\theta^2}$ only predict expected values, a total of 16 different data sets were used for each test and each value of σ in order to obtain a statistically significant approximation of the average error. The data for each test were generated from the same set of ideal correspondences computed for the given motion by initializing the random number generator used to determine the $\delta\mathbf{r}_i$ with different seeds. The errors, shown in Figures 8-4 and 8-5, are plotted as circles when they were computed with either the rotation or the translation given, and as asterisks when the full motion was estimated. In many cases, particularly with $\phi = 20^{\circ}$ and $\mathbf{b} \perp \hat{v}_3$, the estimation of the translation was unstable in the sense that the eigenvector corresponding to the smallest eigenvalue of $\mathbf{C}$ was not the one which was closest to the true translation. In these cases, the error reported is the smallest angle between the true value of $\mathbf{b}$ and either of the other two eigenvectors, and is marked on the graph with the symbol 'U'.

The average of the computed errors agree well with the predicted values in most cases. There is a considerable spread in the results for both the baseline and rotation errors for motions with $\mathbf{b} \perp \hat{v}_3$, however, this should be expected from the higher sensitivity to error in this case. There is much less spread in the results when $\mathbf{b} \parallel \hat{v}_3$, and also many fewer instances of instability, although there does appear to be an increase in the spread of translation estimates at larger fields of view. On closer examination, however, it can be seen that the variation in this case is mostly in the errors from the full motion estimation, while errors in computing the translation with known rotation cluster well about the predicted average. The reason for the increased variation in the estimates from the complete algorithm, in the case of large viewing fields with $\mathbf{b} \parallel \hat{v}_3$, is not apparent from the first-order analysis of this chapter, although we can conjecture that it is caused by the nonlinear terms which were neglected.

The predictions of Sections 8.1.1 and 8.1.3 for the ratio of the middle and largest eigenvalues of $\mathbf{C}$ were also tested for these simulated motions. In Figures 8-6 and 8-7 the ratios predicted from the estimated $\mathbf{b}$ and $\hat{v}_3$, as well as the actual values computed from the matrix $\mathbf{C}$ itself, are shown for each value of σ and ϕ. As before, each test was performed 16 times with different sets of randomized errors added to the correspondence data. The dashed lines indicate the best-fitting curves to the results from the 16 tests plotted as a function of σ. It should be noted the results from all of the tests are shown, including those for which the translation estimate was unstable. The actual ratios are always computed from the middle and largest eigenvalues of $\mathbf{C}$. However, the predicted ratio is computed using the eigenvector closest to the true baseline as the estimate of the translation.

The first observation which can be made based on these tests is that there is a difference between the predicted and actual values for μ_2/μ_3 when $\mathbf{b} \perp \hat{v}_3$ which increases with the field of view. This effect can be attributed to the fact that the value of $D = \tan\phi$ was used to compute the predicted ratio, while it probably would have been better to use $D/\sqrt{2}$, which is the average distance of points from the center of the image, and which is a more appropriate statistical measure of the radius of the viewing field. With $\mathbf{b}$ exactly perpendicular to $\hat{v}_3$ the predicted ratio from equation (8.23) is $\mu_2/\mu_3 = D^2/4$. However the actual ratios agree much better with $D^2/8$ which for $\phi = 20^o$, 40^o, and 60^o, would give predicted ratios of .016, .09, and .375, respectively,

Since equations (8.20) and (8.21) for the eigenvalues of $\mathbf{C}$ were derived by modeling the discrete correspondences as a uniform density spread over the field of view, tests were performed with both $N = 50$ and $N = 100$ to assess how strongly the eigenvalue ratios are affected by this approximation, which clearly depends on N. With $\mathbf{b} \perp \hat{v}_3$ there is not a significant difference in the results for the values of N tested, however, there is a large difference for $\mathbf{b} \parallel \hat{v}_3$. This difference can be explained by noting that the ratio μ_2/μ_3 is a measure of the symmetry in the distribution of the vectors $\mathbf{c}_i = \boldsymbol{\ell}'_i \times \mathbf{r}_i$ in the plane perpendicular to $\mathbf{b}$. The equality of the two largest eigenvalues in the case $\mathbf{b} \parallel \hat{v}_3$ predicted in equation (8.25) is thus a direct consequence of the uniform density assumption. Nonetheless, the values of μ_2/μ_3 are seen to be consistently higher when $\mathbf{b} \parallel \hat{v}_3$ than when $\mathbf{b} \perp \hat{v}_3$, except in the case of $\phi = 60^o$ with $N = 50$, where the values are similar. It can also be seen that the difference in the ratios is largely unaffected either by errors in the data or by the fac that the algorithm sometimes converged to the wrong solution.

It can thus be concluded that the ratio test is an effective indicator for discriminate between correct and false solutions. However, in order to implement the test on a real system, it is necessary to first develop a baseline profile of the expected ratios for translations parallel and perpendicular to the image plane, rather than predicting their values from equations (8.20) and (8.21), since these will depend on the actual geometry of the sensor and focal length of the lens, as well as on the average number of correspondence points returned by the matching procedure.

Finally, the case of systematic measurement errors was investigated for $\mathbf{b} \perp \hat{v}_3$ and $\mathbf{b} \parallel \hat{v}_3$ with the results shown in Figures 8-8 and 8-9 for $\phi = 20^o$, 40^o, and 60^o. Since

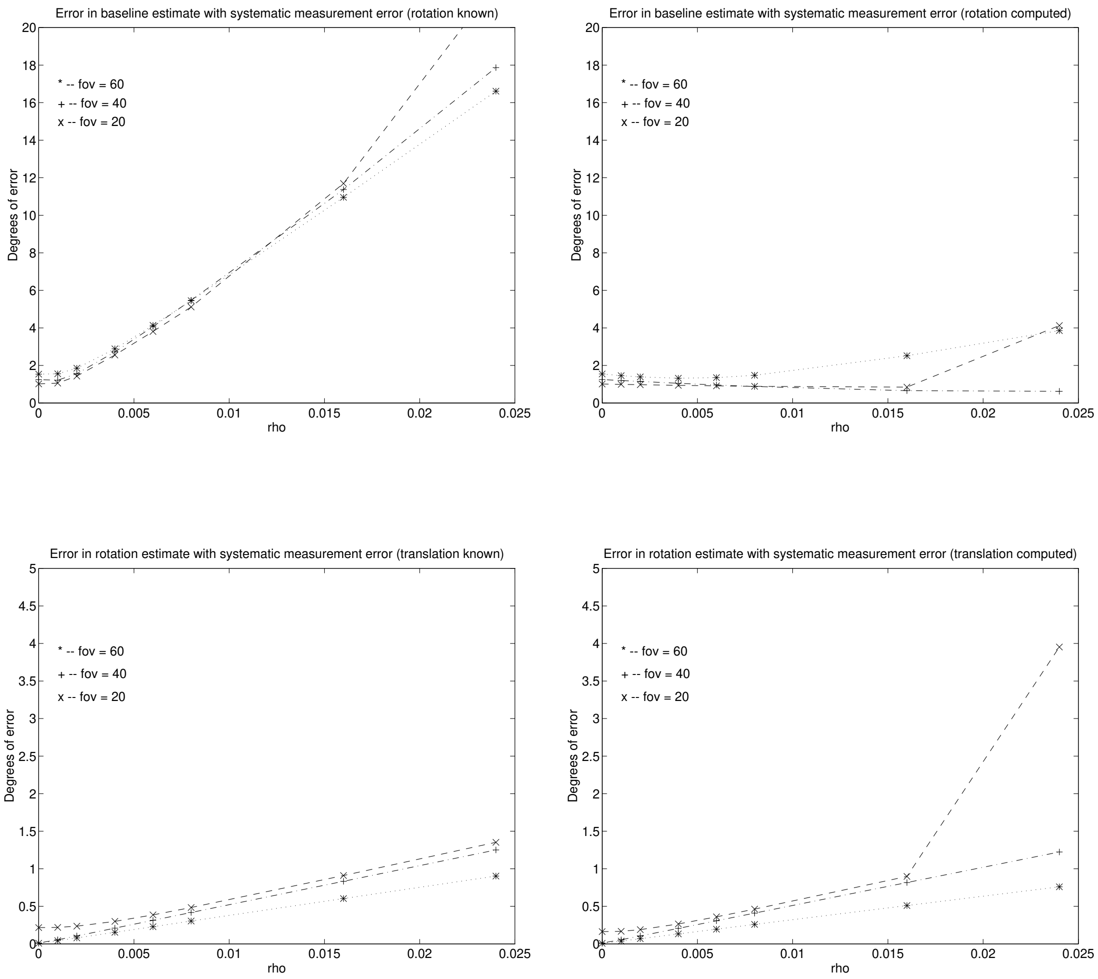


FIGURE 8-8: Errors in estimates of translation and rotation with $\mathbf{b} \perp \hat{v}_3$ (Systematic measurement errors). $\mathbf{b} = (1, 0, 0)$, $\widehat{\omega} = (0, 0, 1)$, $\theta = 5^{\mathrm{o}}$.

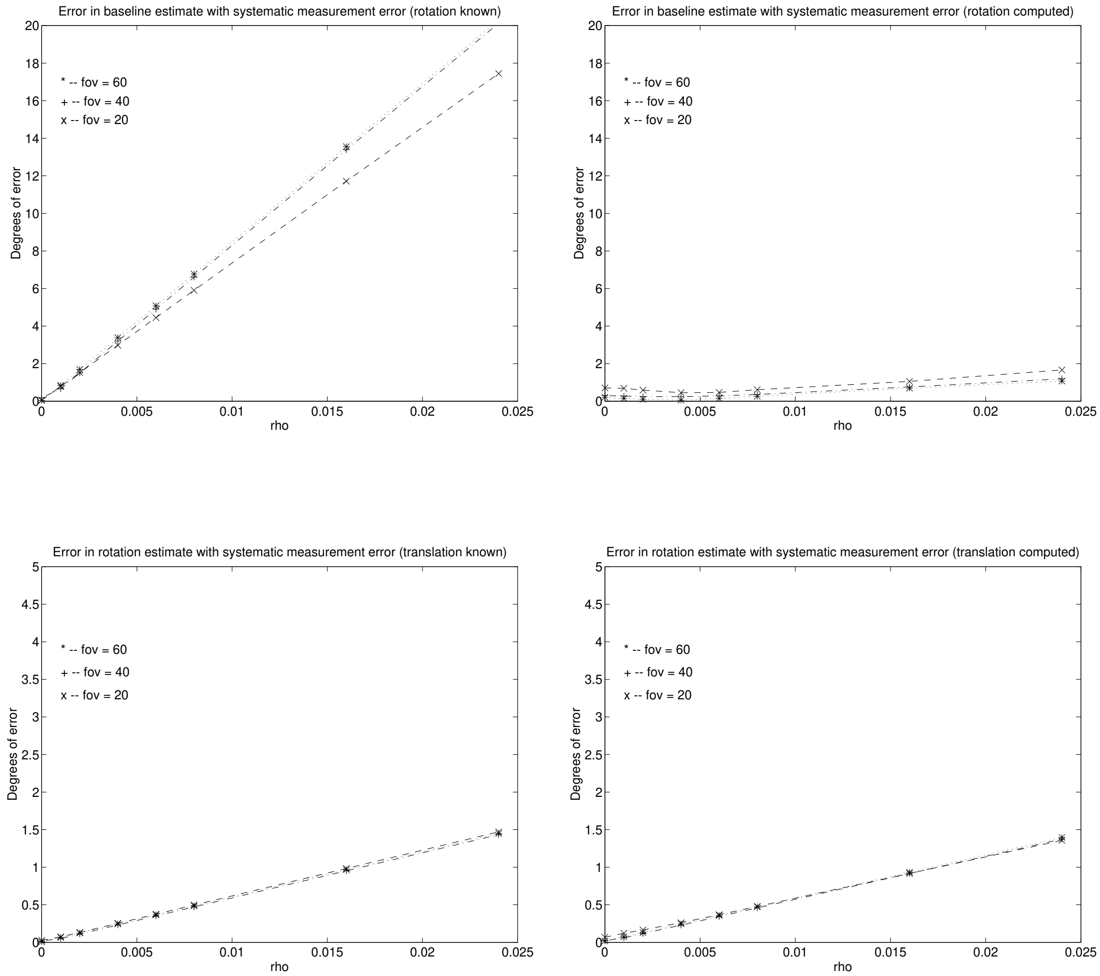


FIGURE 8-9: Errors in estimates of translation and rotation with $\mathbf{b} \parallel \hat{v}_3$ (Systematic measurement errors). $\mathbf{b} = (0, 0, 1)$, $\widehat{\omega} = (0, 0, 1)$, $\theta = 5^o$.

the errors are deterministic, it was not necessary to make numerous tests, as in the case of random data, and so the results of only one simulation are shown for each motion and field of view.

It can be seen that the actual errors agree well with those predicted by equations (8.136), (8.171) and (8.174). For translation with known rotation, the errors are essentially independent of the field of view for both $\mathbf{b} \perp \hat{v}_3$ and $\mathbf{b} \parallel \hat{v}_3$, although there is a slight difference in both cases for $\phi = 20^o$. For the estimates of rotation with known translation, the errors do increase as ϕ decreases when $\mathbf{b} \perp \hat{v}_3$, but are independent of ϕ when $\mathbf{b} \parallel \hat{v}_3$.

When the full motion is computed, the errors in the rotation scarcely change from those computed with the translation known. Errors in the translation estimates, however, are reduced drastically. With $\mathbf{b} \perp \hat{v}_3$, the error is removed almost entirely at $\phi = 20^o$ and is only slightly worse at larger fields of view. With $\mathbf{b} \parallel \hat{v}_3$, on the other hand, the error is essentially removed at all values of ϕ.

As previously noted, the fact that one can obtain very accurate estimates of the translation from the complete algorithm when the measurement errors are correlated, is significant as it implies that accurate internal camera calibration is not required to obtain useful information for many applications, such as passive navigation, that do not require highly precise estimates of the rotation.

Chapter 9

Summary of Part I

Many issues have been covered in the preceding chapters, and it is useful to summarize the major results and conclusions.

In the architecture outlined in Chapter 4, it was determinined that processing in the motion system should be divided into three stages:

- Edge detection,
- Feature matching by block correlation of the binary edge maps, and
- Solving the motion equations.

Edge detection would be performed directly on the analog signals acquired by the photosensors using a fully parallel analog array processor implementing the multi-scale veto algorithm. In Chapter 5, this edge detection algorithm was shown to have the following advantages over classical methods:

- There is no tradeoff between edge localization and smoothing. Noise and unwanted minor features can be effectively removed by adjusting the sequence of thresholds applied at each smoothing cycle. The edges which are detected, however, remain localized at the positions of the features in the original image regardless of the thresholds used.
- The method does not require computing second differences or searching for zero-crossings. Hence the circuitry is much simpler, and all processing is local.

- The algorithm is designed to take advantage of the signal processing capabilities of CCDs. It can thus be efficiently implemented on a CCD array with circuitry placed at each pixel to compute differences and store edge signals.

At the end of Chapter 3, it was concluded that the most appropriate method for obtaining the correspondence points needed by the motion algorithm was a block-correlation procedure using the binary edge maps produced in the first processing stage. Due to the presence of repeating patterns which occur naturally in real scenes, however, similarity measures alone cannot determine the best matches and achieve an acceptably low false-alarm rate. In Chapter 6, a series of tests was developed to add to the correlation procedure in order to minimize the error rate. These tests included rejecting matches from blocks which have too few or two many edge pixels to give a low probability of a false match, and rejecting altogether blocks for which there were multiple possible matches.

Chapter 7 covered the development of the algorithm to estimate motion from the set of point matches found in stage 2. Due to the complexity and nonlinearity of the equations involved, it was determined that the motion algorithm should be executed on a standard digital processor. Nonetheless, given the goal of building a low-power system, it was necessary to simplify the algorithm as much as possible so that minimal processing power would be required. It was shown that by alternating the procedures for updating the baseline and the rotation, the complexity of the operations could be considerably reduced such that the most complex computation would be to solve a 3×3 eigenvalue-eigenvector equation at each iteration. Simulations of the algorithm on several image sequences using the point correspondences determined by the edge detection algorithm and the matching procedure showed that these simple methods developed for efficient implementation in VLSI were as effective for obtaining accurate estimates of the motion as more complex procedures commonly implemented in software.

Finally, we could not build a robust system for computing motion without studying the effects of measurement errors on the estimates, and how these effects are related to the type of motion and scene structure, as well as to the spatial resolution of the image sensor, the field of view, and the number of correspondence points found. In Chapter 8, the numerical stability of the motion algorithm was thoroughly analyzed and expressions were derived for the expected estimation error in the cases of both random and systematic measurement errors. Three important results were obtained from this analysis:

- A test was developed for reliably determining when the motion algorithm converges to an incorrect solution based on the ratio of the two largest eigenvalues of the matrix $\mathbf{C}$, defined in equation (7.6).

- Design guidelines were developed for building a system with the appropriate sensor resolution, field of view, and number of matching circuits to achieve a given maximum expected estimation error.

- It was also discovered that precise internal camera calibration was not required to obtain accurate estimates of the translation, provided that the rotation is estimated as well. This significant result implies that applications such as navigation which require accurate knowledge of the translation direction, but which are less sensitive to errors in the rotation, can implement the motion system without also needing sophisticated calibration procedures.

Part II

Design of a CCD-CMOS Multi-Scale Veto Chip

Chapter 10

Basic Requirements

The plan for the design of the multi-scale veto (MSV) edge detector was outlined in Chapter 5. A two-dimensional CCD array as shown in Figure 5-3 is ideally suited for performing the successive smoothing operations required by the algorithm. The remaining tasks of computing the differences between the smoothed brightness values at neighboring pixels and testing if the magnitude of these differences is above a given threshold, is then performed by additional circuitry placed between each pair of nodes within the array.

Several considerations were important in determining the design of the different elements in the MSV edge detection processor, of which one of the major concerns was the silicon area required for each pixel. Since the number of pixels in the image array directly impacts the robustness of the motion estimates which can be derived from the edge maps, it is important that each cell in the array be as small as possible. One of the ways to reduce the per-pixel area, which is already incorporated in the design, is by using time as a dimension. Since the threshold tests are performed sequentially at the end of each smoothing cycle, only one difference and test circuit is required for each node pair. However, this also means that space is needed to store the intermediate results and that the internal circuits must be fast enough to complete all of the tests within the time allotted for processing the image.

Small area implies simple circuits. However, if the edge detector is to produce useful results, the need for simplicity cannot compromise the resolution requirements of the algorithm. The attenuation factors for several idealized image features were given in Table 5.1 as a function of the number of smoothing cycles performed. If equation (5.8) is used to compute the thresholds τ_k for each smoothing cycle $k = 0, \ldots, n$, the internal difference and test circuits must be able to resolve differences as large as τ_0, the initial threshold, and as small as $\tau_n = G_{n,f}\tau_0$, where $G_{n,f}$ is the attenuation of the chosen model feature after n smoothing cycles.

To translate this requirement into a percentage of the full scale range (FSR), we can take a specific example with $n = 5$ using the horizontal step edge as the model feature. In grayscale images with normal contrast, τ_0 is usually set at around 10% of FSR, and from Table 5.1, the attenuation factor for the horizontal step edge after 5 cycles is seen to be $G_5 = 0.246$. The range of distinguishable differences must thus be between 10% and 2.5% of FSR, or in terms of bits of precision, between 4 and 5 bits. If either the diagonal step edge or the horizontal 2-pixel line is used as the model, the resolution requirement jumps to between 4 and 6 bits.

Designing a small absolute-value-of-difference circuit with this much resolution has been one of the major challenges in building a working MSV chip. In the smoothing and segmentation chip designed by Keast [86], a CCD-based absolute value of difference circuit was used, primarily because of its small size with respect to a transistor-based design. A similar structure was also included in the stereo disparity chip designed by Hakkarainen [65]. Unfortunately, for reasons which Keast discovered and which will be discussed in the next chapter, the CCD circuit has a 'dead zone' for small differences which limits its resolution to less than 25% of FSR. It was thus necessary to design a new transistor-based absolute-value-of-difference circuit occupying the least area possible.

Given the above constraints, a 32×32 prototype array implementing the MSV algorithm was designed and fabricated through MOSIS using the Orbit 2μm CCD/CMOS process. The next three chapters are devoted to discussing the design and testing of this array and to analyzing the changes required to build a full-size (~256×256) processor, possibly operating as an image sensor as well as an edge detector. In order to clarify aspects of the design involving CCDs, the following chapter describes the basic physics of charge storage and transfer and the input and output structures used to interface with CCD arrays. Chapter 12 covers in detail the design of each of the major components in the MSV array, and finally, Chapter 13 describes the test system and results.

Chapter 11

Charge Coupled Device Fundamentals

Charge coupled devices are based on the principle that a charge packet may be confined within a potential well created by applying a voltage to a polysilicon gate and may be moved from one location to another by appropriately manipulating the gate voltages. Conceptually, it is useful to think of CCDs as buckets and of the signal charge as water. The process of charge transfer is similar to that of moving the water by placing the buckets on risers, as shown in Figure 11-1, with tubes connecting the bases of neighboring buckets. The water levels in adjacent buckets are determined by the relative heights of the risers, just as charge levels under adjacent gates of a CCD are determined by the relative difference in the well potentials.

The physics of CCD operation is of course considerably different from that of the bucket brigade. CCDs exist in two forms: surface channel and buried channel devices. Both operate on the same principle of charge transfer; however, their physical structure and device characteristcs are very different. It is useful to examine both structures in order to understand the advantages and limitations of each.

11.1 Surface Channel Devices

The simplest form of CCD, which is the surface-channel device, is constructed from a series of adjacent MOS capacitors operating in deep-depletion mode. Figure 11-2 shows the typical structure of a MOS capacitor formed by sandwiching a thin layer of oxide, SiO_2, between a polysilicon gate and a p-type semiconductor substrate. When a voltage greater

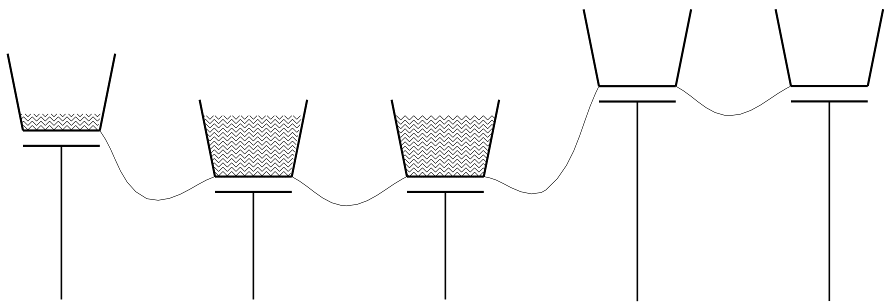


FIGURE 11-1: 'Bucket brigade' analogy of charge coupled devices.

than the flatband voltage V_{fb} is applied to the gate, the substrate is depleted of majority carriers and a depletion layer is formed as shown in Figure 11-3a. If the gate voltage is raised above the threshold voltage V_t of the material, defined as the point at which strong inversion occurs [92], minority electrons are attracted to the Si–SiO_2 interface and the depletion region ceases to increase in depth (Figure 11-3b).

In order for the inversion channel to form, there must be a supply of available electrons. In an NMOS transistor, electrons in the conducting channel are supplied by the metal contact made to the source diffusion. In the MOS capacitor, the electrons must come from the substrate where they are produced by thermal generation of electron-hole pairs. Thermal generation in the bulk results in a flow of electrons from the substrate to the high potential region at the surface, known as *dark current*, which continues until equilibrium conditions are obtained. In a well-designed process, however, the dark current density, J_D, is typically $\leq 1nA/cm^2$. At this level, the time required for the device to reach equilibrium is on the order of minutes [86].

CCDs exploit this long equilibration time to perform useful signal processing tasks. When V_g is raised above V_t, the depletion region initially extends beyond its maximum equilibrium depth, as shown in Figure 11-3c. This is the condition known as *deep depletion.* Signal charge may be introduced into the device either optically or electrically and will be confined to the potential well until its maximum capacity is reached. The maximum charge which can be held is equal to the channel charge of the capacitor at equilibrium and is a linear function of the applied gate voltage. By placing gates connected to independent voltages adjacent to one another, the signal charge may be transferred between gates just as the water is moved in the bucket brigade.

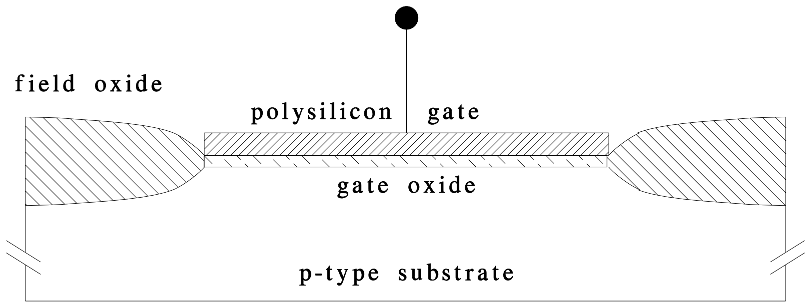


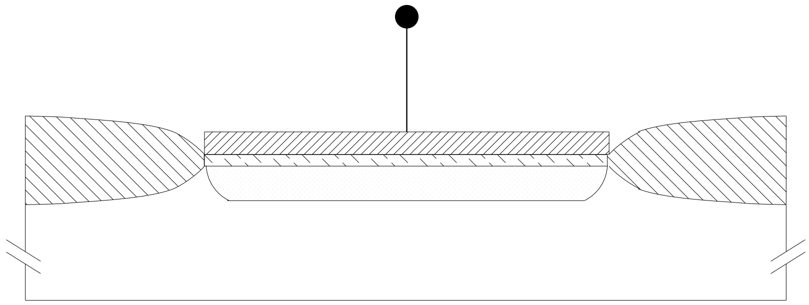
FIGURE 11-2: The MOS capacitor.

The primary advantage of surface channel devices is the linear relation between V_g and Q_{max}, the maximum signal charge. These devices suffer, however, from poor transfer efficiency due to the high number of interface states at the Si–SiO_2 boundary which can trap charge and then release it at a random time afterward. Trapping results in both noise and signal degradation when the charge must be transferred through a long series of gates. Consequently, surface channel devices are never used in the design of large high-quality sensors. Their chief application is in performing operations where the linearity of signal charge with gate voltage is important and where only a few gates are needed.

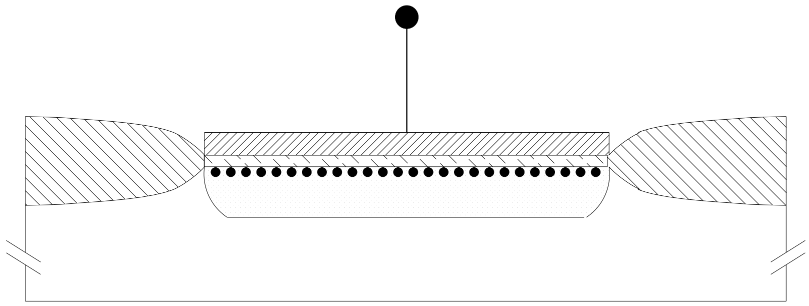
11.2 Buried Channel Devices (BCCDs)

Like the surface channel device, the buried channel CCD is also a non-equilibrium structure. In the BCCD, however, the signal charge is held away from the Si–SiO_2 interface because the potential maximum occurs inside the channel, several hundred nanometers below the surface. The buried channel is created by adding an n-doped implant below the transfer gates. Electrical contact is made to the channel at $n+$ diffusions placed at the extremities of the gate array, and when a sufficiently large positive voltage is applied with respect to the substrate, the buried layer is completely depleted of majority carriers.

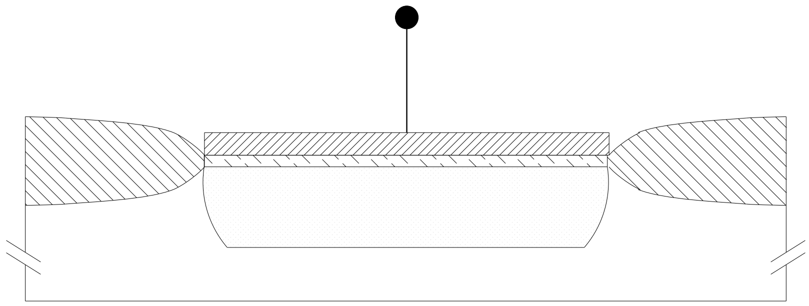
The potential profile, $\phi(x)$, with depth in the semiconductor typically resembles the curves shown in Figure 11-4 [93], with the dotted and solid lines representing the profiles with and without signal charge, respectively. Here x represents depth below the oxide layer, with $x = 0$ at the Si–SiO_2 interface. As seen in the diagram, the addition of the buried layer



a.) Depletion, $V_t > V_g > V_{fb}$



b.) Strong inversion (equilibrium), $V_g > V_t$



c.) Deep depletion (non-equilibrium), $V_g > V_t$

FIGURE 11-3: States of the MOS capacitor.

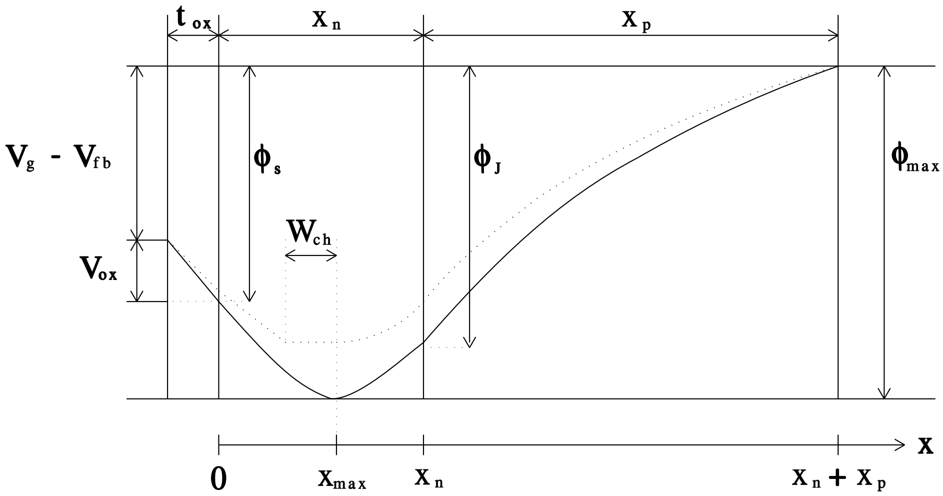


FIGURE 11-4: Potential profile with depth of a buried channel CCD

creates a non-monotonic profile such that the maximum potential ϕ_{max} resides at a distance x_{max} below the interface. If electrons are injected into the layer, they will accumulate near x_{max} rather than at the surface.

The BCCD structure may be modeled as a series connection of lumped capacitances, as illustrated in Figure 11-5. C_{ox} represents the oxide capacitance, ϵ_{ox}/t_{ox}, while C_{d1} and C_{d2} represent the depletion capacitances between the channel and the oxide and the channel and the substrate, respectively. The signal charge occupies a finite width, W_{ch}, which cannot be neglected in computing the values of C_{d1} and C_{d2}. Conventionally, this width is divided equally between the two depletion capacitances [92], resulting in the following expressions:

$$C_{d1} = \frac{\epsilon_{Si}}{(x_{max} - W_{ch}/2)} \tag{11.1}$$

and

$$C_{d2} = \frac{\epsilon_{Si}}{(x_{ch-b} + W_{ch}/2)} \tag{11.2}$$

with

$$x_{ch-b} = x_p + x_n - x_{max} \tag{11.3}$$

The effective capacitance C_{eff} between the signal charge and the gate is the series combination of C_{ox} and C_{d1}.

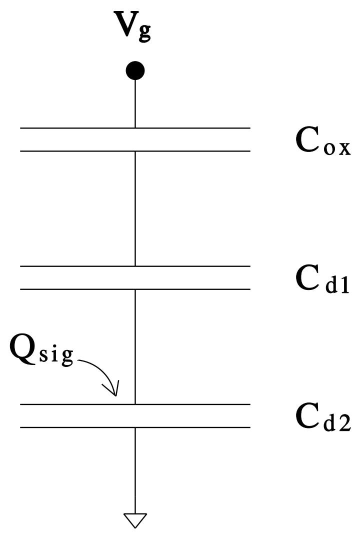


FIGURE 11-5: Lumped capacitance model of a BCCD

$$\frac{1}{C_{eff}} = \frac{1}{C_{ox}} + \frac{1}{C_{d1}} \tag{11.4}$$

We can immediately see two effects of the buried channel. The first is that the signal carrying capacity is less than that of a surface channel device due to the decrease in the effective capacitance caused by moving the charge away from the surface. Second, the effective capacitance depends nonlinearly on both the amount of charge present and, as will be seen shortly, on the gate voltage.

Nonetheless, buried channel devices offer significant advantages in charge transfer efficiency. By keeping the signal charge away from the Si–SiO_2 surface, the interaction of the charge packet with traps at the interface is essentially eliminated. Although bulk traps also occur, they are much less frequent than those at the interface, and from a process standpoint, it is much easier to reduce the bulk state density than that of interface states [92]. Furthermore, the reduced effective capacitance between the signal charge and the gate increases the fringing fields, which are the dominant driving force in the final stages of charge transfer, as will be seen in Section 11.4. BCCDs are thus the structure of choice for large array sensors.

In order to use a buried channel CCD for signal processing, we must first understand the relationships between gate voltage V_g, signal charge Q_{sig} $(coulombs/cm^2)$, and the channel potential $\phi(x)$. From this we can compute the magnitude of the lateral potential barrier between two adjacent gates held at different voltages, and hence the maximum signal charge per unit area which can be confined. This information is necessary in order to determine the design parameters for the transfer gates, as well as for the charge input and output structures.

Under the depletion approximation, the charge density profile of a buried channel device at gate voltage V_g containing N_e *electrons/μm^2* is as shown in Figure 11-6 [93]. We use N_D to denote the doping concentration $(donors/\mu m^3)$ of the buried channel and N_A $(acceptors/\mu m^3)$ to denote that of the p-type substrate. The signal charge distributes itself over a finite width $W_{ch} = N_e/N_D$ due to the attraction of the negatively-charged electrons to the positively-charged fixed donor ions in the lattice. The depth of the charge packet, which ends at $x = x_{max}$, is limited by the depletion region created by the junction between the n-type implant and the p-type substrate. The widths of the space charge regions on either side of the junction are related by the charge balance equation

$$N_A x_p = N_D(x_n - x_{max}) \tag{11.5}$$

where x_p represents the extent of the depletion region into the substrate.

The potential profile $\phi(x)$ is obtained by integrating Poisson's equation

$$\frac{d^2\phi}{dx^2} = -\frac{\rho}{\epsilon_{Si}} \tag{11.6}$$

in each region of constant charge density shown in Figure 11-6. Within these four regions, Poisson's equation is given by

$$\begin{aligned}
\frac{d^2\phi}{dx^2} &= -\frac{qN_D}{\epsilon_{Si}}, && 0 \le x \le x_{max} - W_{ch} && (11.7)\\
\frac{d^2\phi}{dx^2} &= 0, && x_{max} - W_{ch} \le x \le x_{max} && (11.8)\\
\frac{d^2\phi}{dx^2} &= -\frac{qN_D}{\epsilon_{Si}}, && x_{max} \le x \le x_n && (11.9)\\
\frac{d^2\phi}{dx^2} &= +\frac{qN_A}{\epsilon_{Si}}, && x_n \le x \le x_n + x_p && (11.10)
\end{aligned}$$

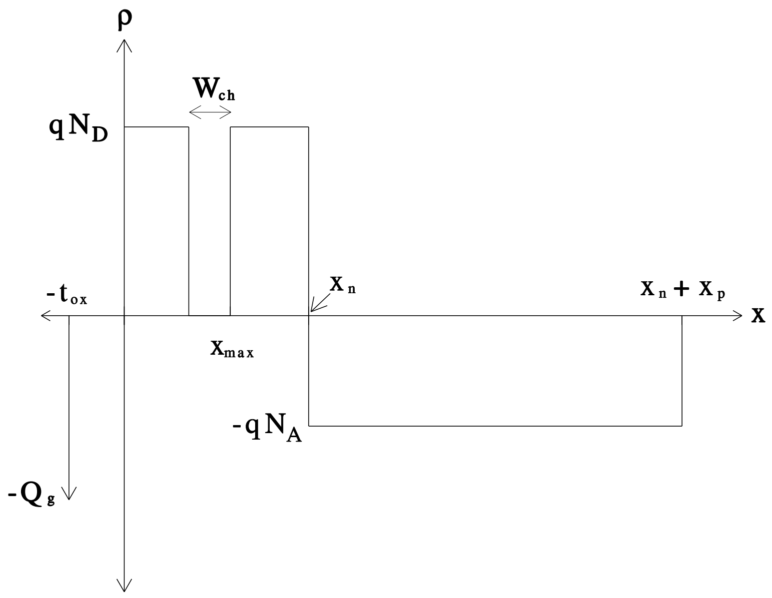


FIGURE 11-6: Charge density profile of a buried channel CCD with signal charge Q_{sig} (using depletion approximation)

The constants of integration are determined by the conditions of continuity of $\phi(x)$ at the region boundaries and the continuity of the electric displacement field at $x = 0$ and $x = x_n + x_p$. These conditions are stated as:

$$\phi(0) = \phi_s \tag{11.11}$$

$$\phi(x_{max}) = \phi_{max} \tag{11.12}$$

$$\phi(x_n + x_p) = 0 \tag{11.13}$$

$$-\epsilon_{Si} \left.\frac{d\phi}{dx}\right|_{x=0} = \epsilon_{ox} E_{ox} \tag{11.14}$$

$$-\epsilon_{Si} \left.\frac{d\phi}{dx}\right|_{x_n+x_p} = 0 \tag{11.15}$$

Referring to Figure 11-4, we see that

$$\phi_s = V_g - V_{fb} + V_{ox} \tag{11.16}$$

and since $V_{ox} = \frac{1}{C_{ox}}$(fixed charge density + mobile charge density) [93], it follows that

$$\begin{aligned} V_{ox} &= \frac{q}{C_{ox}}(N_D x_{max} - N_e) \\ &= \frac{q t_{ox}}{\epsilon_{ox}} N_D (x_{max} - W_{ch}) \end{aligned} \quad (11.17)$$

The electric field across the oxide is given by

$$E_{ox} = -\frac{V_{ox}}{t_{ox}} = -\frac{q}{\epsilon_{ox}} N_D (x_{max} - W_{ch}) \quad (11.18)$$

Direct integration of equations (11.7)–(11.10) applying the constraints (11.13) and (11.14) results in

$$\phi(x) = -\frac{qN_D}{2\epsilon_{Si}}(x - (x_{max} - W_{ch}))^2 + \phi_{max}, \qquad 0 \le x \le x_{max} - W_{ch} \quad (11.19)$$

$$\phi(x) = \phi_{max}, \qquad x_{max} - W_{ch} \le x \le x_{max} \quad (11.20)$$

$$\phi(x) = -\frac{qN_D}{2\epsilon_{Si}}(x - x_{max})^2 + \phi_{max}, \qquad x_{max} \le x \le x_n \quad (11.21)$$

$$\phi(x) = \frac{qN_A}{2\epsilon_{Si}}(x - x_n - x_p)^2, \qquad x_n \le x \le x_n + x_p \quad (11.22)$$

From equations (11.11), (11.16), (11.17), and (11.19) we can obtain a first equation for ϕ_{max}:

$$\phi_{max} = V_g - V_{fb} + \frac{q t_{ox}}{\epsilon_{ox}} N_D (x_{max} - W_{ch}) + \frac{qN_D}{2\epsilon_{Si}}(x_{max} - W_{ch})^2 \quad (11.23)$$

while a second equation may be found by equating the potential across the n-p junction at $x = x_n$ and combining equations (11.5), (11.21), and (11.22):

$$\phi_{max} = \frac{qN_D}{2\epsilon_{Si}}(x_n - x_{max})^2 \left(1 + \frac{N_D}{N_A}\right) \quad (11.24)$$

Equating the righthand sides of equations (11.23) and (11.24) we obtain a quadratic equation which can be solved for x_{max}. Given x_{max} we can find ϕ_{max}, and therefore $\phi(x)$, everywhere within the silicon.

The maximum number of electrons per unit gate area which can be held in the channel is given by:

$$N_{e,max} = N_D x_{max} \tag{11.25}$$

Of more interest, however, is the maximum number which may be confined under one gate that is at a higher voltage than its neighboring gates. Charge is trapped as long as the channel potential under one gate is higher than that of its neighbors. As seen from equation (11.23), ϕ_{max} is a decreasing function of $W_{ch} = N_e/N_D$. The maximum value of N_e is therefore the one for which ϕ_{max} is equal to the channel potential under the neighboring gate.

Let V_{g1} denote the voltage on the neighboring gate, which has zero signal charge, ($W_{ch} = 0$), and let V_{g2} denote the voltage on the gate containing charge $-qN_{e,max}$. From equation (11.23), we obtain

$$\phi_{max1} = V_{g1} - V_{fb} + \frac{qt_{ox}}{\epsilon_{ox}} N_D x_{max_1} + \frac{qN_D}{2\epsilon_{Si}} x^2_{max_1} \tag{11.26}$$

and

$$\phi_{max2} = V_{g2} - V_{fb} + \frac{qt_{ox}}{\epsilon_{ox}} N_D \left(x_{max2} - \frac{N_{e,max}}{N_D} \right) + \frac{qN_D}{2\epsilon_{Si}} \left(x_{max2} - \frac{N_{e,max}}{N_D} \right)^2 \tag{11.27}$$

Equating the righthand sides of the above expressions and noting that equal ϕ_{max} implies equal x_{max}, as seen from equation (11.24), we obtain

$$\begin{aligned} V_{g2} - V_{g1} &= qN_{e,max} \left(\frac{t_{ox}}{\epsilon_{ox}} + \frac{1}{\epsilon_{Si}} \left(x_{max} - \frac{W_{ch}}{2} \right) \right) \\ &= qN_{e,max} \left(\frac{1}{C_{ox}} + \frac{1}{C_{d1}} \right) \\ &= \frac{qN_{e,max}}{C_{eff}} \end{aligned} \tag{11.28}$$

Although it appears from this equation that $N_{e,max}$ is linearly related to the difference in gate voltages, it should be remembered that the depletion capacitance C_{d1} depends both on the size of the signal charge packet and on the value of V_{g1} through its influence on x_{max}.

11.3 Charge Transfer and Clocking

The primary usefulness of CCDs is derived from their ability to move signal charge from one location to another. Raising the voltage on one gate and lowering that on a neighboring gate shifts the position of the potential well and also moves any charge contained in it. By appropriately sequencing the gate voltages, it is not only possible to transfer charge through a long array, but also to perform arithmetic operations, such as adding charge packets or dividing one packet into several others. For now we will focus on the clocking strategies for charge transfer, since other operations are based on permutations of the fundamental transfer sequence.

For best transfer efficiency, there should be little or no spacing between adjacent gates so that the lateral potential in every portion of the channel is always directly controlled. In order to maintain electrical isolation, two layers of polysilicon separated by a thin oxide are generally used for alternating gates. The clocking sequence for charge transfer depends on the number of independent clock signals connected to the gates. CCDs have been designed using two-, three-, and four-phase clocking schemes [94]. The two- and three-phase methods have the advantage of using fewer clocks and allowing higher device density than four-phase clocking. Two-phase clocking, however, requires a special implant and only allows charge transfer in one direction [92]. Three-phase clocking allows bi-directional transfer but necessitates connecting the same clock signals to different polysilicon layers, making it impossible to adjust the signals to overcome threshold mismatches between first- and second-level poly. Since, as will be seen in the following chapters, the operations required by the MSV algorithm necessitate bi-directional charge transfer as well as adjusting for threshold mismatches, a four-phase method was used in this design.

The charge transfer sequence using four-phase clocking is illustrated in Figure 11-7. At the beginning of the sequence, the signal charges are held under the gates labeled ϕ_1 and ϕ_2. In the next clock cycle, the signal ϕ_3 is brought high as ϕ_1 is brought low, causing the charge packets to spill into the empty potential wells created under the gates connected to ϕ_3 and move away from the lower potential regions now under ϕ_1. In the following cycles the process is repeated by raising and lowering the pairs ϕ_4–ϕ_2, ϕ_1–ϕ_3, and ϕ_2–ϕ_4, at which point the sequence repeats.

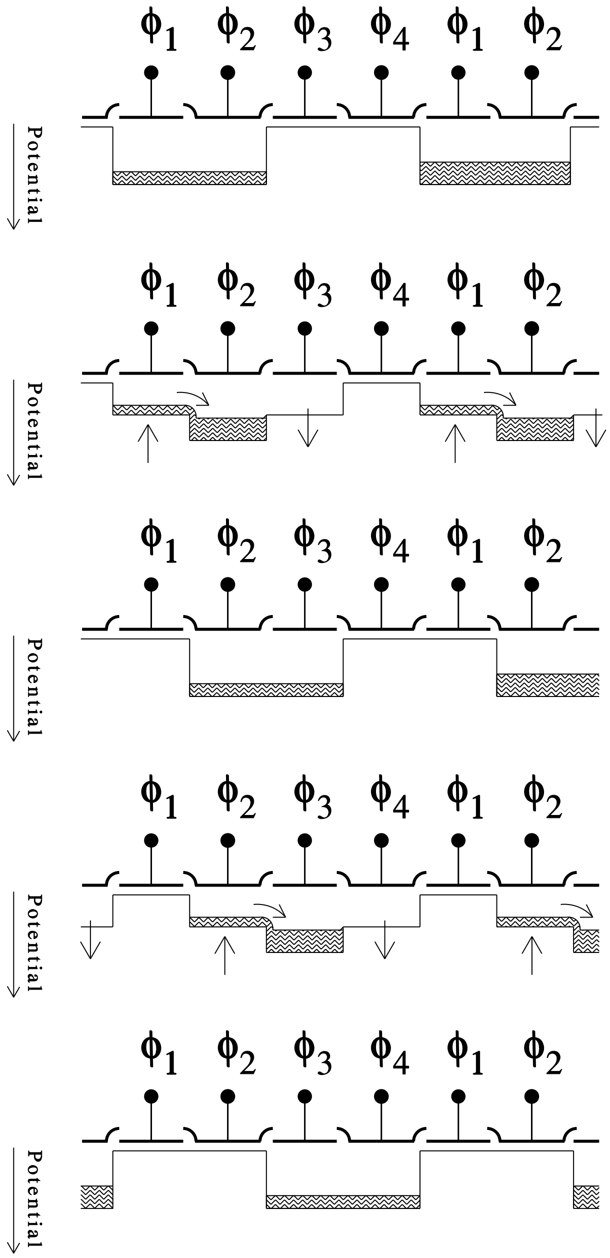


FIGURE 11-7: Charge transfer sequence with four-phase clocking.

11.4 Transfer Efficiency

One of the most important characteristics of charge-coupled devices is their charge *transfer efficiency*, ϵ, defined as the fraction of the total charge transferred to the receiving well in one clock cycle. A related quantity, the *transfer inefficiency* η, given by

$$\eta = 1 - \epsilon \tag{11.29}$$

is the fraction of total charge left behind.

In order for large CCD arrays to be useful for analog processing, transfer efficiencies greater than $\epsilon = 0.99999$ are required. The consequences of poor transfer efficiency can be easily understood from a simple calculation. After transferring a packet of initial size Q_0 through N gates, the final packet size, Q_N is given by

$$Q_N = Q_0 \epsilon^N \tag{11.30}$$

With $N = 1000$ and $\epsilon = 0.99999$, $Q_N/Q_0 = 0.99$, implying that by the time the charge packet reaches the end of the array, the signal will be diminished by 1% of its original value. With $\epsilon = 0.9999$, however, we would have $Q_N/Q_0 = 0.90$, resulting in a 10% loss in the signal.

Unless there is recombination in the channel, which should not occur to a significant extent in a well-designed device, the charge is not actually lost but is dispersed over the array, causing successive packets to be contaminated by the charge left behind. It can easily be seen that the amount of charge left under gate i, counting from $i = 0$ initially, is given by the binomial formula [93]

$$\frac{Q_i}{Q_0} = \left(\begin{array}{c} N \\ i \end{array} \right) \epsilon^i (1 - \epsilon)^{N-i} \tag{11.31}$$

There are three primary mechanisms causing poor transfer efficiency. The first is charge trapping by interface or bulk states. As previously explained, one of the primary advantages to using buried channel devices is the fact that the density of bulk states is much lower than that of the interface states at the Si–SiO_2 surface, resulting in much higher transfer efficiencies in BCCDs than in surface channel devices.

The second cause of poor charge transfer is a 'bumpy' channel. The lateral potential profile between the gate spilling the charge packet and the gate receiving the packet does not necessarily increase monotonically. Potential 'bumps', which can keep some amount of charge from reaching the neighboring well, can occur when the inter-gate spacing is too large, when there are changes in the channel width, or when there are corners in the channel [86]. In addition since, as seen from equation (11.23), ϕ_{max} is directly related to t_{ox}, potential bumps can be caused by variations in the oxide thickness over the length of the transfer gate.

The third mechanism affecting charge transfer is clock frequency. Intuitively, it is clear that for the charge to be completely transferred, it has to be allowed enough time to reach its destination. Quantitatively, the time needed for complete charge transfer in the absence of traps can be determined by analyzing the forces driving the charge packet.

When the empty potential well is created next to the charge packet by raising the voltage on the neighboring gate, the initial force pushing the charge into the new well is the mutual repulsion of the negatively charged electrons which generates a *self-induced field*, E_{si}. As the electron concentration decreases in the lower potential region being emptied, the self-induced field becomes less important and the *fringing field*, E_{ff}, becomes dominant [92], [93]. The fringing field is created by the lateral influence of the neighboring gates which are at a high voltage on the charge remaining in the emptying well. Because this influence increases as the effective capacitance, C_{eff}, between the charge and the gate directly above it decreases, the fringing fields are larger in buried channel than in surface channel devices, further enhancing their transfer efficiency. The final stage of charge transfer, after the self-induced and fringing fields have become negligible, is dominated by thermal diffusion which is the slowest of the three transport mechanisms [92].

Determining the exact time-varying charge distribution during transfer requires numerical solutions to the time-dependent Poisson's equation and continuity conditions. Approximate solutions, however, have been derived in references [93] and in [95]. Citing the results from the first reference, which give somewhat more insight into the nature of the solution, the transfer inefficiency after time t due to self-induced drift is

$$\eta_{si}(t) \approx \frac{1}{1 + t/t_{si}} \tag{11.32}$$

where

$$t_{si} = \frac{2L^2 C_{eff}}{\pi \mu_n Q_0} \tag{11.33}$$

with L being the length of the transfer gate and μ_n the electron mobility in the channel.

The transfer inefficiency due to fringe-field drift after time t is given by

$$\eta_{ff}(t) \approx e^{-t/t_{ff}} \tag{11.34}$$

with

$$t_{ff} \approx \frac{L}{2\mu_n |E_{min}|} \tag{11.35}$$

The minimum strength of the fringing field, $|E_{min}|$, is given by

$$|E_{min}| = \frac{\Delta V_g}{L} \left(e^{-\pi \epsilon_{Si}/C_{eff} L} - e^{-\pi \epsilon_{Si}/3C_{eff} L} \right) \tag{11.36}$$

where ΔV_g is the difference in the gate voltages on the discharging and receiving gates.

From equations (11.33), (11.35), and (11.36) it is clear that the characteristic transfer times increase at least as fast as the square of the gate length, L. Minimizing this parameter is thus crucial to designing a processor with both adequate speed and transfer efficiency.

11.5 Power Dissipation

11.5.1 On-chip dissipation

Because signal charge moves through the CCD array, there is an effective current flowing across a resistive medium causing power to be dissipated on chip. The dissipation per gate is given by [93],

$$\begin{aligned} P_{gate} &= \frac{Q}{\rho} (\mathbf{J} \cdot \mathbf{E}) \\ &= -Q (\mathbf{v} \cdot \mathbf{E}) \\ &= -\frac{Q (f_c L)^2}{\mu_n} \end{aligned} \tag{11.37}$$

where $\mathbf{J}$ is the current density, $\mathbf{E}$ is the lateral electric field, Q is the charge under the gate, $\mathbf{v}$ is the average charge velocity, f_c is the clock frequency, L is the gate length and μ_n is the carrier mobility.

To obtain a rough estimate of the on-chip power dissipation, we can set $\mu_n = 1500\text{cm}^2/$ V-sec, $f_c = 5$MHz, $L = 10\mu$m, and $Q = -6.4 \times 10^{-14}$ coulombs, which is the charge of 400000 electrons, giving

$$P_{gate} = 10^{-9}\,\text{W} \tag{11.38}$$

Even if the array contained 10^6 gates, the total on-chip dissipation due to charge transfer would be no more than 1mW. We can thus for all practical purposes ignore this contribution to the total power required to operate the processor.

11.5.2 Power dissipation in the clock drivers

The primary source of power dissipation in operating a CCD array is in the clock drivers which must supply the current to charge and discharge the large capacitive loads from all of the gates tied to a given clock phase at each cycle. For a square-wave signal the energy dissipated in the internal resistance of the clock driver when either charging or discharging a capacitance C is given by [93],

$$E_{clock} = \frac{1}{2}CV^2 \tag{11.39}$$

where V is the voltage swing on the capacitor. Since each gate is charged and discharged once per transfer cycle, the power dissipated per gate is

$$P_{gate} = CV^2 f_t \tag{11.40}$$

with f_t being the transfer cycle frequency.

A $100\mu\text{m}^2$ gate with nominal capacitance of $0.5\text{fF}/\mu\text{m}^2$ operating with a 5V swing and 1MHz transfer cycle frequency thus requires

$$P_{gate} = 1.25 \times 10^{-6}\,\text{W} \tag{11.41}$$

to be dissipated in the driver circuit. Operating a 256×256 four-phase array would therefore consume at least 82mW.

Power dissipation in the supporting circuitry can be reduced by using tuned sinusoidal drivers [93]. In these circuits, the total power required to drive each gate is

$$P_{gate} = \frac{\pi}{Q_f} CV^2 f_t \tag{11.42}$$

where Q_f is the quality factor of the driving oscillator. With well-tuned circuits, power dissipation in the clock drivers can be reduced significantly.

11.6 Charge Input and Output

In order to interface other circuits to a CCD processor, signal charge must be introduced into the array at some point and the stored charge must later be converted into a usable output signal. This chapter is thus concluded by discussing the specific I/O structures used in the MSV processor.

11.6.1 Fill-and-Spill input method

Input charge may be generated either electrically or optically. Optical input is of course necessary for image sensors, however, in a test circuit it is difficult to control. In designing the prototype MSV processor, an electrical technique known as the fill-and-spill method was used to generate the repeatable input signals required for testing the chip.

The fill-and-spill method exploits the relation (11.28) between maximum signal charge and the difference in neighboring gate voltages. The process is illustrated in Figure 11-8. Voltages V_{ref} and V_{in} are applied to two adjacent gates creating the relative potential profile shown, while the potential barrier to the right of the input gate is created by holding the stop gate (SG) at a lower voltage than V_{ref}.

Signal charge is supplied via the ohmic contact made to the $n+$ diffusion adjacent to the reference gate. In the fill stage, the diffusion potential, controlled by V_d, is lowered causing electrons to flood the higher potential regions beneath both the reference and input gates. On the next clock cycle, V_d is raised so that the diffusion potential is well above the reference channel potential, causing excess electrons to spill back into the diffusion and leave behind a charge packet Q of size, theoretically, given by

$$Q = (V_{in} - V_{ref})\,C_{in} \tag{11.43}$$

where C_{in} is the total capacitance of the input gate. Following the spill operation, the signal charge can be moved into the array by appropriately clocking the stop gate and transfer gate voltages.

Several variations on the basic structure shown in Figure 11-8 have been used with CCDs.

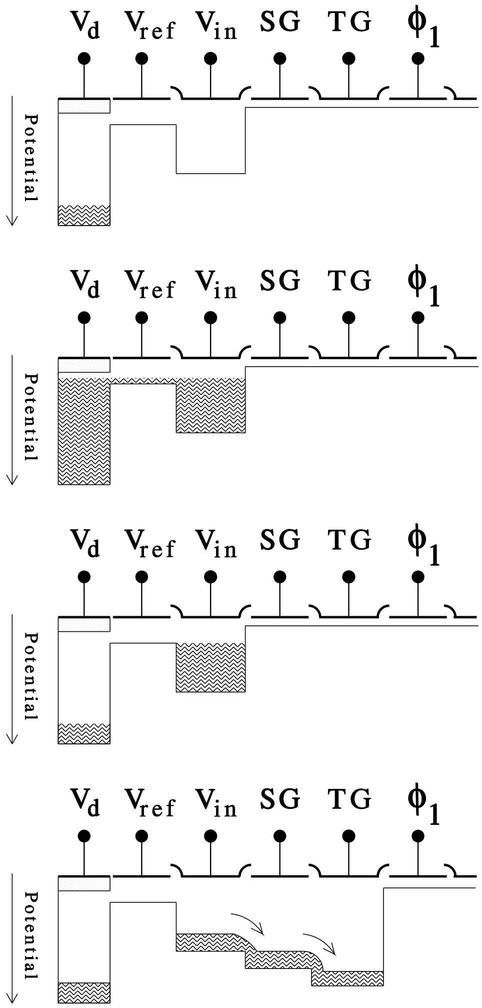


FIGURE 11-8: Fill-and-spill method for charge input

Since it is often useful to maintain a linear relation between the charge packet size and the voltage difference $(V_{in} - V_{ref})$, fill-and-spill structures are sometimes built from surface channel devices [86]. To avoid threshold mismatches, the reference and input gates are usually designed in the same level of polysilicon. This necessitates placing either a floating diffusion or a dummy gate in second-level poly [65] to maintain the lateral continuity of the channel. In the MSV processor, threshold mismatch in the input stage was not a concern as the input test signals were completely controllable and any effects due to mismatch could be removed by adjusting the input level. In the present design, it was thus simpler to use different polysilicon levels for the input and reference gates, as shown in the diagram.

The primary difficulty in designing a fill-and-spill device to produce consistent output levels for given values of V_{in} and V_{ref} is that the charge packet size is in fact a random variable. First, the amount of charge stored in the capacitor under the input gate will fluctuate due to thermal noise, with the mean value of the fluctuations given by [96]

$$\overline{\Delta Q} = \sqrt{kTC_{in}} \tag{11.44}$$

where k is Boltzmann's constant, and T is the absolute temperature. The ratio of noise to the total signal is thus

$$\frac{\overline{\Delta Q}}{Q} = \frac{1}{V_{in} - V_{ref}}\sqrt{\frac{kT}{C_{in}}} \tag{11.45}$$

A second problem causing the signal charge to fluctuate is thermionic emission. This is the problem discovered by Keast [86] which limited the resolution of the CCD-based absolute value of difference circuit. This circuit, which is essentially composed of two cross-coupled fill-and-spill structures, has a *dead zone* for small input differences due to the fact that small potential barriers are not very effective in holding back the signal charge. Thermionic emission is caused by the finite kinetic energy of the electrons. Just as water will boil over the side of a pot filled to the rim, the kinetic energy of electrons at the 'top' of a potential barrier can give them enough boost to jump over the side.

In the fill-and-spill structure, the barrier under the reference gate is used to block electrons from spilling back into the high potential diffusion. Because of the thermionic effect, however, fewer electrons than predicted by equation (11.43) will remain under the input gate. Given sufficient time, the charge level will drop until the potential difference under the input and reference gates is large enough to overcome the average kinetic energy of the electrons. If the difference between V_{in} and V_{ref} is small, however, this may not occur before all of the signal charge is lost.

One solution to limiting the effect of thermionic emission is to run the input process fast enough so that most of the energetic electrons do not have time to jump the barrier. This idea was used by Keast to improve the resolution of the absolute value of difference circuit. It is not a good idea, however, for operating the input circuit if stable signal levels are desired, since the amount of charge lost per unit time is not a well-controlled quantity. For best stability, the fill-and-spill structure should be operated *slowly* so that the amount of charge will decrease to the level, which is a function only of the temperature T, where the emission current is negligible.

11.6.2 Floating gate amplifier output structures

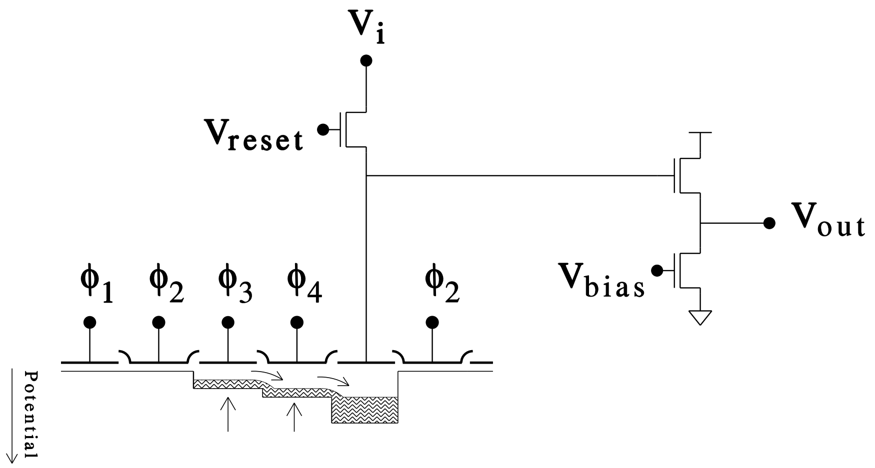
Sensing the amount of charge in a given packet is usually performed by injecting it onto the 'floating' plate of a precharged capacitor and measuring the resulting change in voltage. Typically, the sensing capacitor is either a gate or a diffusion connected to a high input impedance buffer. In the present design, the floating gate technique was used exclusively because it allows for non-destructive sensing, which is necessary since measurements must be made at several different stages. In addition, the floating gate structure allows greater sensitivity, lower noise, and better matching than the floating diffusion structure [97].

A diagram of a floating gate structure placed in a CCD array is shown in Figure 11-9a. A reset transistor initializes the gate to a voltage V_i and is then turned off, disconnecting the gate from the clock signal. Charge is injected into the potential well under the floating gate by clocking the other gates as described in Section 11.3. The change in voltage, $\Delta V_g = V_f - V_i$, is buffered by a source follower which provides a high impedance connection to the gate and sets the load capacitance to a fixed value.

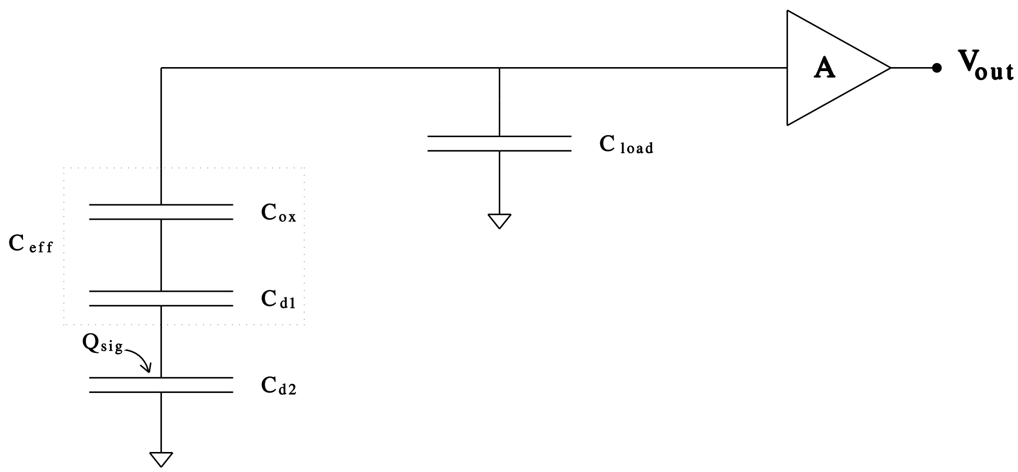
The relation between signal charge, Q_{sig} and ΔV_g, is best understood using the lumped capacitance model of Figure 11-9b. C_{ox}, C_{d1}, and C_{d2} are as defined in Section 11.2 while C_{load} represents the combined normalized capacitance from the source follower input gate, the drain diffusion of the reset transistor, the overlap of the adjacent gates, and parasitic sidewall capacitances.

$$C_{load} = \frac{C_{sf} + C_{drain} + C_{ovl} + C_{sw}}{A_g} \tag{11.46}$$

Normalization by the gate area, A_g, is necessary to maintain consistency of units.



a.) Floating gate structure placed in linear array.



b.) Lumped capacitance model.

FIGURE 11-9: Charge sensing using the floating gate technique

The channel charge is divided between the capacitors C_{eff} and C_{d2}, whose common 'plate' is at the potential maximum ϕ_{max}. If charge Q_{sig} is injected into the channel, it will also divide between the two capacitors such that

$$Q_{sig} = Q_{eff} + Q_{d2} \tag{11.47}$$

where

$$Q_{eff} = C_{eff}\,(\Delta\phi_{max} - \Delta V_g) \tag{11.48}$$

and

$$Q_{d2} = C_{d2}\,\Delta\phi_{max} \tag{11.49}$$

The quantity $\Delta\phi_{max}$ represents the change in the maximum channel potential, while $(\Delta\phi_{max} - \Delta V_g)$ is the change in voltage across C_{eff} as seen from the 'bottom plate'.

The charge Q_{eff} on the bottom plate of C_{eff} must be mirrored by an equal charge of opposite polarity, $-Q_{eff}$, on the top plate. Since the total charge shared between the top plate of C_{eff} and C_{load} is constant, the charge on C_{load} must increase by $+Q_{eff}$

$$\Delta Q_{load} = +Q_{eff} \tag{11.50}$$

so that

$$\Delta V_g = \frac{Q_{eff}}{C_{load}} \tag{11.51}$$

Combining (11.48), (11.49), and (11.51) we obtain the following expression for Q_{d2} in terms of Q_{eff}:

$$Q_{d2} = C_{d2}\,Q_{eff}\left(\frac{1}{C_{eff}} + \frac{1}{C_{load}}\right) \tag{11.52}$$

and taking equation (11.47) into consideration, we obtain the desired expression

$$\Delta V_g = \frac{Q_{sig}}{C_{load}}\left(\frac{1/C_{d2}}{1/C_{d2} + 1/C_{eff} + 1/C_{load}}\right) \tag{11.53}$$

In designing a floating gate amplifier, we normally want as much voltage swing at the output as possible. The final voltage on the gate, V_f, is limited however by the charge capacity equation (11.28). If we let V_l represent the low clock voltage which is applied to the gates neighboring the sense node and let $Q_{max} = -qN_{e,max}$ represent the maximum signal, then from (11.28) we must have

$$\begin{aligned} V_f - V_l &\geq \frac{qN_{e,max}}{C_{eff}} \\ &= -\frac{Q_{max}}{C_{eff}} \end{aligned} \tag{11.54}$$

To simplify notation, let α denote the quantity in parentheses in equation (11.53):

$$\alpha = \frac{1/C_{d2}}{1/C_{d2} + 1/C_{eff} + 1/C_{load}} \tag{11.55}$$

such that the minimum final gate voltage, $V_{f,min}$, is given by

$$V_{f,min} = V_i + \alpha \frac{Q_{max}}{C_{load}} \tag{11.56}$$

We can then eliminate Q_{max} from equations (11.54) and (11.56) to obtain

$$V_{f,min} = \frac{C_{load}V_i + \alpha\, C_{eff}\mathbf{V}_l}{C_{load} + \alpha\, C_{eff}} \tag{11.57}$$

For given values of V_i and V_l, we can design the floating gate for a desired $V_{f,min}$ by adjusting C_{load}. The smaller C_{load} is with respect to $\alpha\, C_{eff}$, the closer $V_{f,min}$ will be to V_l. We do have to be careful, however, to not make C_{load} too small as its value also affects the maximum charge size, and therefore the signal to noise ratio. Eliminating V_f from equations (11.54) and (11.56), we obtain the following expression for $N_{e,max}$

$$N_{e,max} = -\frac{Q_{max}}{q} = \frac{1}{q}\left(\frac{V_i - V_l}{\alpha/C_{load} + 1/C_{eff}}\right) \tag{11.58}$$

As $C_{load} \to 0$ so does $N_{e,max}$.

Chapter 12

CCD Edge Detector Design

The floor plan of the complete MSV edge detection processor is shown in Figure 12-1. There are four basic operations performed by the processor:

1. Charge input,
2. Smoothing,
3. Computing the magnitude of the difference between values at neighboring nodes, and
4. Storing and reading out the binary edge signals.

A fill-and-spill structure placed at the top of the block marked 'Charge Input and Shift Register' is used to load each pixel of the image, converting the brightness values to signal charge. An image is loaded into the array one column at a time, using the vertical shift register to move the pixels to their appropriate rows. Once the last pixel of the column has been read in, the contents of the entire shift register are transferred horizontally into the processing array.

Both the smoothing and differencing operations are performed within the array. Edge charges are stored separately at each cell for every horizontal and vertical node pair. They are not coalesced on-chip into one edge signal per node, as described at the end of Section 5.3, in order to leave more flexibility in the design of external circuits which interface to the processor. The horizontal and vertical edge signals for one row, which is selected by the decoder block to the right of the array, can be output in parallel at the end of any smoothing/differencing cycle without disrupting the proper operation of the algorithm.

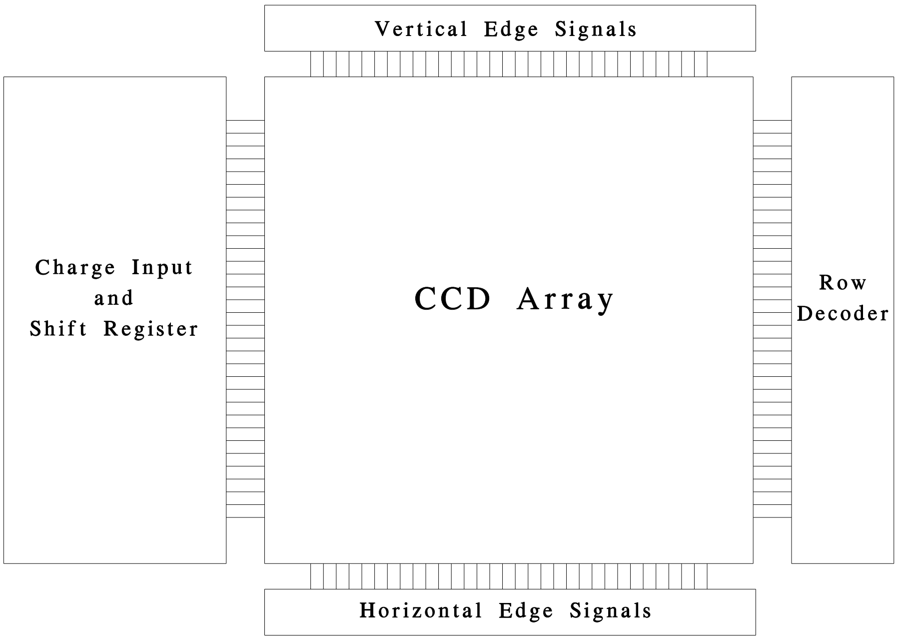


FIGURE 12-1: Floor plan of the complete MSV processor.

In the following sections I will describe the design, operation, and layout of the circuits used in the prototype MSV processor fabricated through MOSIS. I will first discuss the architecture of the unit cells which compose the array and then present the detailed design of the CCD- and transistor-based processing elements. Test results from the fabricated circuits are presented in the next chapter.

12.1 CCD Processing Array

A block diagram of the unit cell is shown in Figure 12-2, with the corresponding layout, measuring $224\lambda \times 224\lambda$[1], shown in Figure 12-3. At the boundary of the cell are the CCD gates in alternating levels of polysilicon which are sized so that when cells are abutted to form the processing array, the gate structure seen in Figure 5-3 results. Signal charges proportional to the pixel brightness values are stored under the large gates at the corners of the cell. One floating gate amplifier (FGA) per cell senses the charge under the gate in the

[1] In the Orbit CCD process $\lambda = 1\mu$m.

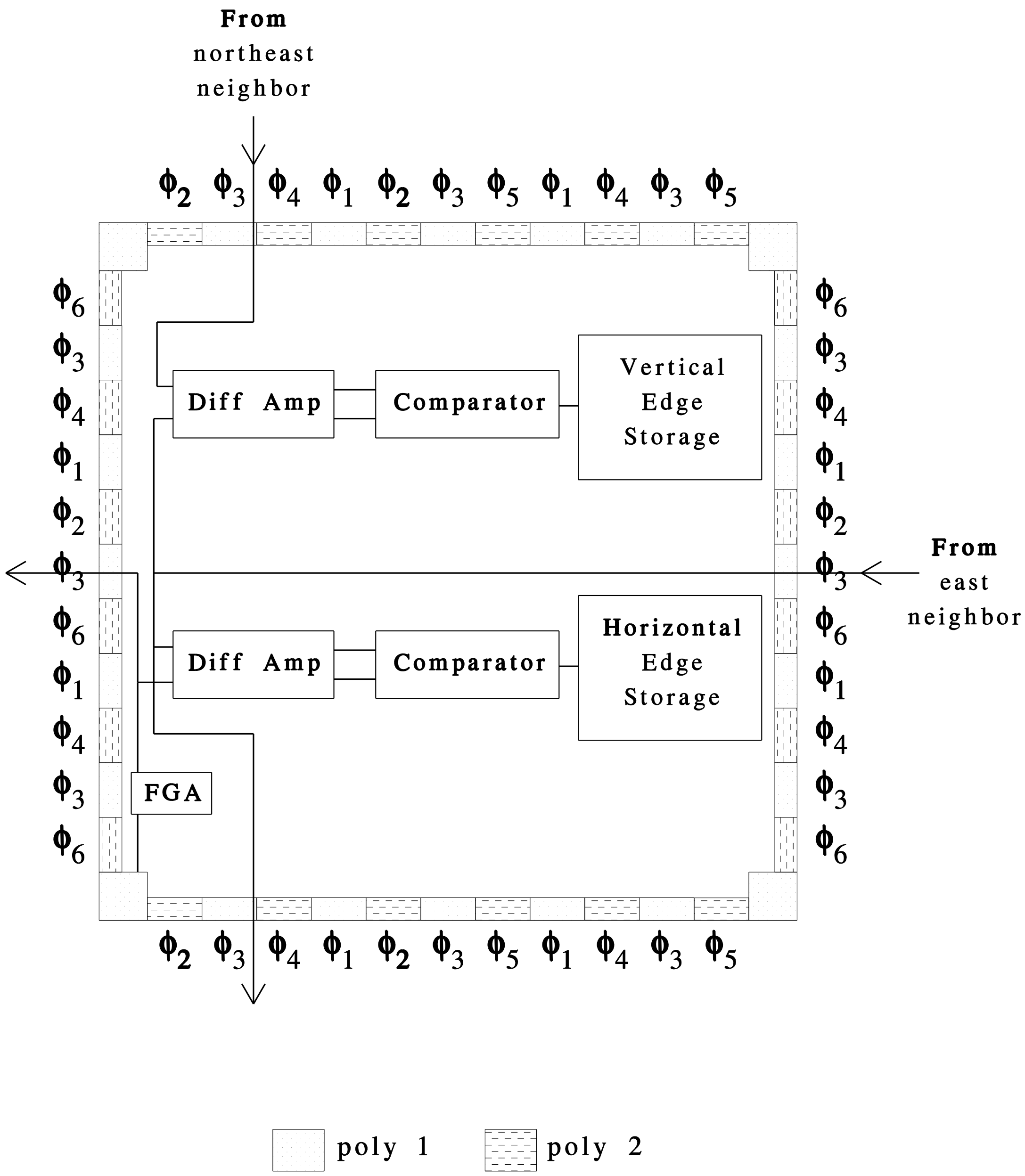


FIGURE 12-2: Unit cell architecture.

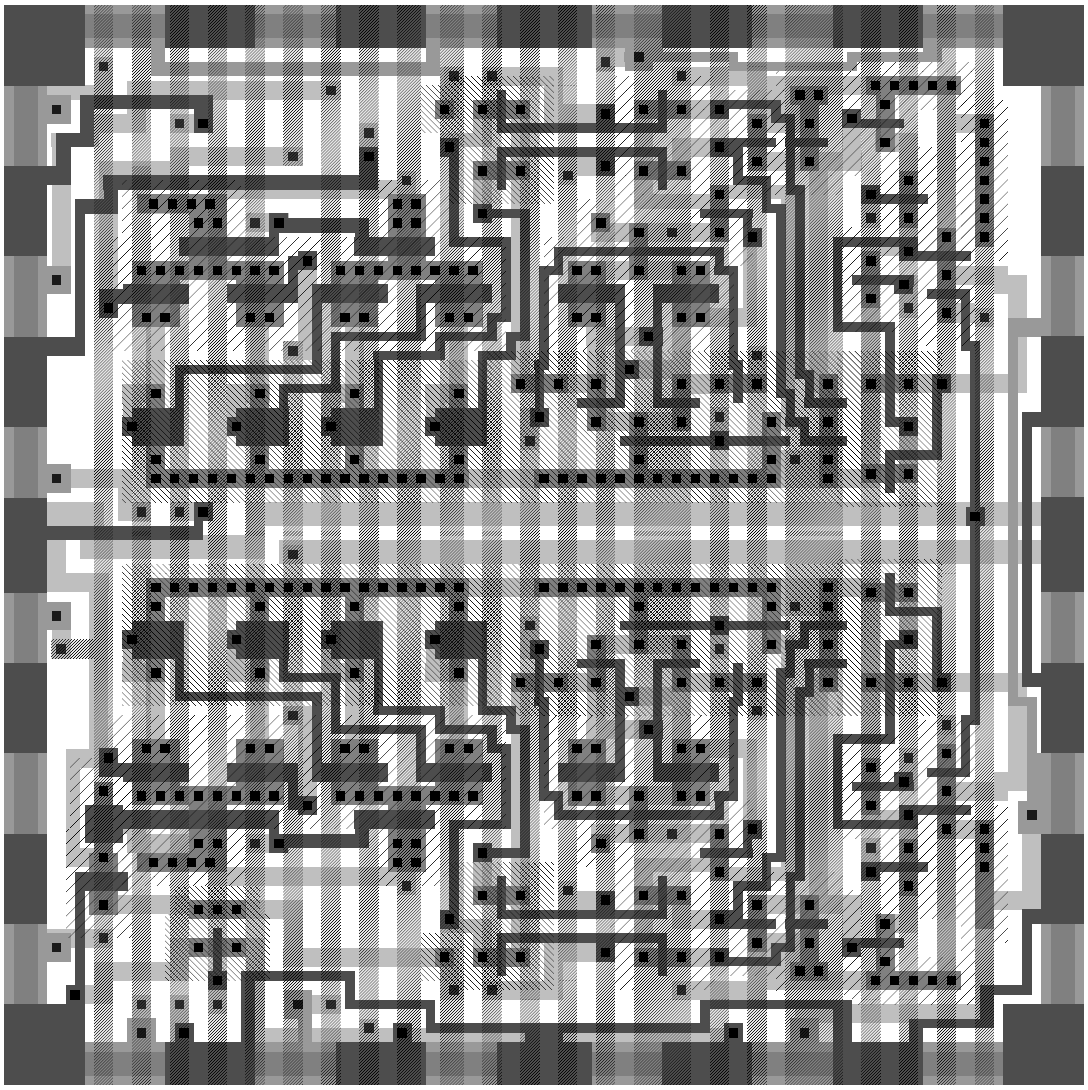


FIGURE 12-3: Unit cell layout.

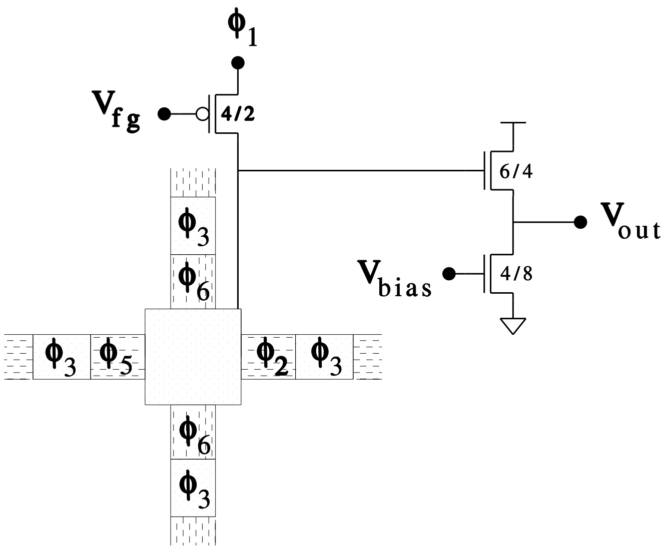


FIGURE 12-4: Floating gate design.

lower lefthand corner and feeds the output voltage to the four differential amplifiers which are paired with its nearest neighbors. For reasons dictated by the layout, it was simplest to have the floating gate amplifier communicate with the differencing circuit directly adjacent to it and to those in the neighboring cells to the west and southwest. The edge signals stored in the blocks marked 'Horizontal' and 'Vertical' thus correspond to edges between the two nodes at the base of the cell and the two on the righthand vertical side, respectively.

12.1.1 Charge sensing

The more detailed picture of the floating gate amplifier used for charge sensing at the signal nodes is shown in Figure 12-4. The clock phase ϕ_1 is gated through the p-type reset transistor controlled by the signal V_{fg}. When V_{fg} is brought low, the node gate voltage is controlled by ϕ_1 for the purposes of charge transfer and storage. When V_{fg} is brought high, however, the gate is left floating and can thus be used for measuring charge levels as described in Section 11.6.2.

In order to sense the signal level, the node must be initially emptied by transferring the charge out through the four connecting branches. Once this is done and ϕ_1 is brought high, V_{fg} is also brought high, turning off the reset transistor and initializing the node gate

Parameter	Value	Units
N_D	3.8×10^{16}	cm^{-3}
N_A	5.05×10^{15}	cm^{-3}
t_{ox}	420	Å
V_{fb}	0.6	V
x_n	0.4	μm
C_{ox}	8.10×10^{-16}	F/μm^2
C_{p1-p2}	5.0×10^{-16}	F/μm^2
C_{jsw0}	3.46×10^{-16}	F/μm
ϕ_{bi}	0.8	V

Table 12.1: Orbit CCD/CMOS process parameters (from 10-28-93 run).

voltage to the value V_i. The signal charge is then returned and dumped back into the empty potential well, causing the floating gate voltage to change to the value V_f, where

$$V_f = V_i + \frac{Q_{sig}}{C_{load}} \left(\frac{1/C_{d2}}{1/C_{d2} + 1/C_{eff} + 1/C_{load}} \right) \tag{12.1}$$

The minimum value which V_f can attain is, from equation (11.57),

$$V_{f,min} = \frac{C_{load} V_i + \alpha\, C_{eff} \mathbf{V}_l}{C_{load} + \alpha\, C_{eff}} \tag{12.2}$$

where V_l is the low clock voltage and

$$\alpha \equiv \frac{1/C_{d2}}{1/C_{d2} + 1/C_{eff} + 1/C_{load}} \tag{12.3}$$

In the test system designed to evaluate the prototype processor, the clock drivers were operated between voltages $V_h = 4.5$V and $V_l = 0.6$V. Targetting a full scale swing of 2V, *i.e.,* $V_{f,min} = 2.5$V, we can compute the nonlinear depletion capacitances C_{d1} and C_{d2} with $N_e = N_{e,max}$ from the equations developed in Section 11.2 using the values of the Orbit process parameters given in Table 12.1. From equation (12.2) we then obtain the required load capacitance, C_{load}. The results are given below in Table 12.2.

The voltage change on the floating gate is measured via a source-follower buffer whose design was determined by two primary considerations. The first was the need to minimize

Parameter	Value	Units
$N_{e,max}$	3272	$electrons/\mu m^2$
ϕ_{max}	4.20	V
x_{max}	0.27	μm
C_{d1}	4.6×10^{-16}	F/μm^2
C_{d2}	9.1×10^{-17}	F/μm^2
C_{eff}	2.9×10^{-16}	F/μm^2
C_{load}	1.4×10^{-16}	F/μm^2

Table 12.2: Floating gate parameter values for $Vg = 2.5$V and $N_e = N_{e,max}$.

the loading capacitance on the floating gate, while the second, and most critical, was the need to have the output voltage in the correct range for interfacing directly to the differential amplifiers. For the latter reason, an n-type source follower was used, despite the fact that a higher gain could be achieved with a p-type design with separate wells connected to the sources of the bias and input transistors.

Within these considerations, it was of course desirable for the gain to be as high as possible. The theoretical small signal gain of the n-type configuration, as shown in Figure 12-4, is given by [98]

$$\frac{v_o}{v_i} = \frac{g_m}{g_m + g_{mb} + 1/R_{eff}} \tag{12.4}$$

where g_m and g_{mb} are the small signal gate-source and source-bulk transconductances, respectively, of the input transistor and R_{eff} is the parallel combination of the input and bias transistor output resistances. Since little can be done to reduce g_{mb}, maximizing the gain involves making R_{eff} as large as possible compared to $1/g_m$.

To increase the drain resistances, both transistors were drawn with long channels to reduce channel length modulation, and the W/L ratio of the bias transistor was dimensioned to make the drain current, I_D, small. Since the output resistance is proportional to $1/I_D$ while g_m increases as $\sqrt{I_D}$ [98], the product $g_m R_{eff}$ increases when the drain current is lowered as $1/\sqrt{I_D}$. The W/L ratio of the input transistor, on the other hand, was determined by the output voltage range needed by the differential amplifiers. The sizes used in the final design were $W/L = 6\lambda/4\lambda$ for the input transistor and $W/L = 4\lambda/8\lambda$ for the bias transistor.

Figure 12-5 shows the simulated behavior of the source follower with a bias voltage of $V_{bias} = 1$V based on the Orbit process parameters from the 10-28-93 run. From the

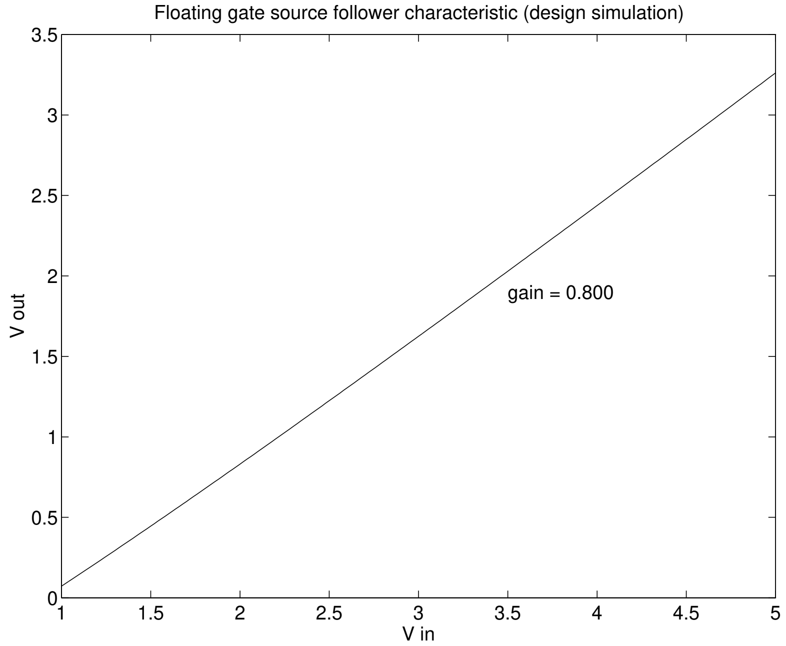


FIGURE 12-5: Simulated source follower characteristic for floating gate amplifier (V_{bias} = 1V).

simulations, I_D is computed as 63nA, giving a static power dissipation of 313nW for V_{DD} = 5V. The predicted DC gain is $v_o/v_i = 0.800$, with a 3dB bandwidth of 800kHz for a 600fF load approximating that of the differential amplifier input gates. For the maximum and minimum inputs of 4.5V and 2.5V, the output voltages are 2.85V and 1.22V, respectively, giving a predicted full scale swing of 1.63V.

The normalized load capacitance, C_{load}, on the floating gate is computed from equation (11.46) by summing the contributions from all loads and dividing by the total gate area. The dimensions of the node gates were $30\mu m \times 30\mu m$, with $40\mu m \times 2\mu m$ of overlap area ($2\mu m$ being the minimum poly1-poly2 overlap width allowed in the Orbit design rules) and $80\mu m$ of sidewall perimeter. The capacitances for each element loading the gate are computed and given in Table 12.3, below. The sidewall capacitance, C_{sw}, which is a function of the voltage between the buried channel and the substrate, was computed at the minimum channel potential of 4.2V, for which the unit capacitance C_{jsw} = .14 fF/μm. The total capacitance from all sources is thus estimated at 82.6fF, giving

$$C_{load} = \frac{82.6fF}{30\mu m \times 30\mu m} = 9.2 \times 10^{-17}\mathrm{F}/\mu m^2 \tag{12.5}$$

which is comfortably below the value of 1.4×10^{-16} F/μm^2 required for a 2V swing.

C_{sf}	input gate	16.5 fF
	gate-source	13.2 fF
	total	29.7 fF
C_{drain}	reset transistor	1.6 fF
C_{ovl}	$80\mu m^2 \times C_{p1-p2}$	40 fF
C_{sw}	$80\mu \times C_{jsw}(4.1V)$	11.3 fF
Total		82.6 fF

Table 12.3: Capacitances loading the floating node gate.

The number of electrons corresponding to the maximum signal is found by multiplying $N_{e,max}$ for the minimum final voltage of 2.5V by the gate area, A_g. From Table 12.2, we have $N_{e,max} = 3493$ electrons/μm^2, giving a total of approximately 3.1×10^6 electrons.

12.1.2 Clock sequences

Charge transfer and smoothing

Six different clock phases were required to operate the MSV gate array. Only four are needed to move charge laterally across the array when loading or unloading an image. However, for the smoothing operation the motion is along all four branches connected to each node and two more clock phases are required to control the direction of charge flow.

For simple lateral transfer a pseudo four-phase clocking scheme was used. Charge movement in this scheme is identical to that described in Section 11.3, however, the clocking sequence is more complicated because the phases are not arranged in a simple 1-2-3-4 repeating pattern. The clock signals connected to each gate are shown in Figure 12-2. When the charge is held under the node gate, the signal ϕ_1 is high and V_{fg} is low, as are the signals ϕ_2, ϕ_5, and ϕ_6 connected to the gates neighboring the node. To move charges from the nodes on the left side of the unit cell to those on the right, V_{fg} and ϕ_6 are held low and the following sequence is executed: ($\phi_2\uparrow$), ($\phi_1\downarrow$, $\phi_3\uparrow$), ($\phi_2\downarrow$, $\phi_4\uparrow$), ($\phi_3\downarrow$, $\phi_1\uparrow$), ($\phi_4\downarrow$, $\phi_2\uparrow$),

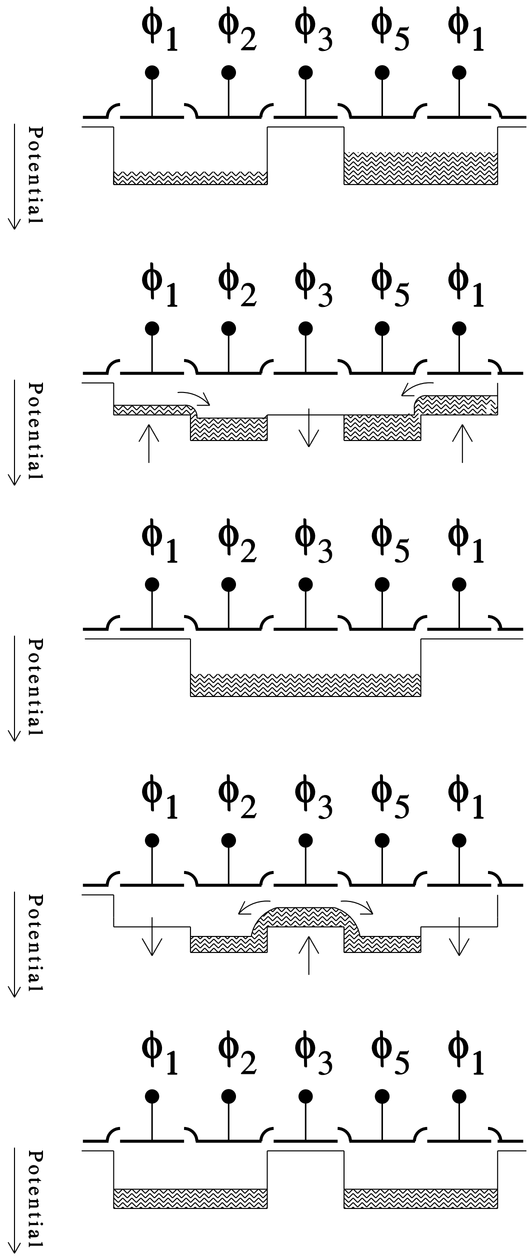


FIGURE 12-6: Charge averaging operation.

$(\phi_1\downarrow,\ \phi_3\uparrow)$, $(\phi_2\downarrow,\ \phi_5\uparrow)$, $(\phi_3\downarrow,\ \phi_1\uparrow)$, $(\phi_5\downarrow,\ \phi_4\uparrow)$, $(\phi_1\downarrow,\ \phi_3\uparrow)$, $(\phi_4\downarrow,\ \phi_5\uparrow)$, $(\phi_3\downarrow,\ \phi_1\uparrow)$, $(\phi_5\downarrow)$. The up arrow, $\uparrow$, is used to imply that the corresponding signal is brought high, while the down arrow, $\downarrow$, indicates that it is brought low.

The 2-D smoothing operation is performed in two passes by sequentially executing a 1-D smoothing operation along the horizontal and vertical branches. Each 1-D operation consists of four steps: (1) splitting the charge held under the node gates into two equal packets, (2) moving the packets out the branches connected to the node towards the mixing gate, (3) averaging the packets from adjacent nodes, and (4) returning the averaged packets back to the node gate where they are added together. Splitting is performed when the charge is entirely confined under the node gate (V_{fg} low and ϕ_1 high) by first raising the signals on the adjacent gates (ϕ_2 and ϕ_5 for horizontal smoothing; ϕ_6 for vertical smoothing) which causes the charge to distribute itself evenly over the high potential regions under the three gates. Bringing ϕ_1 low then divides the charge into two equal and isolated packets.

For the horizontal smoothing operation, ϕ_3, ϕ_4, and ϕ_6 are held low during the splitting phase. Executing the sequence $(\phi_3\uparrow)$, $(\phi_2\downarrow,\ \phi_5\downarrow,\ \phi_4\uparrow)$, $(\phi_3\downarrow,\ \phi_1\uparrow)$, $(\phi_4\downarrow,\ \phi_2\uparrow,\ \phi_5\uparrow$ then moves the charge packets away from the node gate along the horizontal branches and creates the situation shown at the top of Figure 12-6, where the packets from two adjacent nodes are about to collide. The central gate controlled by ϕ_3 on each of the branches is referred to as the *mixing gate*. The two unequal packets from the neighboring nodes, initially separated by the barrier at the mixing gate are added together when ϕ_3 is raised and ϕ_1 is brought low. When ϕ_3 is subsequently lowered and ϕ_1 is brought high, the summed charge is then divided in two. The averaged packets are then returned to their respective nodes, where the results from the opposing branches are combined, by executing the inverse sequence of that used to move them away.

It can easily be shown that the horizontal operation is equivalent to convolving the stored image with the discrete kernel

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \tag{12.6}$$

while the vertical operation, which is performed in a similar manner by appropriately substituting ϕ_6 for the signals ϕ_2 and ϕ_5 in the clocking sequence, is equivalent to a convolution with

$$\frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \tag{12.7}$$

The complete smoothing cycle, which consists of performing first the horizontal and then the vertical operations, is thus equivalent to a convolution with the 2-D binomial kernel

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{12.8}$$

Each successive smoothing cycle repeats this operation so that performing n cycles results in the convolution of the original image with the $(n-1)$th convolution of this kernel with itself. For instance, two cycles corresponds to a convolution with

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{12.9}$$

The size of the smoothing filter, which is controlled by the number of cycles performed, can thus, theoretically, be made arbitrarily large.

Avoiding backspill

One problem which arises from using two polysilicon levels in the gate array is that the channel potentials are not the same for equal gate voltages applied to both levels. For the Orbit process, the difference in the poly1 and poly2 channel potentials is approximately 0.4V, with poly2 being higher. The consequence of the potential mismatch is that it can result in incomplete transfer due to backspill. A more accurate depiction of the potential levels along the array during charge transfer is illustrated in Figure 12-7. When a poly1 gate is brought low, a small amount of charge can be transferred backwards by being pulled into the slightly higher potential region under the adjacent poly2 gate on the other side.

To avoid this problem, while keeping the clock driver circuits simple, a special level shift

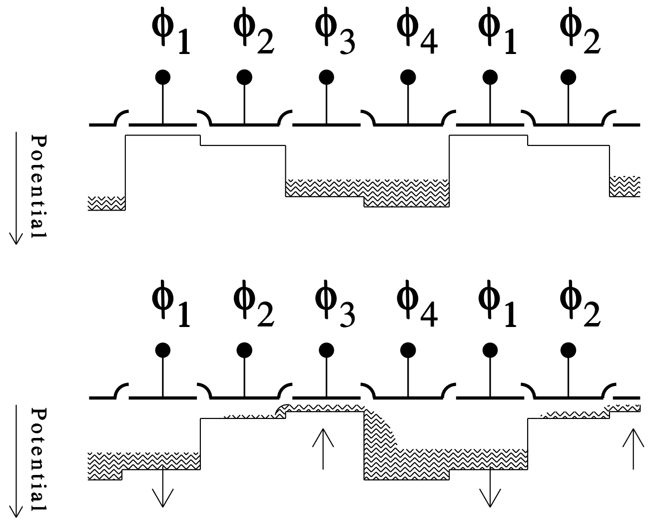


FIGURE 12-7: Backspill caused by potential mismatch.

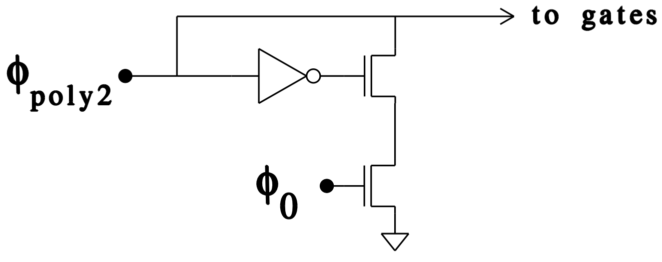


FIGURE 12-8: Level shift circuit for correcting potential mismatch problem.

circuit, shown in Figure 12-8, was added on chip to the poly2 clock lines. When the clock phase is low and the ϕ_0 signal is brought high, the clock line is pulled all the way to ground. Since the low clock voltage used in the test system was $V_l = 0.6$V, this causes the channel potential under the poly2 gate to be slightly higher than that under the poly1 gate which is being brought low, and thus prevents backspill. In the following clock cycle when the poly2 gate is brought low, ϕ_0 is held low so that the potential under the poly2 gate at V_l will be slightly higher than that of the poly1 gate behind it. As a result, there is always some small barrier preventing charge from travelling in the wrong direction.

12.1.3 Boundary processing and charge input

The cells on the array boundary are different from those of the interior, primarily because they are missing one or more neighbors. In addition, the cells along the west boundary of the array contain the charge input structure and the vertical shift register for moving the input signals to the appropriate row. Only the north and west boundary cells, shown in Figures 12-9 and 12-11, contain differencing and threshold circuits—the north cell for the horizontal pair at the base of the cell, and the west cell for the vertical pair on the right side of the cell.

Smoothing on the boundary cells is inhibited in the direction away from the array by keeping the gates normally used for mixing at a low voltage. For the north and south cells, these gates are simply grounded, while on the east and west cells, they are connected to a clock signal, ϕ_{SG} (for stop gate), which is held low during the smoothing operations. An independent signal is necessary for the stop gates in the east and west boundary cells since they must be clocked to allow charge to be moved into and out of the array.

When loading and unloading images, ϕ_{SG} is driven identically to ϕ_3 in the lateral transfer sequence. An image is loaded column by column, shifting the entire contents of the array one node to the east as the next column is entered. Any charge previously held in the array is also moved, and in particular, the charge held under the node gate of the east boundary cell, shown in Figure 12-10, is dumped out into the high potential $n+$ diffusion next to the stop gate. In order to have the option of measuring the signal levels of the smoothed image as it is removed from the array, the floating gate amplifier outputs on the east cells are connected to an output pad driver and can thus be sensed off-chip. To use this option, the horizontal transfer sequence must be modified to include turning off the reset transistor

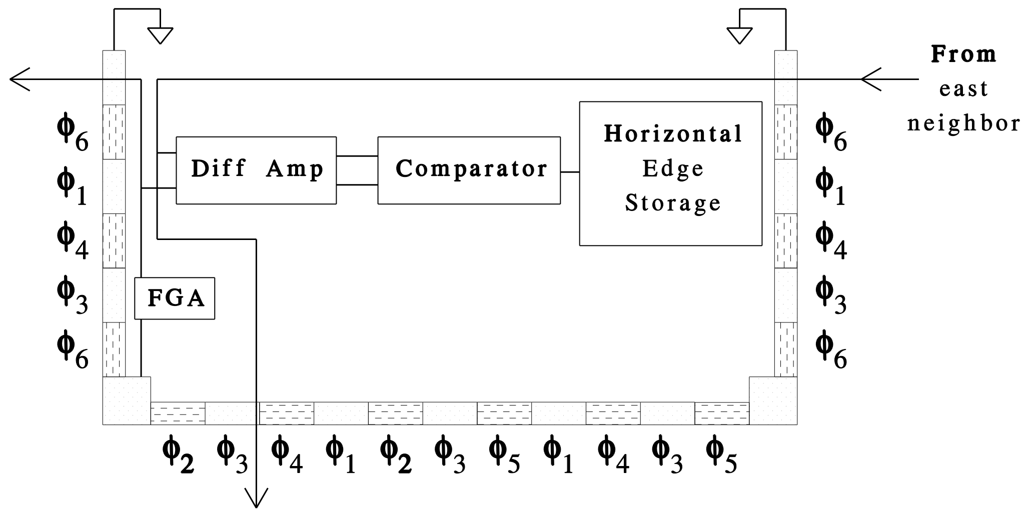


FIGURE 12-9: Boundary cell design (north).

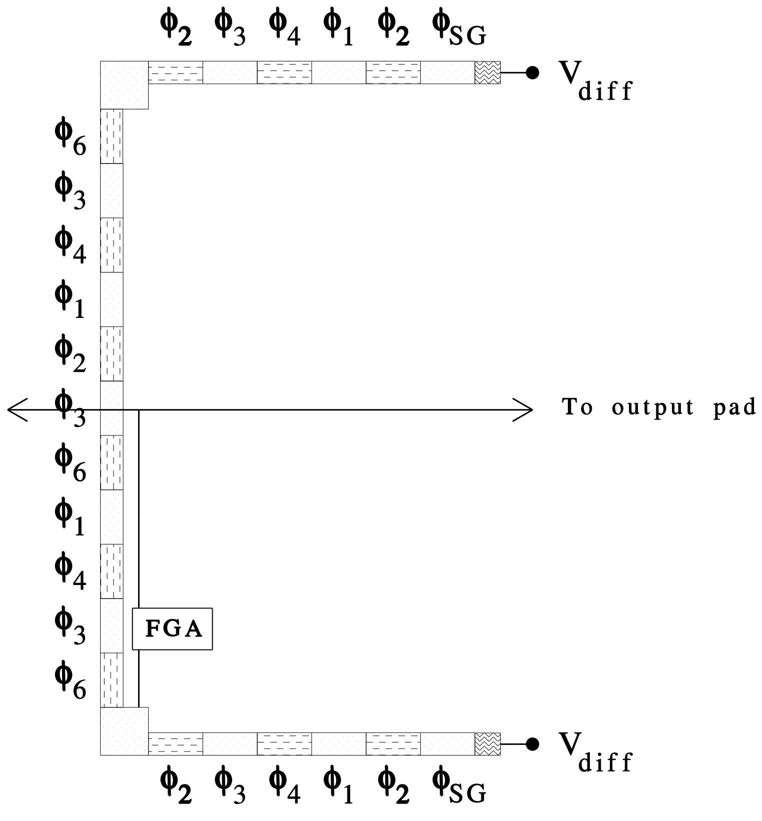


FIGURE 12-10: Boundary cell design (east).

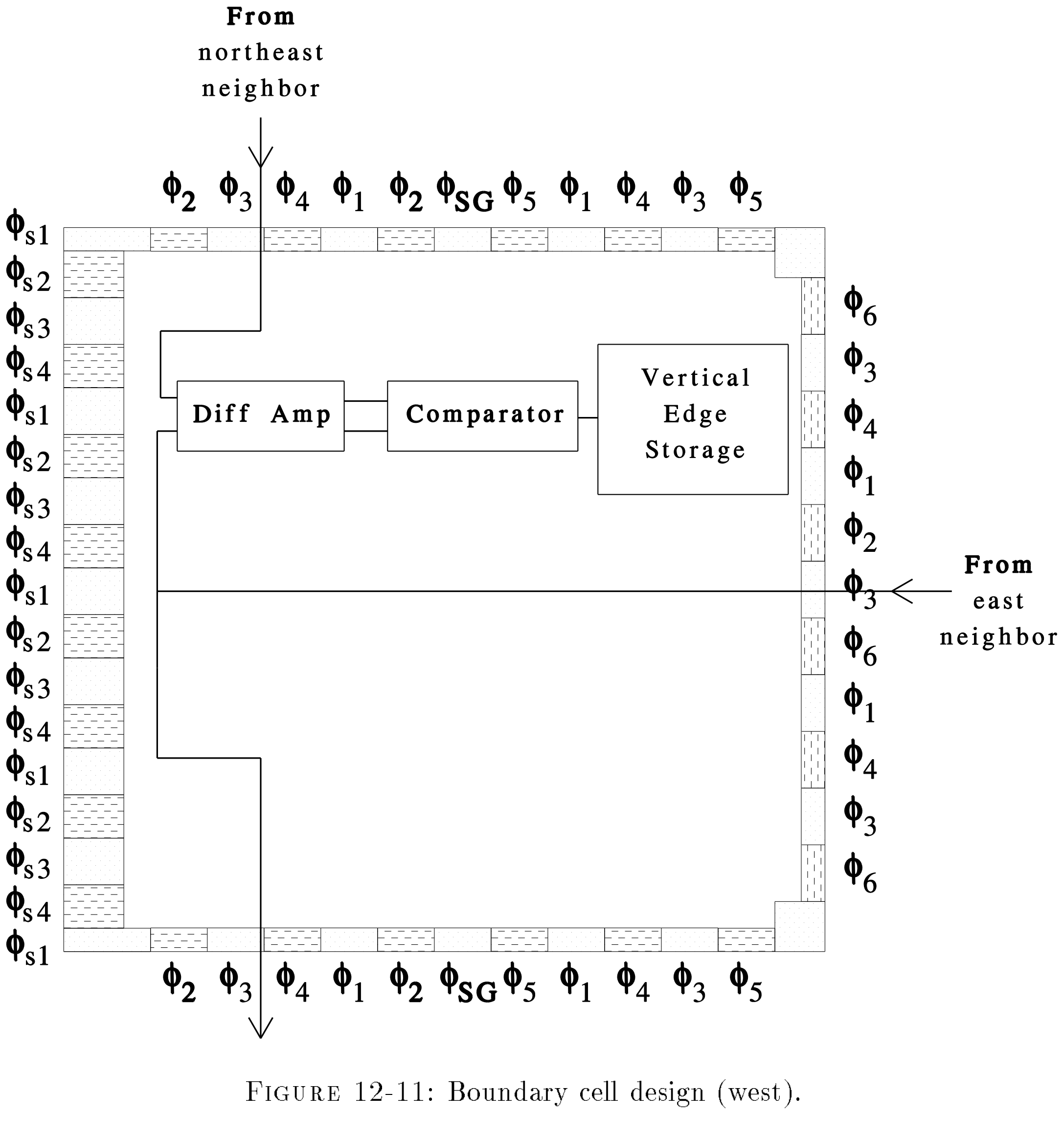


FIGURE 12-11: Boundary cell design (west).

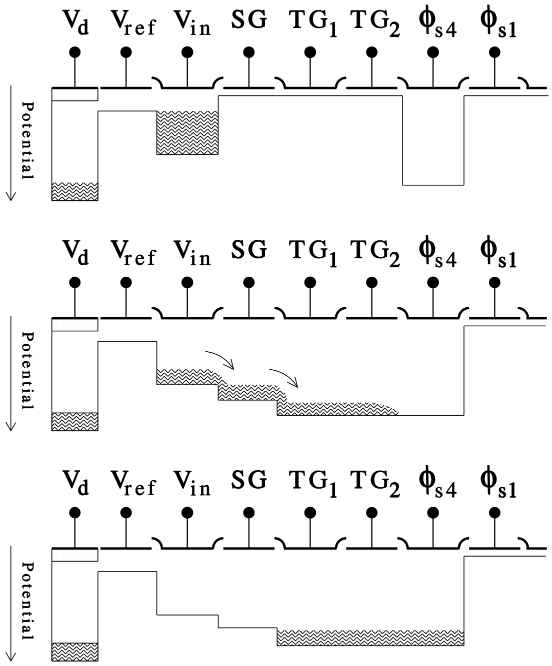


FIGURE 12-12: CCD input structure.

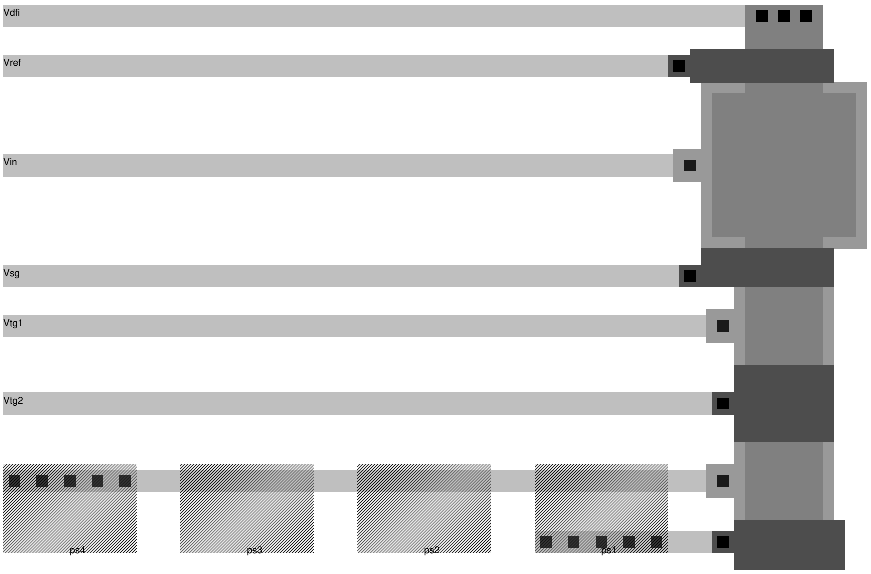


FIGURE 12-13: layout of the fill-and-spill structure.

and allowing the node gate to float.

Charge input occurs in the northwest corner of the array using the fill-and-spill structure diagrammed in Figure 12-12, with the layout given in Figure 12-13. The operation of this structure is exactly as described in Section 11.6.1, except for the names of the clock phases. Signal charge is supplied via the $n+$ diffusion at the top of the structure. The reference voltage, V_{ref}, is connected to the poly1 gate next to the diffusion, while the input voltage, V_{in} is connected to the large, $30\mu m \times 30\mu m$, poly2 gate. The area of the input gate is the same as that of the node gates within the array to ensure that enough electrons can be supplied to achieve the full range of the floating gate amplifiers.

The vertical shift register on the west boundary cells is clocked using a standard four-phase method with the signals ϕ_{s1}, ϕ_{s2}, ϕ_{s3}, and ϕ_{s4}, and is connected to the fill-and-spill structure via the two transfer gates clocked by TG_1 and TG_2, and the stop gate controlled by SG. After the fill-and-spill operation, the input charge is transferred into the shift register by clocking TG_1, TG_2, and ϕ_{s4} high and raising the stop gate voltage, SG, to an intermediate level. In order to achieve complete transfer without spilling charge back into the input gate, two conditions must be satisfied. The first is that the potentials under the input gate, stop gate, and transfer gates must be cascaded such that

$$\phi_{in} < \phi_{SG} < \phi_{TG_1} \tag{12.10}$$

while the second condition is that the maximum signal charge must be contained entirely in the potential well under the two transfer gates and the first shift register gate controlled by ϕ_{s4}. Since the combined area of these gates is $756\mu m^2$, the electron density per μm^2 of gate area for a maximum signal of 3.1×10^6 electrons is $N_e = 4100$ (electrons/μm^2). From the Orbit process parameter values given in Table 12.1 and the equations developed in Section 11.2, the voltage difference between the transfer and stop gates must be approximately 2V. With a high clock level of $V_h = 4.5$V, the maximum value of V_{SG} must therefore be ≤ 2.5V, while the voltage difference between the input and reference gates, corrected for the 0.4V poly1-poly2 potential mismatch, must be 1.4V to input the maximum number of electrons. Condition (12.10) is thus satisfied for all inputs with $V_{ref} = 0.7\text{V} \leq V_{in} + 0.4V \leq 2.5$V.

The transfer into the shift register is completed by bringing V_{SG} back to its low level of 0V and then clocking $TG_1\downarrow$ followed by $TG_2\downarrow$—$\phi_{s1}\uparrow$. The next clock sequence to be performed depends on whether or not the input value is the last pixel of its column. If so,

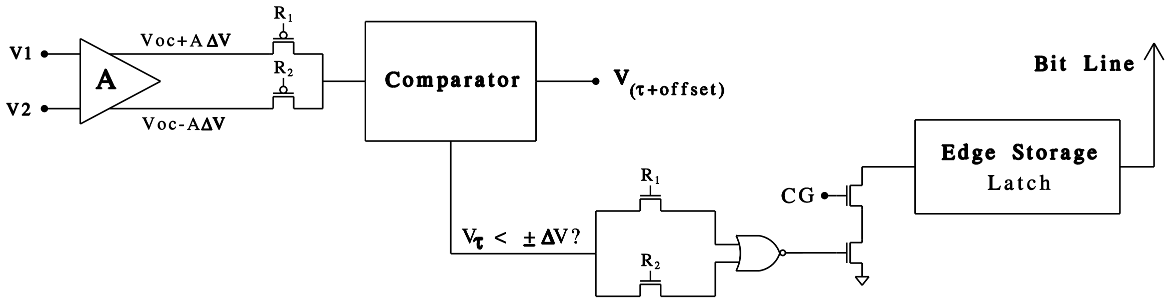


FIGURE 12-14: Multi-scale veto edge detection circuit block diagram.

bringing ϕ_{s4} low positions the signal charge for entry into the top row of the array, so that executing the horizontal transfer sequence (with ϕ_{s1} initially clocked as ϕ_1) will move the column of data in the shift register to the first column of node gates. Otherwise, before the next pixel can be loaded, the charges in the shift register must be shifted down one row by executing four times the sequence: ($\phi_{s4}\downarrow$, $\phi_{s2}\uparrow$), ($\phi_{s1}\downarrow$, $\phi_{s3}\uparrow$), ($\phi_{s2}\downarrow$, $\phi_{s4}\uparrow$), and ($\phi_{s3}\downarrow$, $\phi_{s1}\uparrow$).

12.2 Edge Detection Circuit

In detecting edges using the multi-scale veto rule, differences are computed between the values at neighboring pixels after each smoothing cycle, and an edge is signalled if and only if the magnitude of the difference is above the threshold specified for the cycle in each of the tests. The circuit implementation of the differencing, threshold, and veto operations is shown in block diagram form in Figure 12-14. The outputs of the floating gate amplifiers from two neighboring nodes are connected to the inputs of a double-ended differential amplifier with gain **A**. The two outputs of the differential amplifier are equal to $V_{oc} + A\Delta V$ and $V_{oc} - A\Delta V$, where $\Delta V = V_1 - V_2$ and V_{oc} is the common-mode output when $\Delta V = 0$. Since it is not known whether ΔV is positive or negative, the threshold test is performed by comparing both outputs to a voltage representing the threshold plus an offset to compensate for V_{oc} as well as any systematic bias in the comparator. If the threshold voltage is greater than both $+\mathbf{A}\Delta V$ and $-\mathbf{A}\Delta V$, the edge is vetoed by grounding the input to the storage latch.

Since space is limited in the unit cell, it is not practical to duplicate the comparator

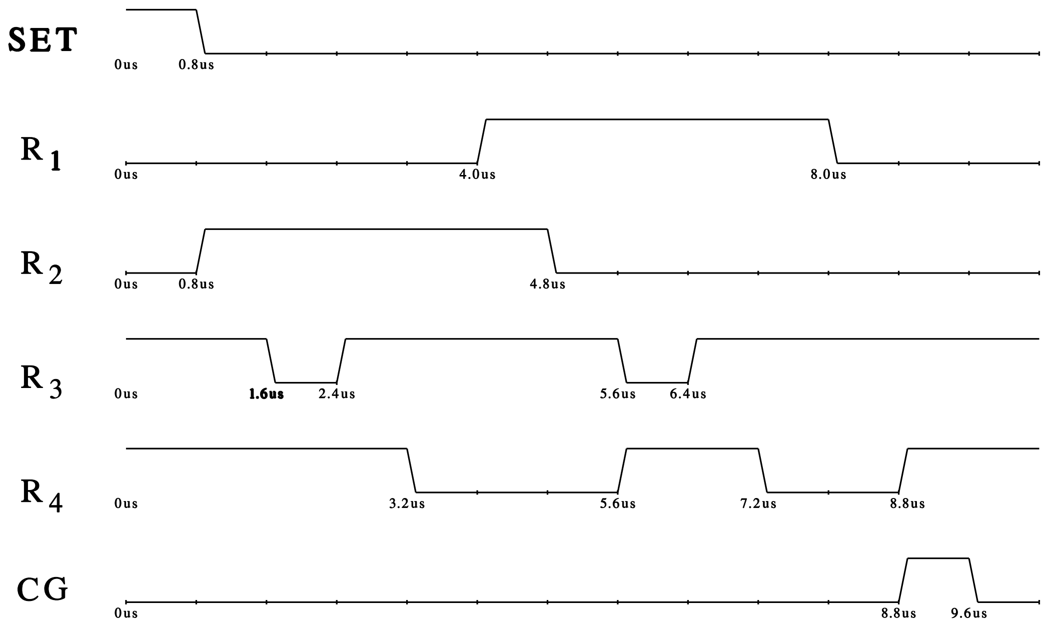


FIGURE 12-15: Clock waveforms for driving edge detection circuit.

circuit to perform both tests simultaneously. Instead, the tests are performed sequentially with a single comparator by selectively gating the differential amplifier outputs using the clock signals R_1 and R_2. The comparator output, which is high if the threshold voltage is less than the differential output, is also selectively switched to one of the inputs of the NOR circuit and is stored on the input gate capacitance when the switch is opened.

The clock waveforms, including R_1 and R_2, used in the edge detection circuit are shown in Figure 12-15. Signals R_3 and R_4 are used in the comparator circuit, which is discussed in Section 12.2.2. The signal SET is used to initialize the edge storage latch, discussed in Section 12.2.3, before starting the series of smoothing cycles and threshold tests, while the signal CG is used to gate the result of each threshold test, once it is valid, to the storage latch input.

12.2.1 Double-ended differential amplifier

The role of the differential amplifier is to generate a signal proportional to the difference in the input voltages which can be compared against a given threshold value. In order to

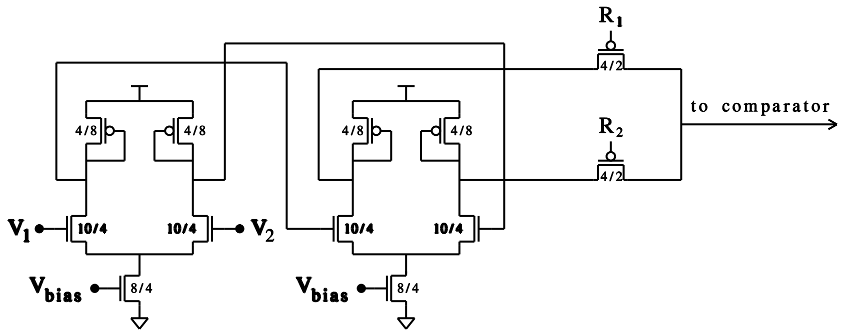


FIGURE 12-16: Differential amplifier circuit.

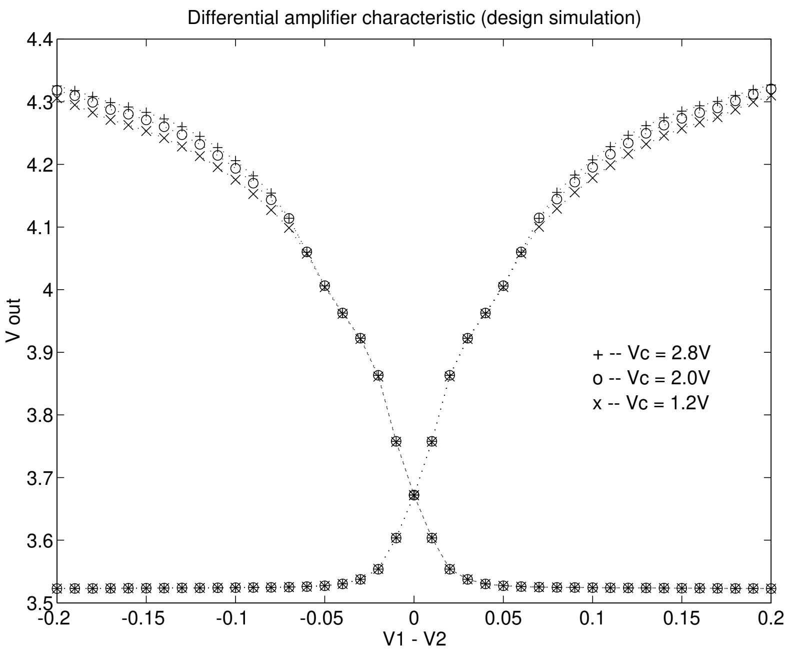


FIGURE 12-17: Simulated differential amplifier characteristic for common mode voltages $V_c = 1.2$V, 2.0V, and 2.8V ($V_{bias} = 1$V).

meet the resolution requirements discussed in Chapter 10, it is necessary for the range of measurable differences to be between 2.5% and 10% of the full scale input range. Since the floating gate amplifiers are designed for a full scale swing of 1.6V, the range of distinguishable differences must be between at least 40mV and 160mV.

If the differencing circuit is to be effective in detecting edges over the full range of input values, two more requirements must be satisfied. First, the differential amplifier must have a very high common mode rejection ratio so that a given difference ΔV corresponds to approximately the same output signal for all common mode input levels. Second, the gain, $\mathbf{A}$, of the amplifier should be large enough to magnify the minimum input difference so that it is greater than the minimum resolution of the comparator circuit. On the other hand, $\mathbf{A}$ can not be so large that the amplifier output saturates when the input difference is less than the maximum value which must be measured. These output constraints translate into requiring the amplifier gain to be between ~ 2 and ~ 15.

The circuit diagram of the differential amplifiers used in the prototype MSV processor is shown in Figure 12-16. It consists of two identical cascaded differential pairs with diode-connected p-fet loads. The common mode output and common mode rejection ratios of each pair are determined by the magnitude of the bias current and by the output resistance of the bias transistor. The small signal differential gain, $\mathbf{A}_d$, of each pair is equal to the gain of the half-circuit composed of a single n-fet input stage with a p-fet load [98]. Let

$$g_{m1} = \sqrt{2K_n \left(\frac{W}{L}\right)_1 I_D} \tag{12.11}$$

be the transconductance of the n-fet input and

$$g_{m2} = \sqrt{2K_p \left(\frac{W}{L}\right)_2 I_D} \tag{12.12}$$

be that of the p-fet load, where I_D is the drain current through both transistors, and the factors K_n and K_p are given by $K_n = \mu_n C_{ox}$ and $K_p = \mu_p C_{ox}$, with μ_n and μ_p being the respective electron and hole mobilitities. Since the output voltage is equal to the gate voltage of the p-fet, it is easily seen that

$$\mathbf{A}_d = \frac{v_{od}}{v_{id}} = \frac{g_{m1}}{g_{m2}} = \sqrt{\frac{\mu_n (W/L)_1}{\mu_p (W/L)_2}} \tag{12.13}$$

The input and load transistors in each pair were sized such that $(W/L)_1 = 10\lambda/4\lambda$ and $(W/L)_2 = 4\lambda/8\lambda$. With nominal values for the Orbit process of $K_n = 46.9\mu A/V^2$ and $K_p = 17.0\mu A/V^2$, the theoretical differential gain is thus $\mathbf{A}_d = 3.7$. The advantage of cascading the two low-gain differential amplifiers is that the combined differential and common mode gains are the products of the individual terms. We thus obtain both a high differential gain and a high common mode rejection ratio with half the input capacitance loading the floating gate amplifier outputs.

The simulated output characteristics of the differential amplifier using the Spice level 2 parameters supplied by Orbit for the 10-28-93 run are shown in Figure 12-17 for common mode inputs of 1.2V, 2.0V, and 2.8V. The gain $\mathbf{A}_d$ of each pair determined by the simulation was 1.97, with a combined gain of 7.70. The difference between these values and that predicted by equation (12.13) is due to the use of a more accurate model which includes second-order effects, such as channel geometry and threshold variations, by the HSPICE simulation program.

The fact that the two ouput voltages are symmetric about $V_{oc} = 3.67$V only for input differences less than 10mV has no impact on the threshold test as only the range of the positive difference, $V_o > V_{oc}$, is important. The output response shows good common-mode rejection for the first 70mV of input difference and only a slight dependence on the common mode level, which should not greatly affect the overall performance of the edge detector, for differences between 70mV and 200mV.

12.2.2 Comparator circuit

The comparator and dynamic NOR circuits used for the threshold tests are shown in Figure 12-18. The basis of the comparator is a standard clocked CMOS sense amplifier developed for measuring small voltage differences in memory circuits [99]. When the clock signal R_3 is low and R_4 is high, one of the gates of the back-to-back inverters is precharged to the output value from the differential amplifier, while the other is precharged to the voltage representing the threshold value. When R_3 is brought high, with R_4 still high, the sources of the two n-fets at the base of the sense amp are grounded. The gate of the transistor precharged to the higher of the two input voltages will pull more current initially, and will thus bring its drain voltage to ground more quickly, than the other transistor whose gate is precharged to the lower voltage. Since the drain of each transistor is connected to

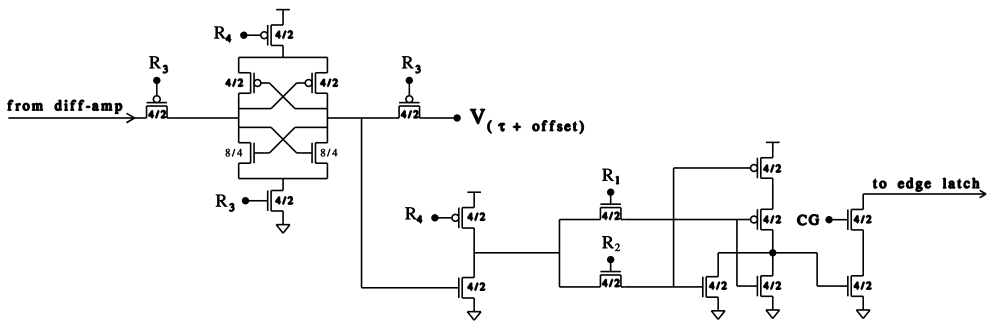


FIGURE 12-18: Sense amplifier voltage comparator for threshold tests.

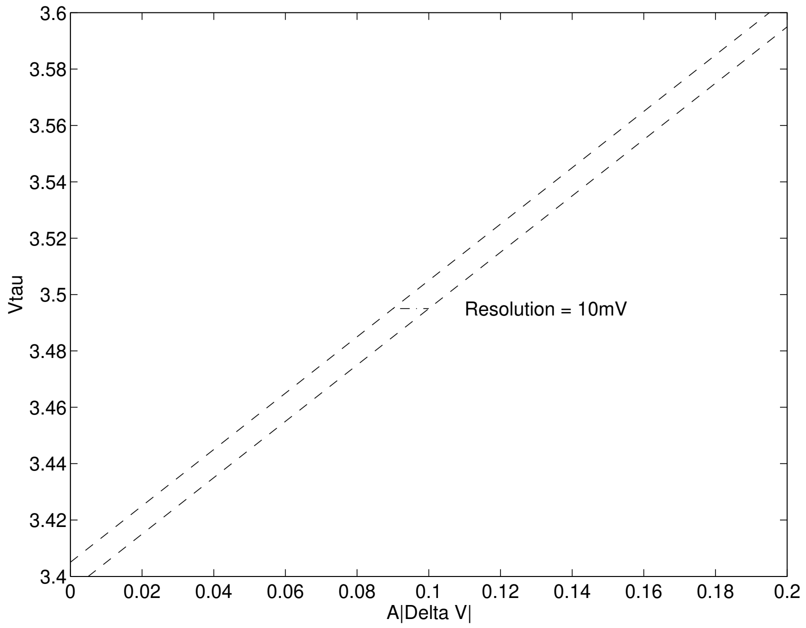


FIGURE 12-19: Sense amplifier response with resolution of $\delta = 10\text{mV}$.

the gate of the other, when one drain goes to ground it shuts off the opposing transistor, preventing its drain from discharging more. One clock tick after R_3 is brought high, and the drain-gate voltages on the transistors have settled, R_4 is brought low, connecting the sources of the two p-fets at the top of the sense amp to V_{DD}. The process is then repeated in reverse such that the side which was not completely discharged when R_3 was brought high is now brought all the way to V_{DD}.

The sense amplifier thus produces two binary ouputs, one for each input, which are high when the correponding input is greater than the other one and low when the corresponding input is less. A real sense amplifier, however, has a finite resolution due to the $\sqrt{kTC}$ noise in charging the gate capacitances[2] and thus cannot measure differences in voltages that are arbitrarily close together. The resolution, δ, is defined for a given confidence level α such that the sense amplifier will produce the correct output with probability $p > \alpha$ when the magnitude of the difference in the two inputs is greater than $\delta/2$. An example response characteristic for the present situation is illustrated in Figure 12-19 where the x-axis is plotted for $\mathbf{A}|\Delta V|$, given $V_{oc} = 3.6$V and assuming no systematic offset in the sense amplifier. In between the dashed lines, where $V_{oc} + \mathbf{A}|\Delta V| < V_\tau + \delta/2$ and $V_{oc} + \mathbf{A}|\Delta V| > V_\tau - \delta/2$, the probability that the sense amplifier output is correct is less than α. The resolution, δ, shown in the diagram as 10mV, is the horizontal distance between the two dashed lines and represents the minimum voltage difference which can be reliably measured.

The sense amplifier output connected to the V_τ input side is fed into an inverter whose output is gated to one of the inputs on the adjoining NOR circuit. The inverter, which consists of a single n-fet with a p-fet load clocked by R_4, is used to isolate the sense amplifier from the uneven capacitance of the NOR input gates. The fact that the inverter input itself creates a capacitive imbalance between the two sides of the sense amplifier is unimportant as the base level of the threshold voltage V_τ can be adjusted to compensate for the resulting offset. The important issue is that the input capacitance to the inverter is constant, while that of the NOR circuit depends on the result of the previous test. An analysis of the NOR circuit shows that the gate-source capacitances, C_{gs}, of the two p-fets are each a function of the charged state of the other transistors, as this determines whether or not there is a conducting path through the circuit during the precharge phase of the sense amplifier.

Since the inverter output is low when V_τ wins the comparison, the NOR output will be

[2] See Section 11.6.1 for a discussion of charging noise.

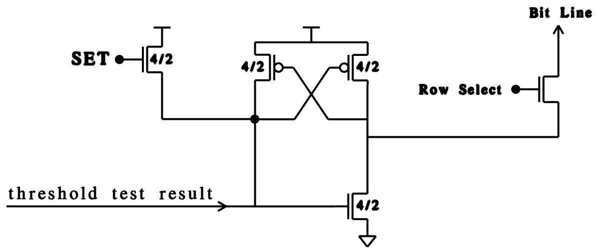


FIGURE 12-20: Edge charge storage latch.

high only if the threshold voltage, V_τ is greater than both $V_{oc} + \mathbf{A}\Delta V$ and $V_{oc} - \mathbf{A}\Delta V$. In this case the switch transistor at the bottom of the two transistor chain connected to the edge storage latch input is turned on. The second transistor, which is clocked by the signal CG, is turned on after both comparisons have been completed and the NOR output is stable. If the NOR output is high, the storage latch input is connected to ground, and the edge signal is discharged. If it is low, however, the lower switch is open, and raising CG has no effect on the state of the storage latch.

12.2.3 Edge storage

The edge storage latch, which consists of a pair of cross-coupled p-fets along with two n-fets for initializing and discharging the edge signal, is shown in Figure 12-20. At the beginning of the multi-scale veto procedure, before any threshold tests are performed, the latch is initialized by bringing the SET signal high. This action turns on the lower n-fet, bringing its drain voltage to ground, and thereby turning on the upper left p-fet which pulls the n-fet gate all the way to V_{DD}. When SET is brought low, the positive feedback of the n-p transistor combination will maintain the charged state of the latch indefinitely as long as the gate of the lower n-fet is not grounded.

If the edge is vetoed, however, *i.e.,* if the NOR output is high, the n-fet gate will be connected to ground when CG goes high. If this occurs, the upper right p-fet will be turned on and will pull the drain of the n-fet to V_{DD}, shutting off the upper left p-fet. The discharged state of the input to the CMOS inverter formed by the righthand n-p transistor combination is also stable since the SET transistor is not turned on again for the remainder of the edge detection procedure and there is no other mechanism for recharging the latch.

The edge signals for a given row are read out by bringing the 'Row Select' signal high which connects the latch output to the bit line for its column. The bit line is in turn connected to an inverting digital output pad driver to bring the signal off-chip. The current for charging and discharging the pad driver is supplied by the CMOS inverter on the right-hand side of the latch. Since this current can be supplied without affecting the state of the inverter input, the edge signals can be read out nondestructively at any point during the edge detection procedure.

Chapter 13

Edge Detector Test Results

Several fabrication runs through the Orbit CCD process were necessary to finalize and debug the design of the multi-scale veto edge detector. After several unsuccessful attempts to build a charge-based absolute value of difference circuit, similar to the one used by Keast [86], with the required resolution, the transistor-based design described in Section 12.2.1 was developed. Tinychips[1] containing test structures from the new circuit were sent out on 6-30-93, and based on the results from this run, a 32×32 array—which is the largest size that could fit on the maximum 7.9mm × 9.2mm die—along with a second full-size chip containing isolated test structures, were sent out for fabrication on 10-28-93.

Using the test system described in the next section, which was designed based on the layout of the 32×32 array processor, it was possible to obtain a more accurate characterization of the CCD performance than had been available from the much simpler test setup used with the Tinychips. Several previously unnoticed problems, such as charge trapping in the input shift register and the backspill problem discussed in Section 12.1.2, were thus discovered, and a final design, with these minor issues corrected, was sent out on 4-27-94. The die photographs of the fabricated 32×32 array and the full-size test structures chips are shown in Figures 13-1 and 13-2. The test results presented in this chapter were obtained from the chips returned from this latest run.

[1] A Tinychip is a low-cost 2.22mm × 2.25mm die size offered by MOSIS for test designs.

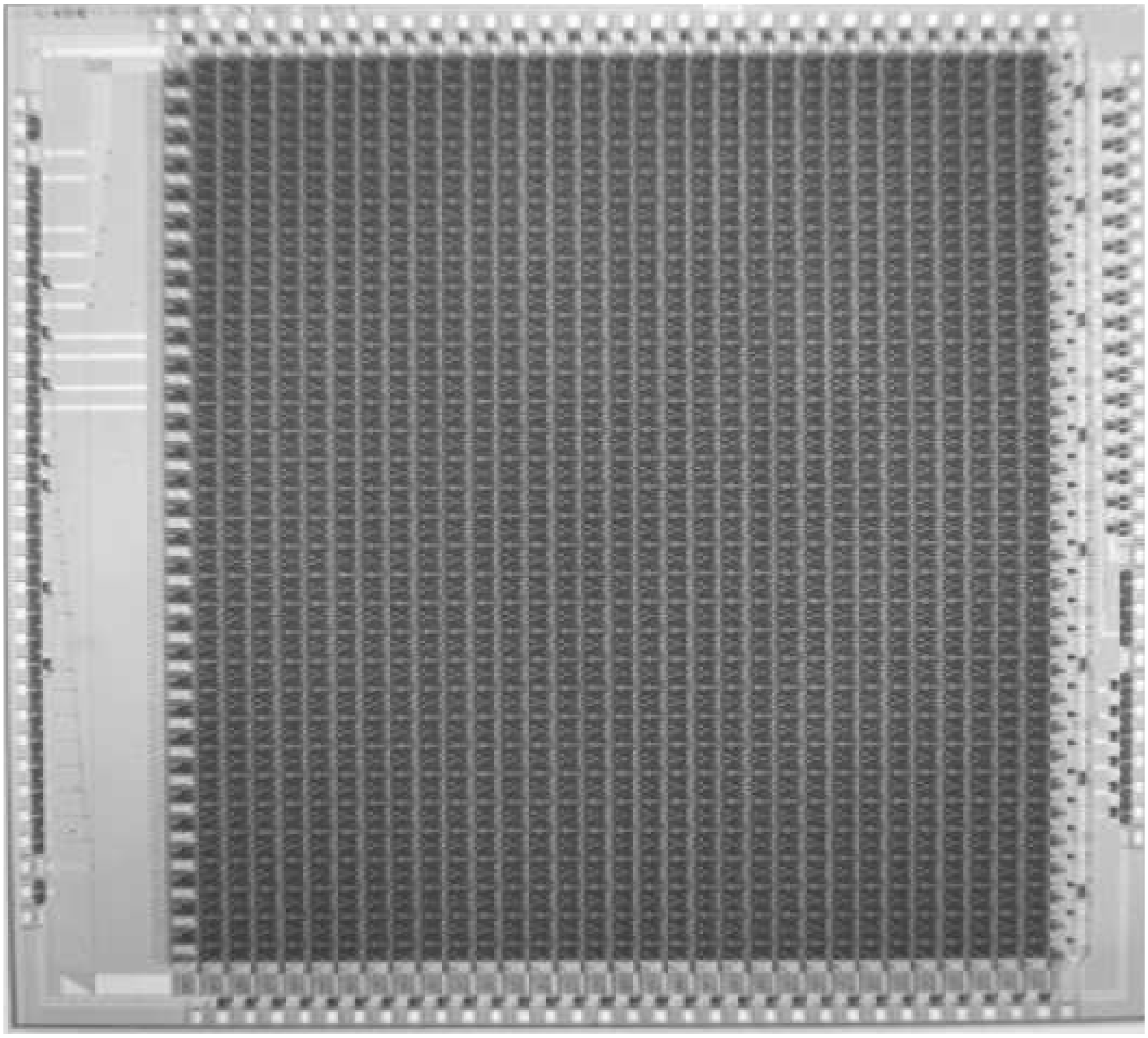


FIGURE 13-1: Die photograph of 32×32 array.

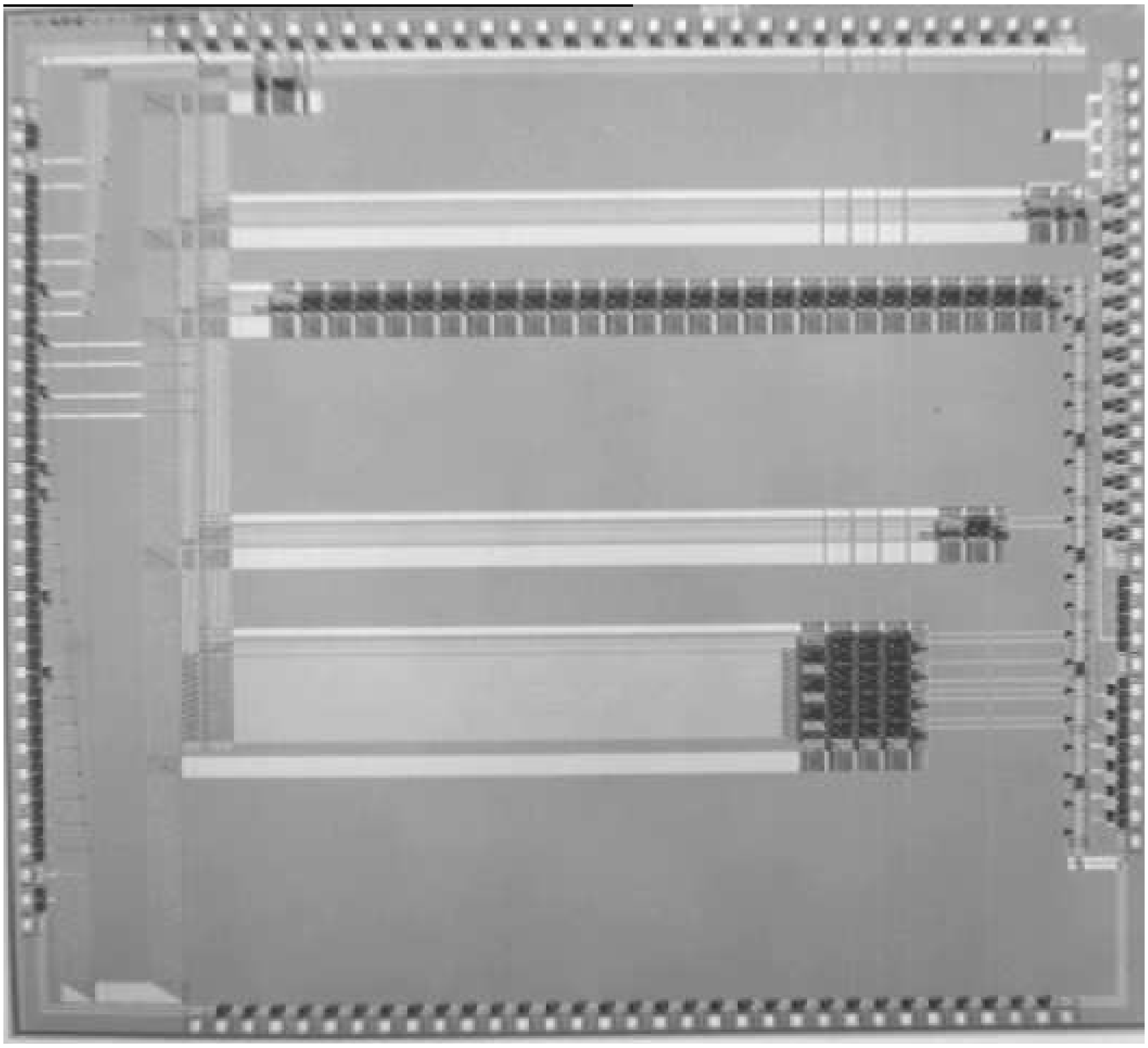


FIGURE 13-2: Die photograph of the test structures chip.

13.1 Test System

In order to test the prototype multi-scale veto chip it was necessary to supply a total of 31 programmable control signals:

- The six clock phases for driving the CCD array, ϕ_1–ϕ_6, plus the floating-gate reset signal, V_{fg}, and the control signal, ϕ_0 for the backspill prevention circuit;
- The edge detection control signals: R_1, R_2, R_3, R_4, CG, and SET;
- The input and shift register clock phases: TG_1, TG_2, ϕ_{s1}–ϕ_{s4}, and ϕ_{SG}, and the two control signals for switching the variable analog waveforms connected to the stop gate, SG, and the input diffusion, V_d; and
- Eight signals for selecting the row of edge outputs to be read and for enabling and pre-charging the output drivers.

It was also necessary to supply progammable analog waveforms for the input and reference voltages, V_{in} and V_{ref}, as well as for the edge threshold voltages, V_τ, and to read and digitize the analog voltages from the floating gate amplifiers representing the smoothed brightness values. Finally, a wide data path connected to a high-speed memory buffer was needed to store the edge outputs as they are read out.

The test system designed to perform these functions is illustrated in Figure 13-3. In order to preserve maximum flexibility and to minimize programming time, the system was built around a DELL 486 personal computer housing three commercially-made boards for facilitating the interface to the device under test (DUT). The first of these boards, manufactured by DATEL, Inc., is a 4-output power supply for driving the MSV processor and its supporting circuitry. The second, also manufactured by DATEL, Inc., is an I/O board capable of digitizing 16 independent analog inputs to 12 bits with a conversion time of 12μs per input. It can also supply 4 independent analog outputs from 12-bit digital data stored in 4 on-board registers with a settling time of 5μs to 0.05% of full scale range. The third board, made by White Mountain DSP, Inc., is a TI TMS320C40-based evaluation board, originally designed for testing applications using the 'C40 microprocessor before building a stand-alone system. In the present test system, this board turned out to be very useful due to its 32-bit bus interface and access to the 'C40 read/write control signals via the 96-pin on-board Eurocard connector. It also has a high-speed internal path to an on-board 4K

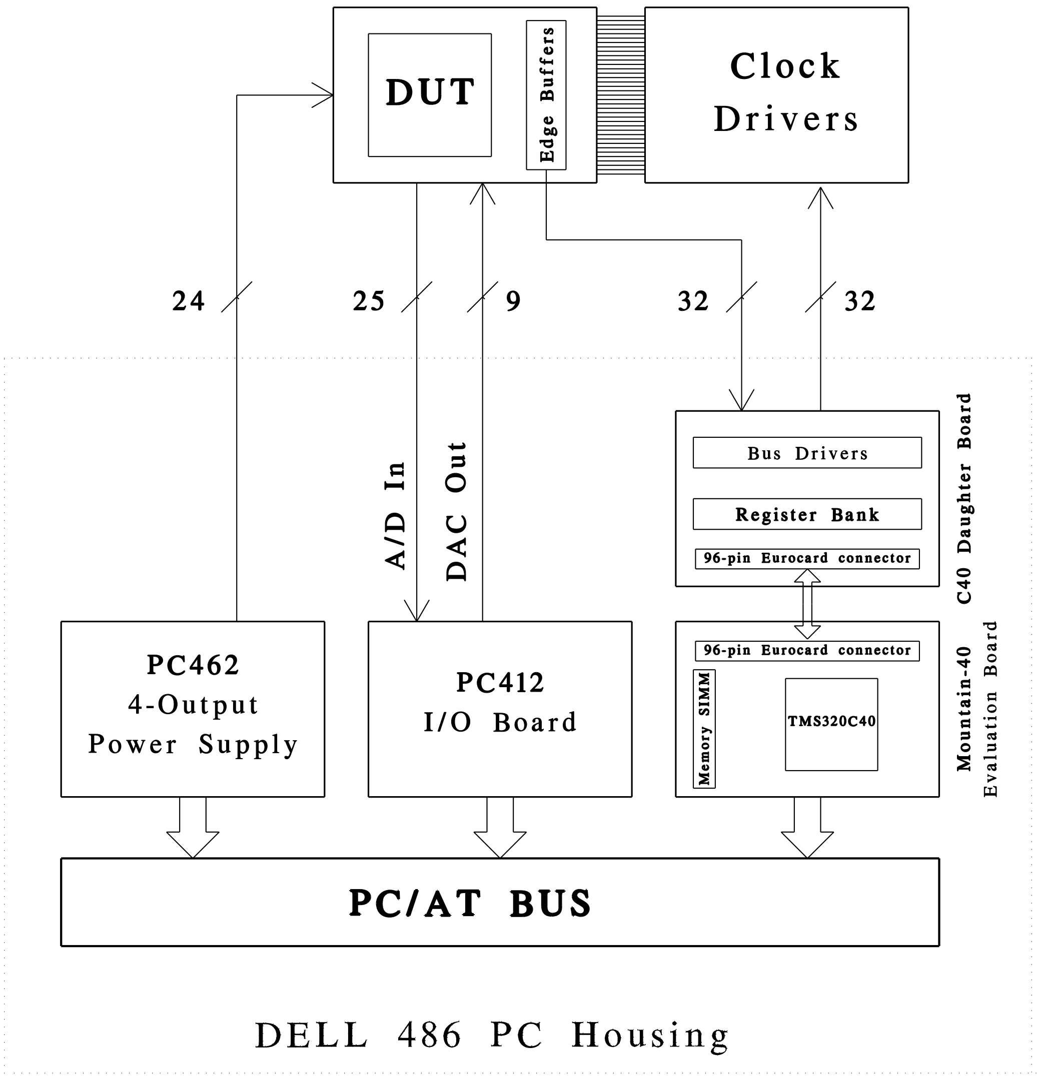


FIGURE 13-3: Test system design

dual-port SRAM which can be accessed by the PC as shared memory. All three boards communicate with the DELL over its PC/AT bus and can be programmed in a high-level language. In addition, programs for the 'C40 evaluation board can be loaded and executed independently of the DELL's 486 microprocessor so that separate programs can be executed asynchronously to drive the clock waveforms for the test system and to send and acquire data over the I/O board.

The custom-designed portions of the test system include a pair of boards outside the PC which contain the device under test, circuitry for generating and switching the constant analog bias voltages used in the processor, and the clock driver circuits. In addition, a daughter board, which mates with the 'C40 evaluation board via the 96-pin connector, was designed to manage the flow of data across the 32-bit bus. The daughter board contains a bank of registers where the values for the 31 control signals for the MSV processor are latched and a set of tri-state bus drivers used to transmit the edge outputs over the bus to the internal SRAM. One 32-bit path connects a set of tri-state buffers on the device board to the bus drivers on the daughter board. Since the prototype processor contains a 32×32 array, there are 63 edge outputs per row—32 vertical and 31 horizontal—which are output simultaneously. The external buffers are used to select one-half of the edge signals to transmit to shared memory during one read cycle.

A second 32-bit path connects the 31 register outputs and the system ground on the daughter board to the clock drivers that generate the control signal waveforms for the device under test. External drivers are necessary as the register outputs are neither clean enough nor strong enough to drive the CCD gates and the edge detection circuits directly. For proper operation, it is important to control the rise- and fall-times of the waveforms and to minimize ringing. The drivers must also supply enough current to bring the highly capacitive loads formed by the CCD gates to their final levels within the time allowed.

The clock drivers were built based on a standard circuit, shown in Figure 13-4, used in some CCD cameras and described in reference [100]. This circuit uses a National Semiconductor DS0026 chip containing two drivers which, when supplied with TTL digital inputs, will produce an inverted output capable of driving a 1000pF load at 5MHz. A 2KΩ potentiometer is used to adjust the rise- and fall-times of the clock signals to approximately 100ns, and a diode protection circuit prevents the clock waveform from ringing below the MSV chip substrate voltage, V_{SS}. This protection circuit is crucial to prevent the on-chip input protection diodes from turning on and causing charge injection into the substrate

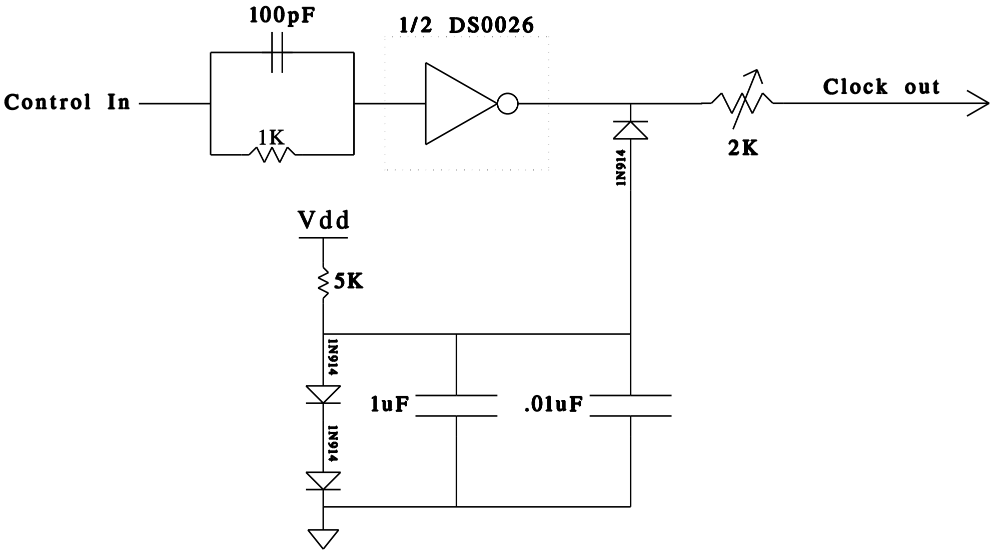


FIGURE 13-4: Clock driver circuit used in the test system

which can then be collected in the CCD array.

One disadvantage of the test system as it is designed is that, despite the 40ns instruction cycle time of the 'C40, the minimum control signal pulse width is limited to 800ns both by the programming overhead and by the propagation delays through the system. It was thus impossible to test how much faster than 625KHz that the MSV processor could be operated. Convenience and flexibility in debugging the design of the processor itself were determined to be more important, however, than optimizing the test system for speed.

13.2 Driving Signals Off-Chip

Two types of output pad drivers, one digital and one analog, were used to drive signals from the processor off-chip. The digital pad drivers, which are used with the edge signal bit lines, are simply large CMOS inverters designed to drive a 20pF load from 0V to 4V, or from 5V to 1V, in less than 10ns for a $\pm$5V step input. The input capacitance of the digital driver, as seen from the edge storage latches in the array, is approximately 2.5pF, of which only .5pF are due to the gate capacitance of the inverter and the other 2pF are due to the capacitance of the bit line itself. Simulation results indicate that the storage latch can drive

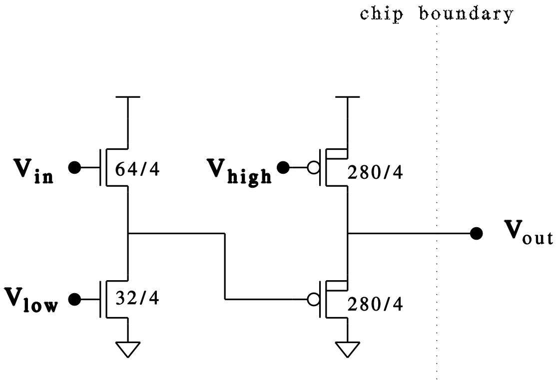


FIGURE 13-5: Analog output pad driver circuit

the inverter input from 5V to < 1V in 35ns, and from 0V to > 4V in 55ns. The rise and fall times of the edge output signals on the test chips, measured using an oscilloscope from the time that the 'Row Select' signal was brought high, were confirmed to be approximately the same as predicted by the simulation. Given the 800ns control signal pulse widths used in the test system, the edge outputs had ample time to become stable before being read into the 'C40 SRAM.

The analog pad drivers were designed to buffer the output voltages from the different isolated structures laid out on the test chip, as well as from the floating gate amplifiers on the east boundary of the array which were used to output the smoothed image data. Since the original voltages could only be recovered by correcting the measured signals for the pad driver response, it was very important to achieve good matching between the different drivers in order to accurately measure the responses of the on-chip structures.

The analog pad driver circuit, which is shown in Figure 13-5, consists of a pair of cascaded n- and p-type source followers. To achieve good matching, the transistors were drawn with large W/L ratios to minimize the percentage effects of process variations. Since the on-chip structures generally pull very little current, it was important to minimize the additional load capacitance due to the drivers. The total gate capacitance of the first stage is approximately 230fF, which is almost 7 times less than the 1.5pF capacitance of the second stage that drives the output load. The metal2 lines leading to the pad drivers, which are much shorter than the edge signal bit lines that run across the chip, have an average capacitance of approximately 120fF, giving a total load of about 350fF.

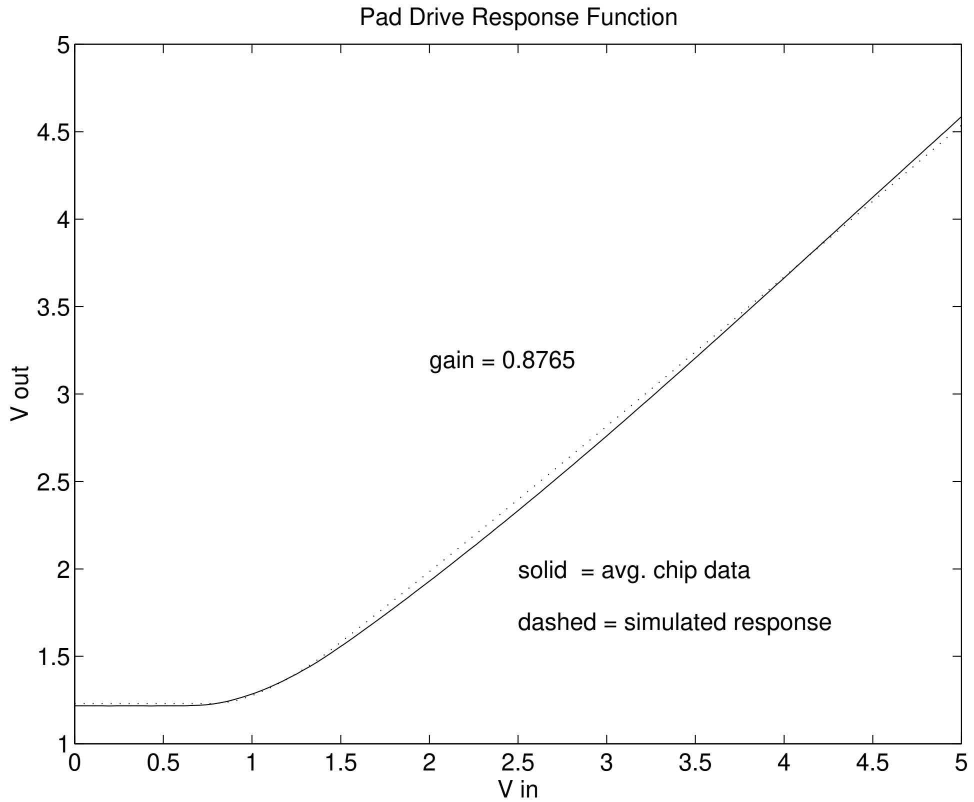


FIGURE 13-6: Analog output pad driver characteristic – simulated vs. actual (V_{low} = 1.2V, V_{high} = 3.8V).

The input stage was chosen as an n-type device in order to sense the full range of output voltages from the various on-chip structures, which are between 1.5V and 5V. Since the voltage levels are shifted down by the first stage, the second stage could be built with a higher gain p-type device laid out with separate wells for each transistor to eliminate the backgate effect. The total power dissipation of each driver is approximately 390μW and is the highest of any structure on the chip. Separate V_{DD} and GND rails were thus drawn so that the relatively large currents pulled by the drivers would not affect the circuits on the rest of the chip and also so that the actual power dissipation could be measured independently.

Figure 13-6 shows the results of test measurements from 24 different pad drivers on 12 different chips compared with the performance predicted by simulation. The solid line represents the average output from all 24 drivers. The standard deviation of the individual responses from the average curve is 8mV with a maximum absolute variation of 17.6mV. The average measured gain is 0.877, which is slightly better than the predicted value of 0.848.

13.3 CCD Tests

Correct operation of the MSV processor depends critically on the device characteristics of the CCD structures used in the array. The parameters which most affect signal processing ability are the magnitudes of the channel potentials, the amount of dark current, and the charge transfer efficiency. Isolated test structures were laid out to measure each of these parameters, as well as to measure the the input/output characteristics of the fill-and-spill and floating-gate amplifier structures used to interface with the array.

13.3.1 Channel potential measurements

To measure the maximum potential, ϕ_{max}, in the buried channel as a function of the applied gate voltage, a separate structure was laid out on the test chip containing two CCD 'transistors' formed by placing $n+$ diffusions on either side of an isolated polysilicon gate covering a segment of buried channel. Two such transistors were needed for each of the two polysilicon layers. The four diffusions were connected via metal contacts to separate unprotected and unbuffered pins so that their voltages could be measured directly.

To measure the potentials, the 'source-follower' method, described by Taylor and Tasch in [101], was used, as this approach was the simplest to apply given the test system setup. In this method, the substrate is grounded and a voltage is applied to the gate of each CCD transistor. The diffusions on one side of the transistors, acting as the drains, are set to a high voltage, while the source diffusions are connected to ground via a high impedance load. In this situation, the transistors are at the threshold of conduction so that the source voltage is equal to ϕ_{max}. Due to the potential differences across the metal contacts and the n–$n+$ and p–$p+$ junctions, the measured voltage is $V_S - V_{bi}$, where V_{bi} is the built-in voltage across the n-p junction at the channel-substrate interface. The channel potentials are thus recovered by adding 0.8V, which is the value given by Orbit for V_{bi}, to the measured values.

The ϕ_{max}–V_g curves obtained for the chips from the 4-27-94 and the 10-28-93 runs are

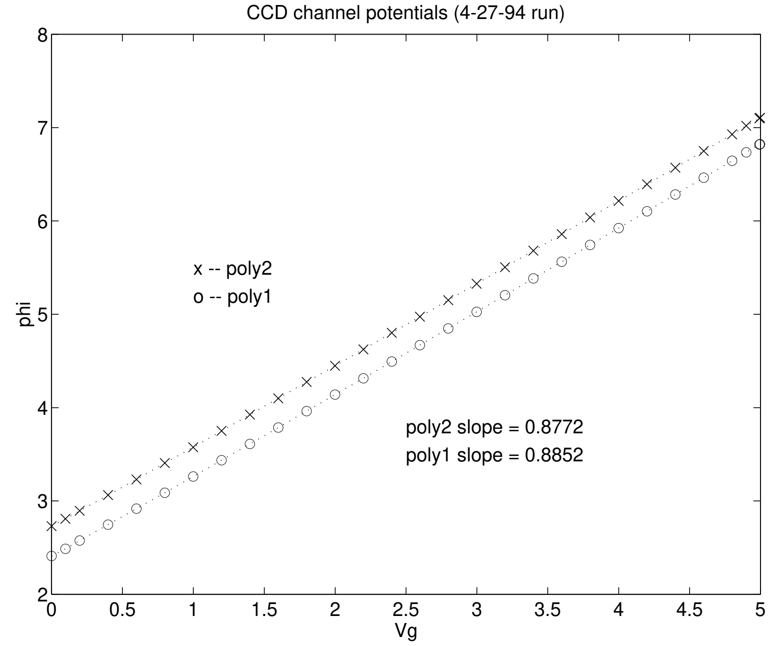


FIGURE 13-7: CCD channel potentials (4-27-94 run).

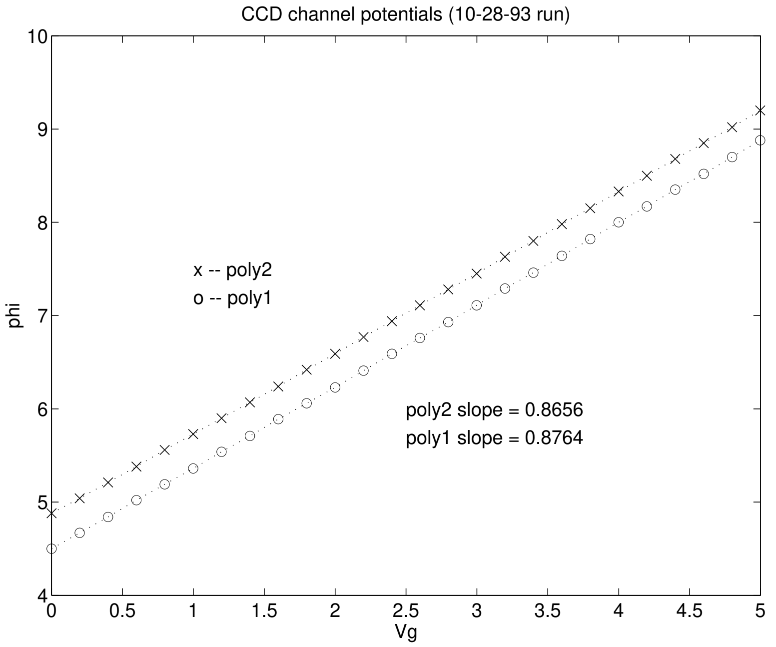


FIGURE 13-8: CCD channel potentials (10-28-93 run).

plotted in Figures 13-7 and 13-8. A process change was made by Orbit for the 4-27-94 run which had the effect of lowering the potentials by approximately 2V from those of the earlier run. This decrease does not greatly affect the results of the calculations in Sections 12.1.1 and 12.1.3 used to determine the sizes of the input and floating-gate structures, as these depend mostly on the differences in the channel potentials at different gate voltages rather than on their actual value. It does, however, affect the charge-carrying capacity of the CCD gates, and therefore the signal-to-noise ratio of the devices.

As can be seen from the diagrams, there is little change in the slopes of the ϕ_{max}–V_g curves between the two runs. The less than unity slope of .885 for poly1 and .877 for poly2 is due to the capacitive divider between C_{eff} and C_{d2} explained in Sections 11.2 and 11.6.2. Since

$$\frac{\Delta\phi_{max}}{\Delta V_g} \approx \frac{1/C_{d2}}{1/C_{d2} + 1/C_{eff}} \tag{13.1}$$

we have (for poly1)

$$\frac{C_{d2}}{C_{eff}} \approx .13 \tag{13.2}$$

which is less than half the value calculated from the estimated design parameters given in Table 12.2.

13.3.2 Input and output

The fill-and-spill and floating-gate amplifier structures were tested jointly by combining both devices in a single 1-D gate array. A separate test structure containing only the source-follower buffer used in the floating-gate amplifier was also laid out on the test chip so that it could be characterized independently.

The measured source-follower response from one chip is plotted in Figure 13-9, with the dashed line indicating the measured voltages, and the solid line representing the data corrected for the analog pad driver response. The average gain measured from twelve devices on twelve different chips was 0.895, which is considerably higher than the value of 0.800 predicted by the simulation shown in Figure 12-5. This discrepancy is due to the smaller actual values of the process parameters $\gamma = \sqrt{2q\epsilon_{Si}N_A}/C_{ox}$ and λ, the channel length modulation parameter, both of which determine the magnitude of g_{mb}, from those used in the simulation. The values given for the 4-27-94 run were $\gamma = 0.4493$ and $\lambda = 0.0304$, as opposed to the values from the 10-28-93 run given as $\gamma = 0.4977$ and $\lambda = 0.0318$. Matching

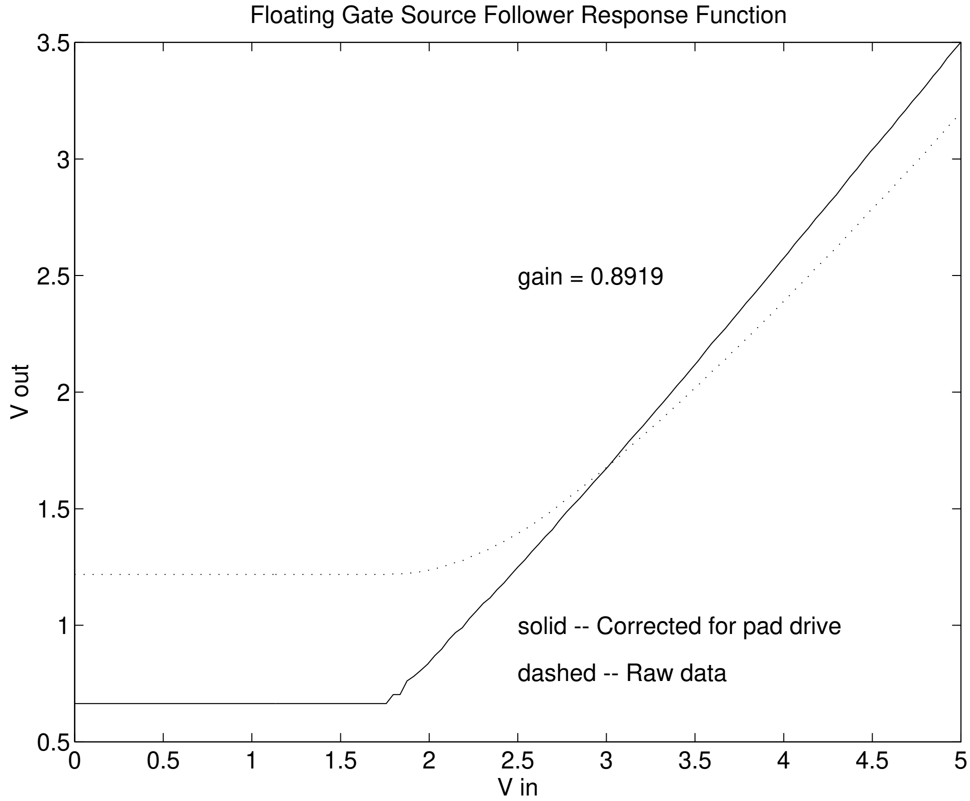


FIGURE 13-9: Measured source follower characteristic for floating gate amplifier (V_{bias} = 1V).

between the different source-followers was relatively good, with an overall standard deviation of 14.5mV from the average output values, and a maximum deviation of 37mV.

The results of the combined fill-and-spill and floating-gate amplifier structures from twelve different chips are shown by the individual points in Figure 13-10, with the solid line representing the average output. The values are plotted against the difference in the voltages applied to the input and reference gates. Since these gates are in different levels of poly (see Figure 12-13 for the layout of the fill-and-spill structure), the plot begins for $V_{in} - V_{ref} = -0.3$V, which is approximately the magnitude of the potential mismatch between the two levels.

The standard deviation of the different outputs from the average is 31mV with a maximum deviation of 121mV. It should be noted that the differences in the outputs are a combination of the mismatches between the source-follower buffers and the variations in

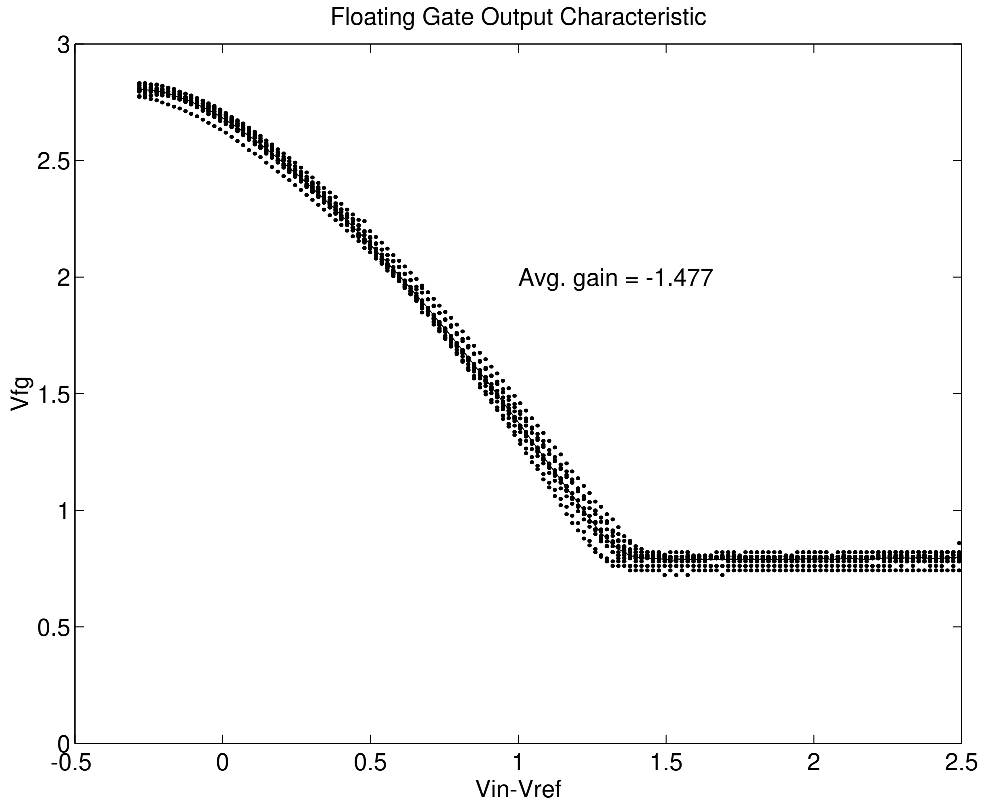


FIGURE 13-10: Measured floating gate amplifier output vs. fill-and-spill input, $V_{in} - V_{ref}$.

the channel potentials of the fill-and-spill input devices. Since there is only one input structure for each processor, differences in the floating-gate amplifier responses within the same array should be due only to mismatches in the source-followers.

The total output swing of the floating-gate amplifiers is 2V, from a maximum of 2.8V for $V_{in} - V_{ref} = -0.3$V to the minimum value of 0.8V at $V_{in} - V_{ref} = 1.4$V. The average slope of the response curve over this range is -1.48. Correcting for the source-follower gain of 0.895 thus gives the change in the floating gate voltage per unit change in the applied signal voltage as

$$\frac{\Delta V_g}{\Delta V_{sig}} \approx -1.65 \tag{13.3}$$

where $V_{sig} = V_{in} - V_{ref} + \delta_{p_1-p_2}$, with $\delta_{p_1-p_2}$ representing the poly1-poly2 mismatch.

From equations (11.28) and (11.53), we have

$$\Delta V_g = \frac{Q_{sig}}{C_{load}} \left(\frac{1/C_{d2}}{1/C_{d2} + 1/C_{eff_{p1}} + 1/C_{load}} \right) \tag{13.4}$$

and

$$Q_{sig} = -C_{eff_{p2}} V_{sig} \tag{13.5}$$

where the distinction has been made between the effective capacitances of the poly1 and poly2 structures. Combining these equations gives

$$\frac{\Delta V_g}{\Delta V_{sig}} = -\frac{C_{eff_{p2}}}{C_{load}} \left(\frac{1/C_{d2}}{1/C_{d2} + 1/C_{eff_{p1}} + 1/C_{load}} \right) \tag{13.6}$$

Using the value of $C_{d2}/C_{eff_{p1}} = .13$ computed from the channel potential measurements, equations (13.3) and (13.6) are consistent with

$$\frac{C_{eff_{p2}}}{C_{load}} \approx 2 \tag{13.7}$$

If we take the value of $9.2 \times 10^{-17}\mathrm{F}/\mu m^2$ computed for C_{load} in equation (12.5), we thus have $C_{eff_{p2}} = 1.8 \times 10^{-16}\mathrm{F}/\mu m^2$. With $\delta_{p1-p2} = 0.3\mathrm{V}$ and 1.4V being the maximum value of $V_{in} - V_{ref}$, we find

$$N_{e,max} = -\frac{Q_{sig,max}}{q} = 1912\ electrons/\mu m^2 \tag{13.8}$$

The input gate area being $900\mu m^2$, the maximum number of signal electrons is approximately 1.7×10^6.

13.3.3 Dark current measurements

In order to use CCDs for signal processing, all computations must be completed in less time than it takes for dark current to appreciably affect the signals. Dark current levels for the test chips were measured using the floating-gate amplifier structure by initializing the node gate to its high voltage and then allowing it to float without introducing charge via the gate array. Since dark current is the only source of charge into the potential well under the floating gate, the magnitude of the current can be estimated by measuring the change

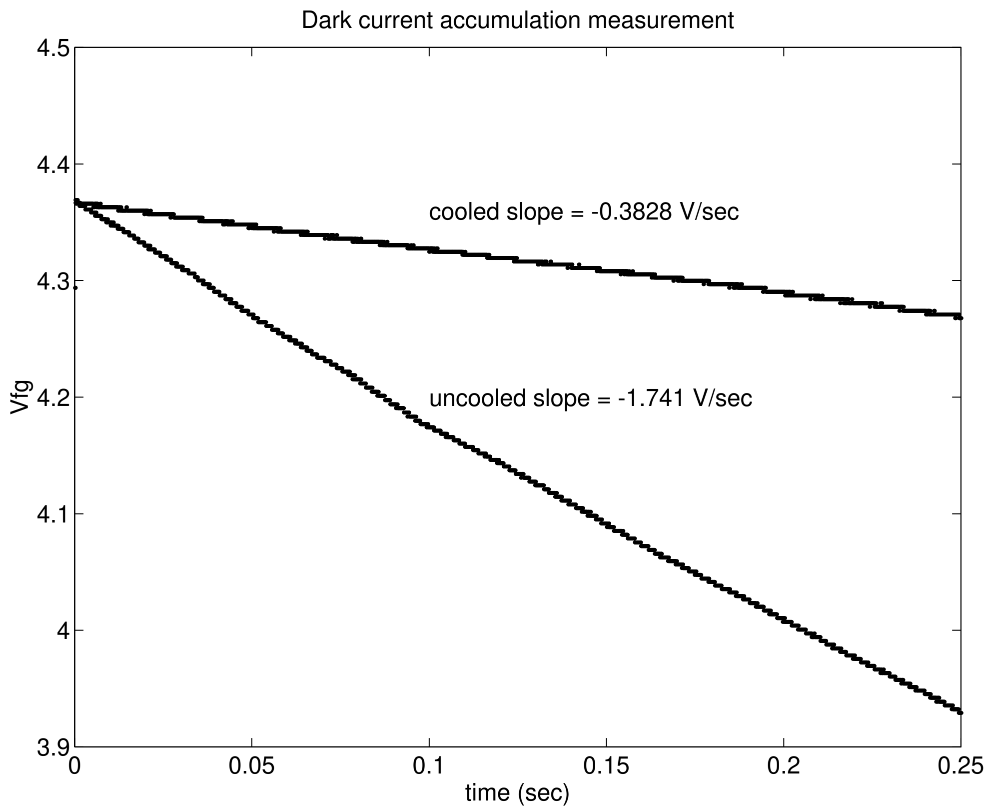


FIGURE 13-11: Dark current accumulation for cooled and uncooled chips.

in the gate voltage over time.

Figure 13-11 shows the results of two tests, one with the chip at room temperature and the other with the chip cooled to approximately 0°C. The output voltages were sampled at 125μs intervals for 0.25 seconds. The values plotted are the computed gate voltages after correcting for the responses of both the source-follower on the floating gate amplifier and the output pad driver. From equation (13.4), the slopes of the curves are related to the dark current density, J_D by

$$\frac{\Delta V_g}{\Delta t} = \frac{J_D}{C_{load}} \left(\frac{1/C_{d2}}{1/C_{d2} + 1/C_{eff_{p1}} + 1/C_{load}} \right) \qquad (13.9)$$

Using the previously estimated values for C_{load} and $C_{d2}/C_{eff_{p1}}$, we thus have

$$J_D = \frac{\Delta V_g}{\Delta t} \times 1.1 \times 10^{-8} A/cm^2 \tag{13.10}$$

For the chips at room temperature with a slope of $\Delta V_g/\Delta t = -1.74$V/sec, J_D is approximately $-19nA/cm^2$, while for the cooled chips, with $\Delta V_g/\Delta t = -0.38$V/sec, J_D is only $-4.2nA/cm^2$. It should be noted that the uncooled dark current levels fluctuate tremendously as both room and chip operating temperatures vary. The uncooled values, however, were never measured at less than -1.4V/sec and were often as high as -2.5V/sec. These values are high with respect to commercial grade CCDs, but are not unusual for the Orbit process which is not optimized for dark current[2]. For tests involving total processing times of more than 5ms, it was thus necessary to cool the chips to achieve proper operation.

13.3.4 Transfer efficiency

In order to measure charge transfer efficiency, a special structure composed of a chain of 27 1-D unit cells together with a boundary cell containing a fill-and-spill input device was laid out on the test chip. The 1-D cells are simply truncated and rearranged versions of 2-D unit cells which are missing the vertical sections. The layout of these cells is such that, when placed end-to-end, a linear array is formed which is identical in operation to the 2-D array, with the exception that there is no vertical movement of charge. Since each cell contains 12 gates, the signal charge in the test array is transferred through 336 gates from the end of the fill-and-spill structure to the final node gate, whose floating-gate amplifier output is connected to one of the analog pad drivers.

As explained in Section 11.4, charge transfer efficiency can be measured from the ratio of the size of the charge packet under the Nth transfer gate to the original packet size at gate 0. From equation (11.31), we find that

$$\frac{Q_N}{Q_0} = \epsilon^N \tag{13.11}$$

To measure transfer efficiency with the given test structure, the array was initially flushed by executing the horizontal transfer sequence many times (> 100) with no input charge

[2]Private communication with Paul Suni of Orbit Semiconductor, Inc.

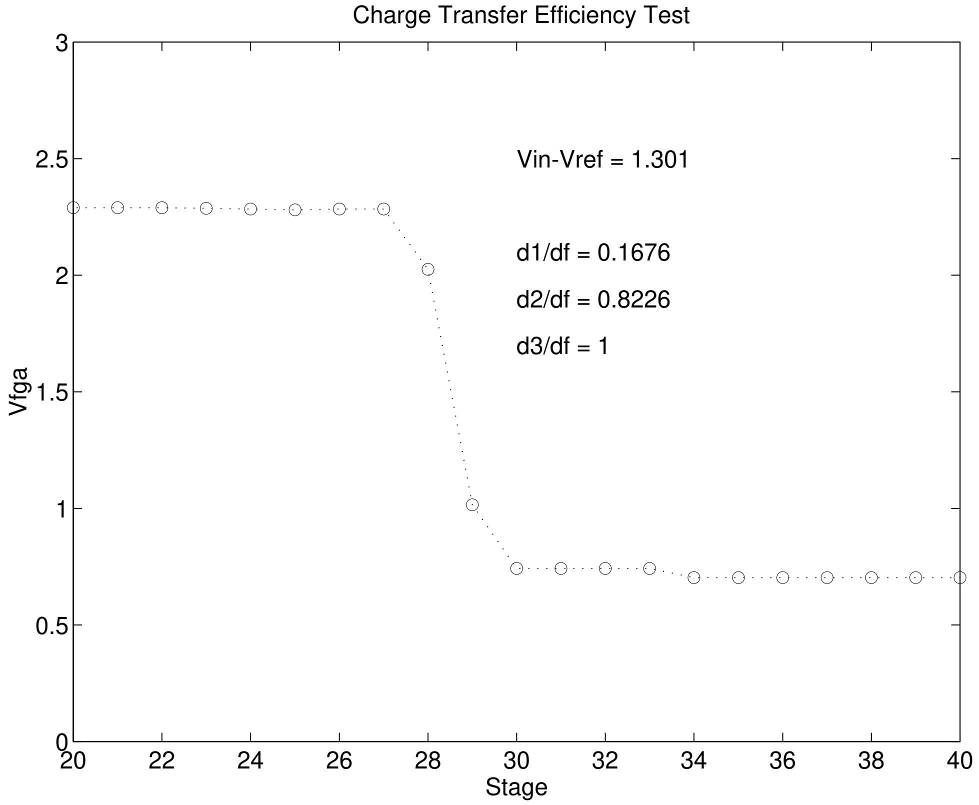


FIGURE 13-12: Charge transfer efficiency measurement.

introduced through the fill-and-spill device. The control voltages, V_{ref} and V_{in}, were then set to give a maximum size charge packet, and the fill-and-spill and transfer sequences were executed repeatedly to move a series of equal size packets into the array. By measuring the floating-gate amplifier output of the final node at the end of each transfer stage, the size of the first charge packet in the series which arrives at the node can be compared with the size of the following packets. After several transfer sequences, steady-state conditions will be reached—in which as much charge is lost to the trailing packets as is picked up from the previous ones—and the size of the packets will be approximately that of the original signal.

Figure 13-12 shows the results of one test which is typical of the behavior observed in all of the chips. Approximating the floating-gate response as being linear with ΔQ, the ratio of Q_N/Q_0 is given by the ratio of the initial voltage drop from the zero-signal level at stage 28, when the first packet arrives, to the final voltage drop measured several stages later. As seen in the plot, this value is computed as $\Delta V_i/\Delta V_f = .17$, and from equation (13.11) with $N = 336$, we find the per-gate transfer efficiency to be $\epsilon = 0.995$. Using $N = 28$ in equation (13.11), we can also compute ϵ_{stage}, the transfer efficiency per stage as $\epsilon_{stage} = 0.939$.

The measured transfer efficiencies for the Orbit CCD process are very low compared with desired values of 0.99999 or better required for large gate arrays. As seen in Section 13.5, the low CTE of these devices does affect the results of the 32×32 prototype array and limits our ability to characterize very large structures.

13.4 Differencing and Threshold Test Circuits

Three isolated structures were laid out to test the operation of the edge processing circuits contained in the unit cells. The two major components, the differential amplifier and the voltage comparator, were set up to be tested individually, while a third structure contained the complete absolute-value-of-difference and threshold test circuit.

13.4.1 Differential amplifier

Differential amplifier responses were measured for twelve different test structures under the same conditions ($V_{bias} = 1$V, and common mode values of $V_c = 1.2$V, 2.0V, and 2.8V) used in the simulation described in Section 12.2.1. The results from one of the test circuits are shown in Figure 13-13. Over all, the basic shape of the response curve is very similar to that predicted by the simulation, with good common mode rejection and differential gain, $\mathbf{A}_d$, of 7.2. However, for a given input difference, there are significant variations in the output voltages of the twelve circuits due to variations both in the amplifier offset voltages and in the values of the common mode outputs, V_{oc}.

The average value of V_{oc} among the twelve circuits was 3.43V, with a standard deviation of 38mV. Referring to Figure 12-16, the common mode output voltage is determined by the amount of drain current flowing through each side of the second-stage amplifier when it is in its balanced state and by the effective resistance of the diode-connected p-fet load. Given that the drain current in the balanced state is one-half of the current through the lower

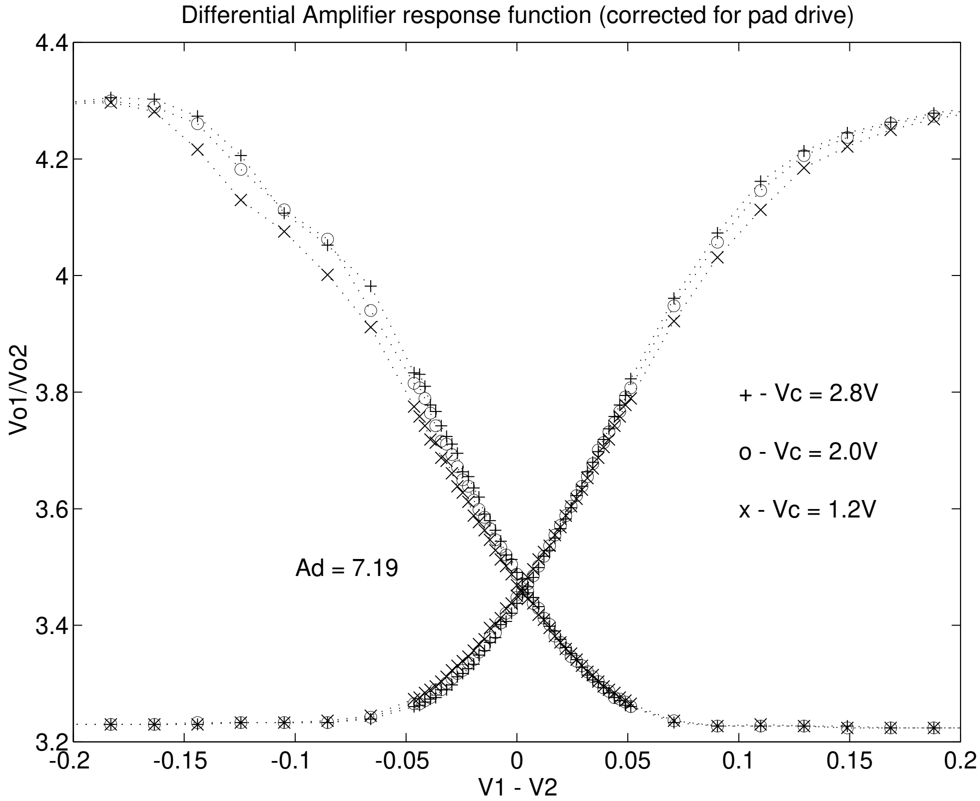


FIGURE 13-13: Measured differential amplifier characteristic for common mode voltages $V_c = 1.2$V, 2.0V, and 2.8V with $V_{bias} = 1$V.

bias transistor, it can easily be seen, using the notation of Section 12.2.1, that

$$V_{oc} = V_{DD} + |V_{tp}| - (V_{bias} - V_{tn})\sqrt{\frac{K_n(W/L)_{bias}}{2K_p(W/L)_2}} \tag{13.12}$$

where V_{tn} and V_{tp} are the n- and p-fet threshold voltages. Since the quantity under the radical is much larger than one, variations in V_{oc} thus depend strongly on variations in the threshold voltages of the bias transistors.

The offset voltage of a differential pair, defined as the difference in the inputs $V_1 - V_2$ required to make the output voltages equal, is, from [98],

$$V_{OS} = \Delta V_t + \sqrt{\frac{I_D}{K_n(W/L)}}\left[g_{mp}\Delta\left(\frac{1}{g_{mp}}\right) + \left(\frac{\Delta(W/L)}{W/L}\right)\right] \tag{13.13}$$

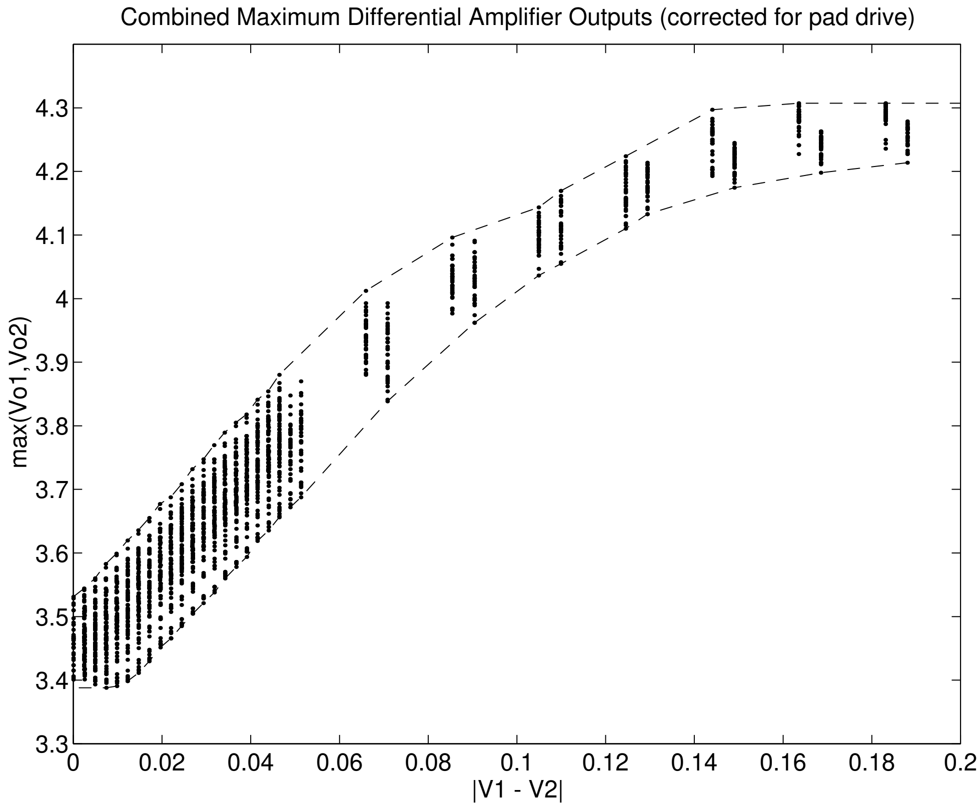


FIGURE 13-14: Combined maximum differential amplifier outputs vs. $|V_1 - V_2|$ for common mode voltages V_c = 1.2V, 2.0V, and 2.8V with V_{bias} = 1V.

where ΔV_t is the difference in threshold voltages of the two input transistors, I_D is the average drain current through each side, (W/L) is the average size ratio of the input transistors, and g_{m_p} is the average transconductance of the two p-fet loads. The quantities $\Delta(1/g_{m_p})$ and $\Delta(W/L)$ represent the differences in these parameters between the two sides of the differential pair.

The average offset voltage of the twelve circuits was +4.5mV, with a standard deviation of 3.3mV. Given the very low bias current ($\sim 400nA$ with V_{bias} = 1V) and the large W/L ratios used in the transistors, the variations in both V_{OS} and V_{oc} can be attributed almost entirely to differences in the threshold voltages.

The effect of these variations on the overall operation of the MSV processor can be judged from the plot, shown in Figure 13-14, of the maximum amplifier outputs for all

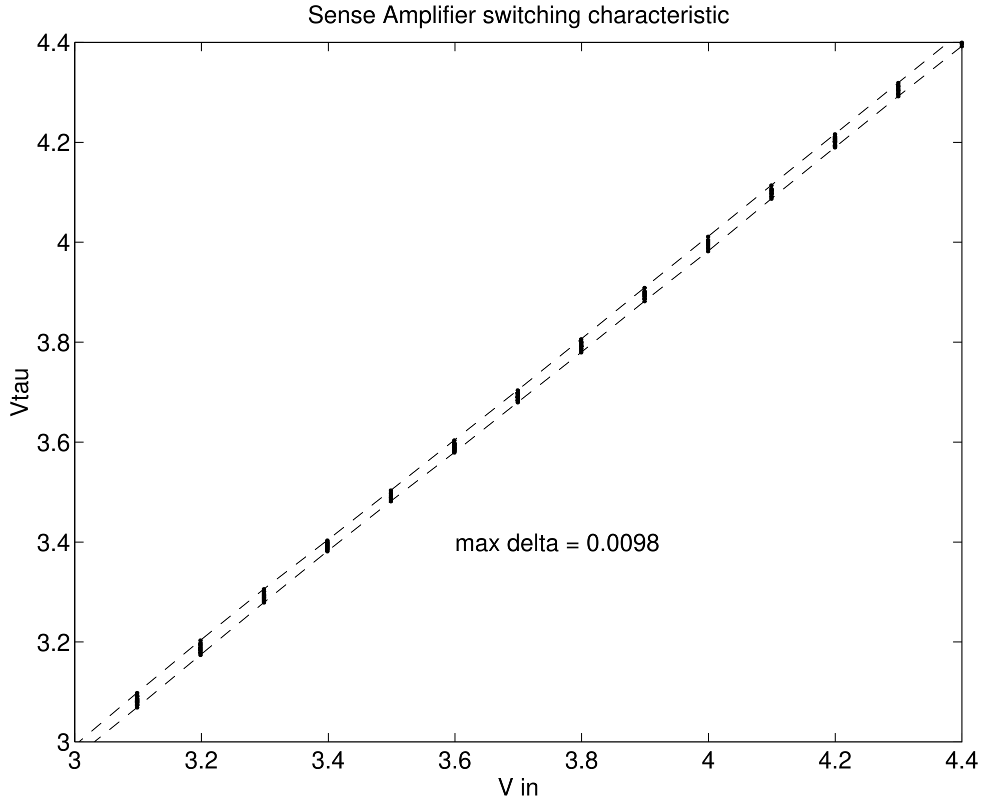


FIGURE 13-15: Measured sense amplifier switching characteristic.

twelve circuits at each of the three common mode values against the absolute input difference $|V_1 - V_2|$. This plot can be considered as a measurement of the resolution of the differential amplifier and gives a lower bound on the resolution of the complete edge detection circuit. For $|V_1 - V_2| < .1$V, the horizontal distance between the dashed lines, which approximates the smallest measurable difference in the inputs, is roughly 28mV. For absolute differences above .1V, however, the amplifier saturates rapidly so that it is impossible to distinguish between differences that are greater than .14V. When considered as a percentage of the full scale input range, if this is set to 1.4V, the 28mV to 140mV range of measurable differences for the actual circuits is in fact better than the targeted swing of 2.5% to 10% of FSR.

13.4.2 Sense amplifier voltage comparator

The sense amplifier switching characteristic was measured by setting one input, V_{DA}, to a fixed value and varying the other input, V_τ until finding the point at which the comparator output changed. Since, as explained in Section 12.2.2, the output is random when the two inputs are closer than $\delta/2$, where δ is the resolution of the sense amplifier, the actual measurement used to determine the switching point was the number of times out of 100 that the comparator output was high. Two values, $V_{\tau,low}$ and $V_{\tau,high}$, were thus measured for each value of V_{DA}, with the first being the highest value for which the output was high $\leq 1/100$ times, and the second being the lowest value for which the output was high $\geq 99/100$ times.

The cumulative results from twelve different comparator circuits are plotted in Figure 13-15, with the dashed lines representing the envelope of the high and low threshold voltages from all twelve circuits. The maximum horizontal distance between these lines, which is a measure of the sense amplifier resolution, is 9.8mV. Given that this value is divided by the differential gain, $\mathbf{A}_d$, when the input is referred back to the that of the differential amplifier, the 10mV resolution of the comparator should have a negligible effect on the overall edge detector circuit performance.

It should be noted from the plot that there is a slight offset of approximately 15mV between the threshold and input voltages when the comparator switches. This offset is caused by the capacitive imbalance created by connecting an inverter input to the V_τ side. Since the offset is constant, however, it can be compensated for in setting the value of the threshold voltage.

13.4.3 Combined edge veto test circuit

The complete multi-scale veto edge detection circuit, starting from the inputs to the source-follower buffers of the floating-gate amplifiers and ending with the output of the edge storage latch, was laid out as an individual test structure to evaluate the combined performance of all of its elements. The circuit was tested by providing a fixed voltage difference to the source-follower inputs and determining the maximum value of V_τ for which the input difference could be considered as an edge.

Figures 13-16 and 13-17 show the results from one circuit, plotted against both $V_1 - V_2$ and $|V_1 - V_2|$, for input common mode values of 4.2V, 3.3V, and 2.5V, corresponding roughly to the high, medium, and low floating-gate voltages. The common mode variation in the results reflects that of the differential amplifier, while the offset in the circuit is the

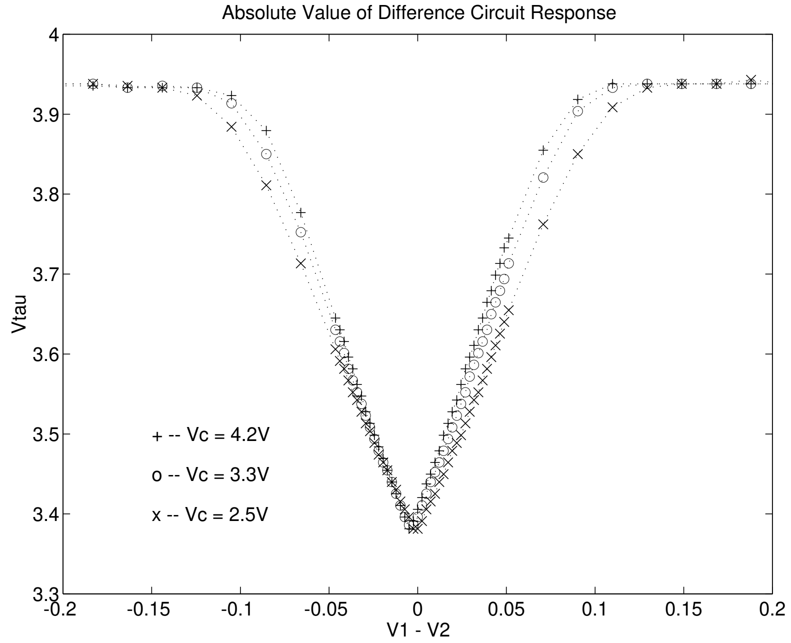


FIGURE 13-16: Measured response of one absolute-value-of-difference circuit.

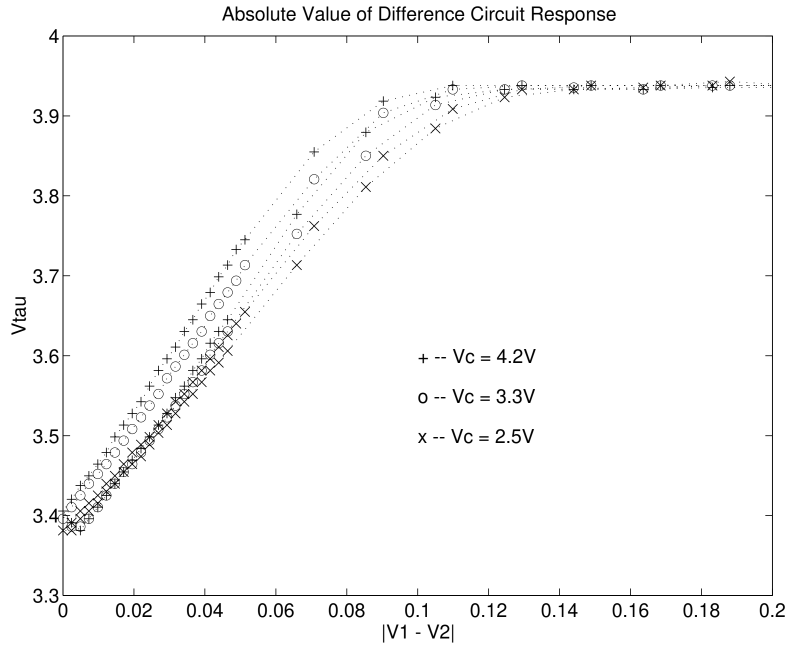


FIGURE 13-17: Measured response of one absolute-value-of-difference circuit, plotted against $|V_1 - V_2|$.

combination of both the differential amplifier offset and the mismatch in the source-follower buffers.

It should be noted that the curves flatten much more abruptly for absolute input differences greater than 110mV than those of the differential amplifier by itself. Closer analysis of the combined circuit reveals that the cause of this abrupt flattening is the diminishingly small current supplied to the sense amplifier by the high differential amplifier output side. As the output approaches the saturation level, it becomes unable to charge the sense amplifier input gates within the alotted precharge period. Increasing the precharge time is not an effective solution to widening the range of the absolute-value-of-difference circuit, however, as the time needed rises very rapidly as the current goes to zero. The preferred method for increasing the range is to raise the rail voltage, V_{DD}, on the differential amplifier, thereby increasing its saturation voltage. Raising V_{DD} will also increase power dissipation during the edge detection cycles. However, as the time spent in edge detection is much less than that required to load the image, the net increase in average power should be negligible. Unfortunately, the test system was not designed to allow separate power supply voltages for the edge detection circuits and the CCD clock drivers, and hence it was not possible to implement this option in present setup.

The composite results from twelve different absolute-value-of-difference (AVD) circuits for common mode input voltages of 4.2V, 3.3V, and 2.5V, with $V_{DD} = 5$V, are plotted as individual points in Figure 13-18. The horizontal distance between the dashed lines bounding the results indicates the overall resolution of the edge detection circuits for the array processor. For values of $|V_1 - V_2| < 90$mV, this distance is approximately 65mV, while for differences greater than 100mV, the distance becomes infinite. The value of 65mV would be correct for the lower bound of distinguishable differences of 2.5% FSR, if the input range is 2.6V. The upper limit, however, is clearly inadequate for the 10% of FSR which was desired.

13.5 Operation of the Full Array Processors

Two array sizes were built to test the operation of the complete MSV processor. A 32×32 array—the largest which would fit on the maximum available die size—was laid out as a separate chip, while a smaller 4×4 array was included on the test structures chip. Given the poor charge transfer efficiency measured for the CCDs and the limited resolution

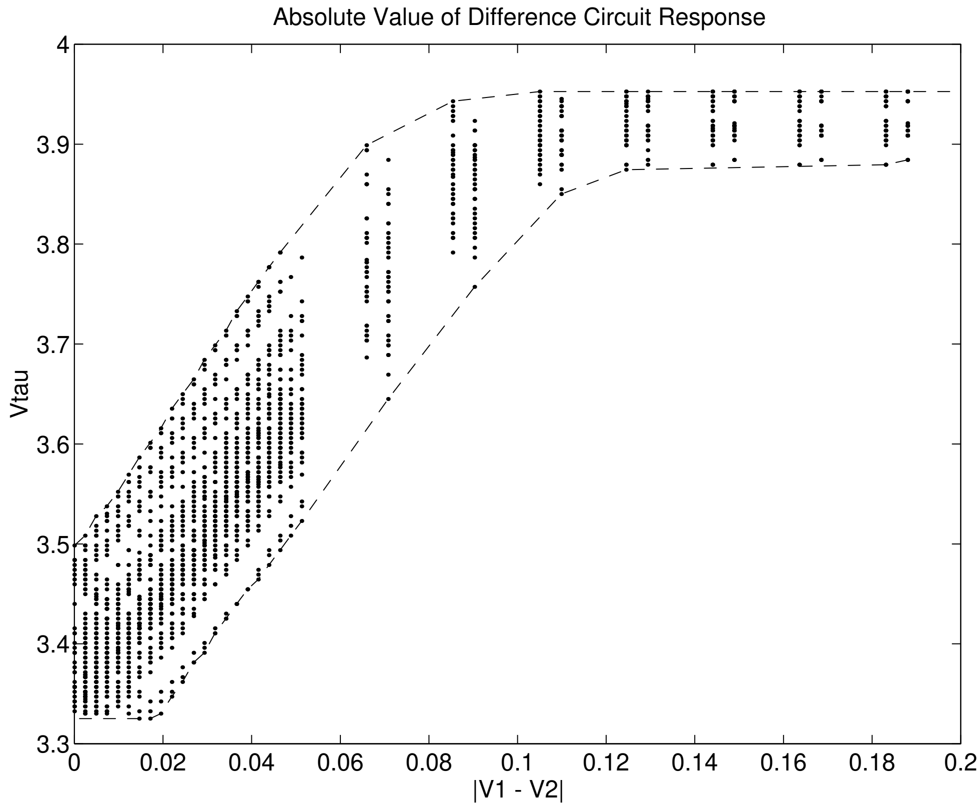


FIGURE 13-18: Composite response of twelve AVD circuits.

of the AVD circuit, it was clear that it would not be possible to test very precisely the processor's ability to discriminate between step edges, lines, and impulse noise as described in Chapter 5. The low CTE in effect results in a pre-smoothing operation as the image is loaded, while the limited AVD resolution restricts the number of smoothing cycles for which interesting results can be obtained. Nonetheless, it was possible to test several general characteristics of the array processors and verify that their overall operation was as planned.

The first test performed on the 4×4 and 32×32 arrays was to compare the I/O characteristics of the different rows. This was done by loading an entire column with the same value of $V_{in} - V_{ref}$ and measuring the floating-gate amplifier outputs of each row as the column was shifted to the end of the array. The results are shown for all rows of one 4×4 array in Figure 13-19 and for rows 3 through 16 of one 32×32 array in Figure 13-20. The curve for the 4th row of the smaller array is seen to be significantly shifted above those for

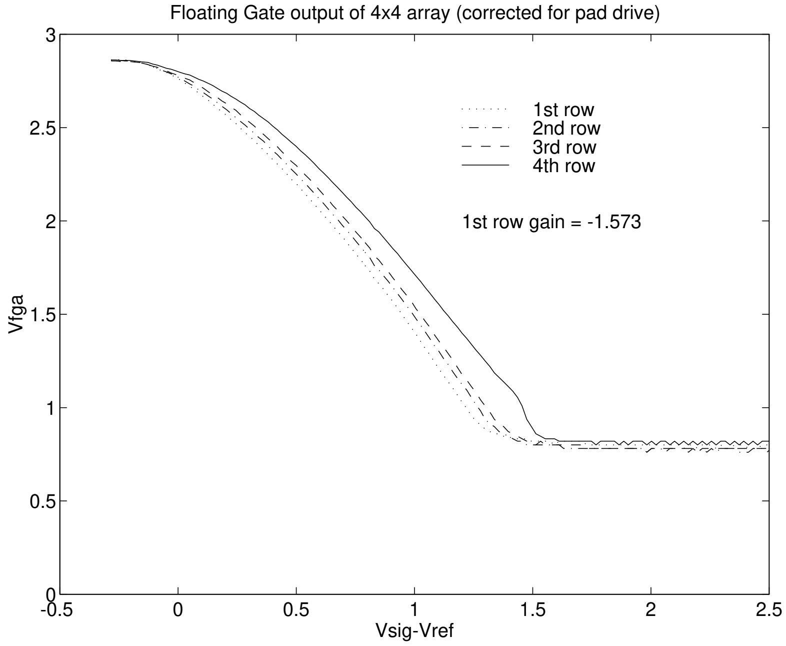


FIGURE 13-19: Floating-gate outputs of 4×4 array processor.

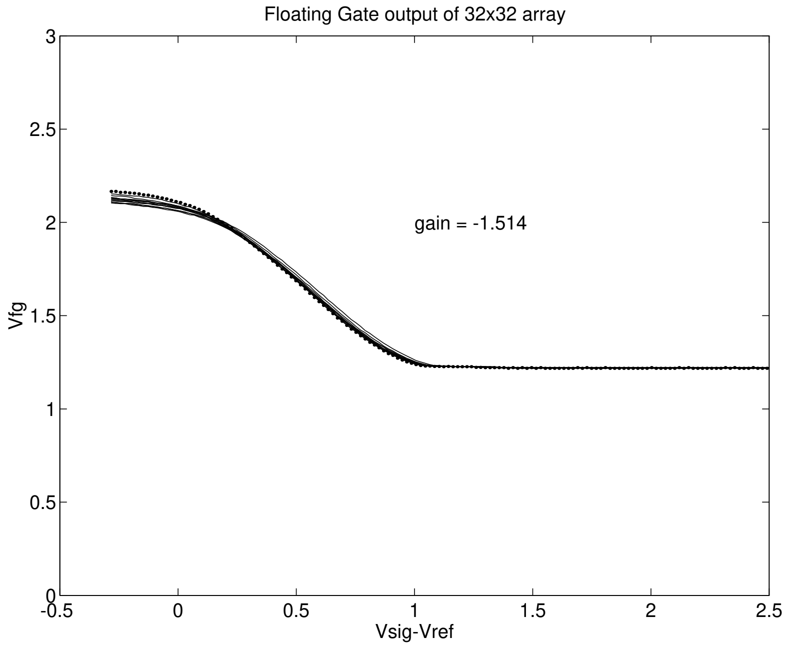


FIGURE 13-20: Floating-gate outputs of 32×32 array processor.

Input Data				4×4 Array Output Values			
0.000	0.000	0.000	0.000	2.644	2.639	2.634	2.644
0.000	0.000	1.199	0.000	2.595	2.488	2.075	2.634
0.000	0.000	0.000	0.000	2.634	2.632	2.627	2.632
0.000	0.000	0.000	0.000	2.634	2.632	2.629	2.634

a.) 0 smoothing cycles.

4×4 Array Output Values				Theoretical Smoothed Values			
2.632	2.595	2.573	2.610	2.624	2.582	2.558	2.605
2.581	2.510	2.463	2.559	2.603	2.523	2.476	2.565
2.620	2.583	2.563	2.593	2.617	2.576	2.552	2.597
2.634	2.632	2.627	2.622	2.634	2.632	2.631	2.632

b.) 1 smoothing cycle.

4×4 Array Output Values				Theoretical Smoothed Values			
2.627	2.588	2.573	2.610	2.606	2.573	2.559	2.580
2.576	2.539	2.527	2.561	2.597	2.557	2.541	2.566
2.617	2.573	2.561	2.578	2.608	2.581	2.570	2.587
2.634	2.625	2.622	2.622	2.627	2.619	2.616	2.620

c.) 2 smoothing cycles.

FIGURE 13-21: Smoothing of one-pixel impulse on 4×4 array.

the other three rows due to charge loss in the input shift register. Since the last row is the first to be input, it receives the smallest input charge packet while the preceding rows receive successively larger packets until steady-state conditions are reached (see Figure 13-12 for reference). The full effect of charge loss in the shift register is thus observed in the plot of the 4×4 array output, while the output curves for the 14 rows at the top of the 32×32 array are indicative of the steady-state results. The chips were frozen prior to testing so that dark current would not be a significant factor.

The small size of the 4×4 array was nonetheless convenient for testing the smoothing and edge veto functions. A test input was provided to the array consisting of a single pixel impulse with $V_{sig} = V_{in} - V_{ref} + \delta_{p_1 - p_2} = 1.199V$ at the 2nd row and 3rd column.

```
X     X     X     X
—     —     —     —
X  |  X  |  X  |  X
—     —     —
X     X     X     X
—
X     X     X     X
```

a.) smoothing cycle 0, $\tau_0 = 3.872$

```
X     X     X     X
—     —     —     —
X  |  X     X  |  X
—     —     —
X     X     X     X
—
X     X     X     X
```

b.) smoothing cycle 1, $\tau_1 = 3.860$

```
X     X     X     X
—     —     —     —
X     X     X     X
—
X     X     X     X
—
X     X     X     X
```

c.) smoothing cycle 2, $\tau_2 = 3.848$

FIGURE 13-22: Edge detection results for 4×4 array with impulse input.

Figure 13-21a shows the input data along with the floating-gate amplifier output of each row with no smoothing. Again, due to the poor charge transfer efficiency, the impulse is spread along the 2nd row over the first and second columns as well the third. Referring to the I/O transfer function curves plotted in Figure 13-19, the output value of 2.075V measured at the original location of the impulse corresponds to an input of $V_{sig} \approx 0.985V$, which is very close to the value of 0.943V predicted using a per-gate CTE of 0.995 given that the charge is transferred through 48 gates before reaching the output device.

The array outputs after one and two smoothing cycles are shown in Figures 13-21b and 13-21c next to the values predicted by applying the binomial kernels of equations (12.8) and (12.9) directly to the unsmoothed outputs given in Figure 13-21a. Comparing the results, it can be seen that the actual and predicted values are within 10–20mV of each other and indicate that the smoothing operation does in fact closely approximate a 2-D binomial convolution. It should be noted that this method for generating the theoretical values is acceptable as long as the outputs lie within the (approximately) linear range of the floating-gate amplifiers. Given the amount of variability in the I/O transfer functions of each row, this method is also preferable to that of translating the output data to their equivalent inputs and then performing the convolutions.

The edge detection/veto results for the same input pattern are shown in Figures 13-22a—13-22c. The threshold values of $\tau_0 = 3.872$, $\tau_1 = 3.860$, and $\tau_2 = 3.848$ were chosen from Figure 13-17 according to the expected differences in the pixel values at each smoothing cycle. Edges are indicated in the diagrams by the horizontal and vertical lines between the 'X's that mark the pixels in the array.

With no smoothing, edges are found between every vertical pair of pixels in the first column and between the first and second rows, between the vertical pairs of columns 1–3 in the second and third rows, and between each horizontal pair of the second row. Some of these edges are clearly in error. It turns out that the edges shown between the vertical pairs of the first column are meaningless since they are caused by a bad connection at pin 9 of the 'C40 bus interface which receives these signals (see the die photograph of the test chip in Figure 13-2). Unfortunately, this problem, which could be repaired only by rebuilding the 'C40 daughter board, was not discovered until relatively late in the testing phase when there was not enough time left to remake the board.

With the exception of the edge between the vertical pair at the top of the last column, which can be explained only by an extreme offset in the absolute-value-of-difference circuit

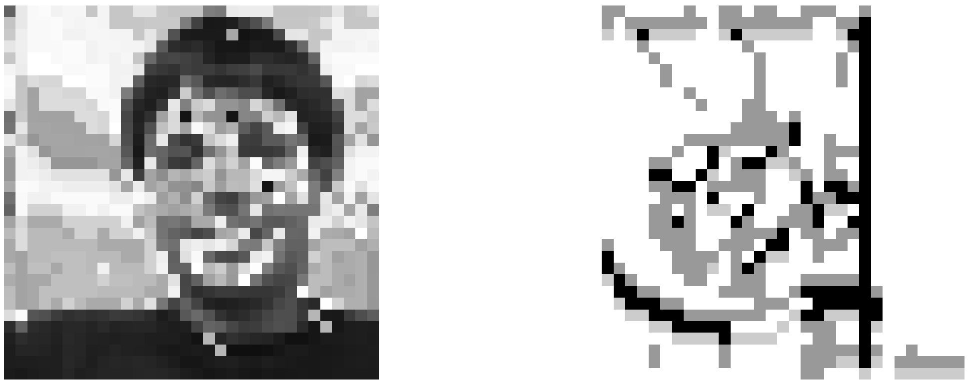


FIGURE 13-23: Test image used on 32×32 processor and corresponding edges.

at that location, the other edges found appear to be plausible as they occur around the smeared impulse input. After one smoothing cycle the edge between the horizontal pair at the center of the second row is removed, and after two cycles, all of the edges except the vertical edges between the first and second rows and the meaningless edges of column 1 are removed. The fact that the edges between the top two rows persist may be partially explained by the fact that the differences between rows 1 and 2 are not smoothed away as strongly as those between rows 2 and 3 due to the effect of the array boundary. Another more likely explanation, however, is the variation in edge threshold values between the different AVD circuits.

Edge detection on the 32×32 processor was also tested by supplying a sample input image and recording the edge outputs. The input image used in one test, which is shown at the left of Figure 13-23, was sampled down to 32×32 pixels from an original 256×256 image. Edges are displayed by coloring one of the adjacent image pixels in different shades of gray. An image pixel which has a vertical edge directly above it is colored light gray, while a pixel with a horizontal edge on its left side is colored dark gray. If edges exist both above and to the left of an image pixel, it is colored black.

The results, shown on the right side of Figure 13-23, are not simple to interpret given that we do not know the actual signal levels stored in the array. With the 32×32 array, the floating-gate outputs of only the top 16 rows were brought to output pads as this was the maximum number of A/D channels available in the test system. Even if we did have the outputs from all 32 rows, however, we would still not have an accurate representation of the internal signal levels since they would be further distorted as they were transferred to the output devices. Nonetheless, one can discern some general outlines, such as the edges found around the face area and near the shoulders and neck. The vertical black line towards the righthand side of the edge image, on the other hand, is due to the previously discussed bad connection at pin 9 of the 'C40 daughter board.

13.6 Recommendations for Improving the Design

Several problems were uncovered in testing the circuits used in the MSV processors which prevented all of the design goals from being met. Some of these problems had a trivial solution, such as redesigning the test system to provide a separate rail voltage to the differential amplifiers, while others, such as the poor charge transfer efficiency of the CCDs, could not be solved without changing the fabrication process. Nonetheless, it is clear from the overall results of the individual test circuits and the edge detection and smoothing tests on the arrays that given the proper resources, a processor can be built which does meet the design goals specified for the system. In this sense, the results of this research are positive.

After thoroughly studying the advantages and limitations of the current design, however, several changes to the array architecture which would greatly improve its effectiveness in the motion estimation system are now apparent. One problem with the present design is that the unit cell, which measures $224\mu m \times 224\mu m$, is too large. Even if scaled down by a factor of 4 to $56\mu m \times 56\mu m$, one could at best build a 160×160 array on a 1cm die, while for reliable motion estimation, the minimum array size needed is closer to 256×256.

The unit cell could be greatly reduced by removing most of the edge detection circuits and bringing them outside the array, as shown in Figure 13-24, where they can be shared by all cells in one row or column. The only circuits absolutely needed at each pixel are the floating-gate amplifier for sensing signal levels and the latches for storing the vertical and horizontal edge charges. The new unit cell structure, illustrated in Figure 13-25, could be as much as a factor of two smaller than the current design. Furthermore, bringing the absolute-value-of-difference circuits outside the array increases the flexibility for further improving their design for better matching and higher resolution, as the constraints on circuit area are no longer as severe.

Changing from pixel-parallel to row- and column-parallel processing will of course increase the total time needed for edge detection. One advantage to the reduced structure,

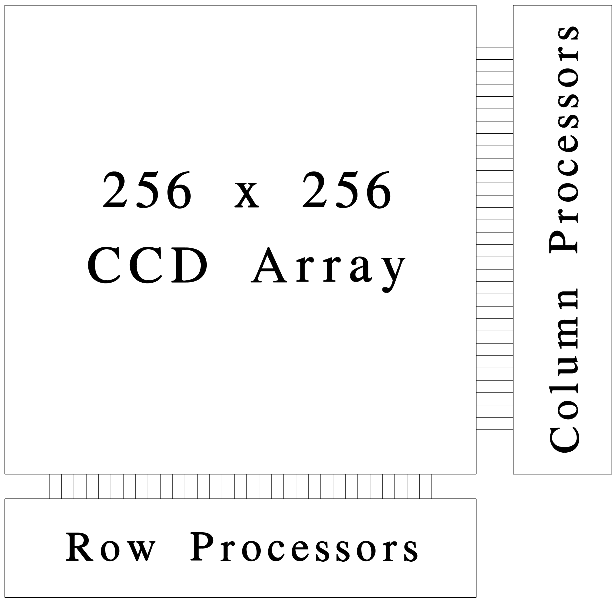


FIGURE 13-24: Proposed architecture for focal-plane processor

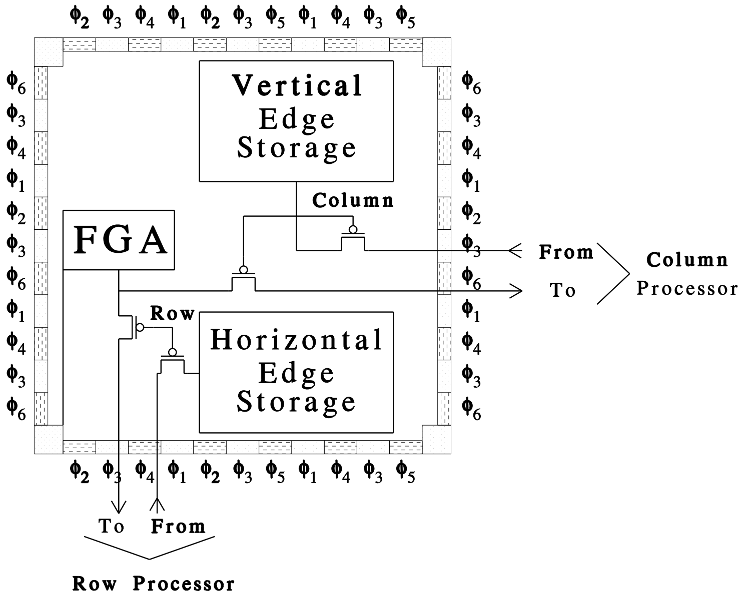


FIGURE 13-25: Unit cell structure in focal-plane processor.

however, is that it should be possible to use this array for imaging as well as for edge detection since, unlike the present design, it does not require a large fraction of the total pixel area to be allocated to n-wells and diffusions held at V_{DD} which can both trap and sink light-generated charge. The current array structure, on the other hand, is not as well-suited for imaging; and if it were to be used in the motion estimation system, not only would a secondary imaging device be needed, but the time required to load images into the array would also have to be taken into consideration. The suggested design improvement would thus not significantly affect overall processing time, and would in fact reduce the complexity of the system by removing the need for the additional sensor.

Making the suggested changes, it should be possible to build a 256×256 focal-plane MSV processor on a single chip using a $0.8\mu m$, or smaller, CCD-CMOS process. In Chapter 17, we will examine how the complete real-time motion estimation system could be assembled with this chip, along with the matching processors presented in Part III.

Part III

A Mixed Analog/Digital Edge Correlator

Chapter 14

Design Specifications

It is useful to recall the basic steps of the matching procedure, presented in Chapter 6, which are performed on each $M \times M$ block in the base edge map. The same notation is used as before where b denotes the 1-bit value of an individual pixel in the block being matched, and s denotes the value of an individual pixel in the search window. In addition, we let $P \equiv M^2$ represent the total number of pixels in the block, and define V_{ij} as the sum of absolute values of difference computed at position (i, j) in the search window. The steps of the matching procedure are summarized as follows:

1. Count the number of edge pixels in the block from the base edge map to find $\|B\| = \sum b$.

2. Compute $\alpha_1\|B\|$ and test that

$$V_l \leq \alpha_1\|B\| \leq V_h \tag{14.1}$$

   where

$$V_h = \alpha_1\frac{P}{2}, \text{ and, } V_l = \alpha_2 P \tag{14.2}$$

   and α_1, α_2 are constants chosen to allow acceptable detection and false-alarm rates, satisfying

$$1 > \alpha_1 > 2\alpha_2 > 0 \tag{14.3}$$

   as described in Section 6.1.

3. If $\alpha_1\|B\|$ is outside these bounds, stop. The block cannot produce an acceptable

match. Otherwise, for each position (i, j) of the search window:

(a) Compute for each pixel in the block, the binary function: $b\overline{s} + \overline{b}s$ and sum these values over the entire block to find the score

$$PV_{ij} = \sum b\overline{s} + \overline{b}s \tag{14.4}$$

(b) If $PV_{ij} \leq \alpha_1 \|B\|$

i. Store the current value of (i, j) as a candidate match position.

ii. If $V_{ij} < V_{min}$, the current minimum score, set $V_{min} = V_{ij}$ and store the current value of (i, j) as the best position.

4. After the score has been computed at every position of the search window, if at least one candidate match has been found, compute Δx and Δy, the maximum spread in the x and y coordinates of the candidates, and test if both $\Delta x \leq d_{max}$ and $\Delta y \leq d_{max}$, where d_{max} is the maximum possible spread for considering that the minimum is unique and well localized. If the results of both tests are true, signal that the match at the position of V_{min} is acceptable.

In order for the matching circuit to produce useful data for computing motion, three primary constraints must be satisfied. The first of these is that the block size, P, must be large enough to ensure that we can find constants α_1 and α_2, as defined in step 2 above, to give adequate detection and false-alarm rates. The second constraint is that the search window must be large enough to account for the maximum displacements in the image plane caused by the motion. Finally, the circuit must be designed to compute the quantities $\alpha_1 \|B\|$ and PV_{ij} with sufficient precision to accurately perform the validation tests and find the minimum score.

These constraints, which will be examined more closely in the following sections, determine the minimum design specifications for the matching processor. Of course, the best design, given the requirements of the motion system, will be the one that not only meets these specifications, but also consumes the least power and silicon area. In the next two chapters, we will look at several different implementations of the matching procedure to find the one which is best according to all of these criteria.

14.1 Required Block Size

We can find the minimum required block size from equations (6.16) and (6.17) for the mean and variance of V_{ij} under the hypothesis, H_0, that the match is false.

$$\mu_{H_0,\|B\|} = p_s \left(1 - \frac{\|B\|}{P}\right) + \frac{\|B\|}{P}(1 - p_s) \tag{14.5}$$

and

$$\sigma^2_{H_0,\|B\|} = \frac{p_s(1-p_s)}{P} \tag{14.6}$$

where p_s is the probability that an individual pixel in the search window will be an edge pixel. Since $\|B\|/P \leq 1/2$, due to the validation test (14.1), and $p_s \geq 0$, we know that

$$\mu_{H_0,\|B\|} \geq \frac{\|B\|}{P} \tag{14.7}$$

Given that $P = M^2$, we also have the following bound on the variance

$$\sigma_{H_0,\|B\|} \leq \frac{1}{2M} \tag{14.8}$$

with equality being achieved only for $p_s = 1/2$.

In order to give a low false-alarm rate, the threshold, τ, for deciding to consider a match as a candidate is chosen such that

$$\sigma_{H_0,\|B\|} \ll |\tau - \mu_{H_0,\|B\|}| \tag{14.9}$$

Let n be the smallest number such that

$$|\tau - \mu_{H_0,\|B\|}| \geq n\sigma_{H_0,\|B\|} \tag{14.10}$$

for all permissible values of $\|B\|$. From the validation test (14.1), we have

$$\frac{\|B\|}{P} \geq \frac{\alpha_2}{\alpha_1} \tag{14.11}$$

while the cutoff threshold is given by $\tau = \alpha_1\|B\|/P$. Combining these equations with the inequality (14.7) gives

$$\begin{aligned} |\tau - \mu_{H_0,\|B\|}| &\geq \frac{\|B\|}{P}(1-\alpha_1) \\ &\geq \frac{\alpha_2}{\alpha_1}(1-\alpha_1) \end{aligned} \tag{14.12}$$

We can thus ensure that the inequality (14.10) is satisfied for some suitably large value of n by choosing M such that, given α_1 and α_2,

$$\frac{\alpha_2}{\alpha_1}(1-\alpha_1) \geq \frac{n}{2M} \tag{14.13}$$

or,

$$M \geq \frac{n}{2\alpha_2/\alpha_1(1-\alpha_1)} \tag{14.14}$$

In using these bounds to determine M, we need to set α_1 as large as possible and α_2/α_1 as small as possible so that a sufficiently large number of blocks from the base edge map will pass the validation test (14.1). Otherwise, it may not be possible to find enough correspondence points to obtain good motion estimates.

The values of α_1 and α_2/α_1 most used in testing the matching procedure on real images were 0.5 and 0.15, respectively. For the tests presented in Chapter 6, M was set equal to 24 pixels, giving $n \geq 3.6$. As can be judged from the quality of the motion estimates listed in Chapter 7, this value was adequate for achieving a low error rate. Based on these results, we will thus require that the matching circuit be able to accomodate block sizes of at least 24×24.

14.2 Required Search Area

In Section 2.3, image plane displacements were calculated, assuming a focal length of $f = 200$ pixels, for three special cases: pure translation along the $\hat{x}$ direction, pure translation along the $\hat{z}$ direction, and pure rotation of θ about $\hat{\boldsymbol{\omega}} = \hat{y}$. At a frame rate of 1/30 sec, the maximum absolute displacements were found to be

1. For pure translation along $\hat{x}$: $88.8/Z$ pixels, where Z is the distance in meters to the object being viewed.

2. For pure translation along $\hat{z}$: $(x, y)/300$ pixels where (x, y) are image plane coordinates measured from the principal point, and

3. For pure rotation of θ about $\hat{\omega} = \hat{y}$: 3.5 pixels per degree of rotation, at the center of the image.

These motions are quite typical of those that could be encountered in actual imaging situations. With $f = 200$ pixels, a sensor size of 256×256 pixels corresponds approximately to a field of view of 32.6°, as measured from the $\hat{z}$ axis, which is close to the largest that can be obtained from ordinary lenses without significant distortion.

As can be seen from a few rough calculations, the largest displacements are caused by rotation about an axis parallel to the image plane. For example, with 5° of rotataion about $\hat{y}$, the smallest offset is 17.5 pixels at the center of the image. Combined with a translation along $\hat{x}$, the displacement could easily exceed 30 pixels if there are objects in the scene closer than 10 meters. Even if the motion is primarily a translation along the $\hat{z}$ axis, as would be the case for a camera mounted on the front of a car, any small rotation caused by vibrations or by turning can result in large offsets.

We must thus plan for relatively large search windows of anywhere from 30×30 to 200×200 pixels[1]. Of course, increasing the pixel size would reduce the magnitude of the displacements in number of pixels. However, it would also increase the actual image area covered by a single block since the required number of pixels per block would not change. As the area covered by the blocks becomes larger with respect to the total sensor area, not only can fewer blocks containing different features be extracted from the image, but the error in assigning the correspondence point to the center of the best matching block in the second image will increase.

In addition to requiring large search windows, we should also require the size of the window to be adjustable according to the type of motion which is expected. For example, if the motion is primarily in the $\hat{x}$ direction, or if the $\hat{y}$ axis is the primary axis of rotation, as in the case of the turning car, the image plane displacements will mostly be in the $\hat{x}$ direction, and hence there is no point in wasting time searching over large $\hat{y}$ offsets.

It should be noted that the search area requirements for the matching circuits to be used in the motion system are very different from those of the motion estimation chips typically

[1]The window sizes for the astronaut and lab sequences shown in Part I were 120×120 and 200×60, respectively.

used in video applications for camera stabilization or image sequence compression. In these applications, the camera is mostly stationary while the motion in the scene is caused by the people or objects being filmed. The differences in successive frames in these situations are usually small, and it is commonly assumed that maximum displacements are on the order of ± 8 pixels.

14.3 Precision

There are three tests performed to validate a candidate match. The first two test the eligibility of the entire block by verifying that the total number of edge pixels is within the acceptable upper and lower bounds. Combining equations (14.1) and (14.2), we have

$$\alpha_2 P \leq \alpha_1 \|B\| \leq \alpha_1 \frac{P}{2} \tag{14.15}$$

The above expression is written in a manner to emphasize that the known quantities $V_l = \alpha_2 P$ and $V_h = \alpha_1 P/2$ should be premultiplied and fed directly into the matching circuit, rather than be computed on-chip. Only the value of $\alpha_1 \|B\|$ needs to be computed by the circuit, and this can be done as the block is being read in. The precision required for representing $\alpha_1 \|B\|$ depends on whether the comparison is performed digitally or in analog. In an analog circuit, we only need to ensure that the difference $V_h - V_l$ is large enough to discriminate a sufficient number of different levels in the value of $\alpha_1 \|B\|$. In a digital circuit, if α_1 and α_2 are negative powers of 2, the operations required to perform the comparisons are trivial. If more precision is required, however, floating-point arithmetic must be used.

The closest powers of 2 to the values of 0.5 and 0.15 used in simulating the matching procedure on the test image sequences are $\alpha_1 = 1/2$ and $\alpha_2/\alpha_1 = 1/8$. These numbers may be adequate for many images, however, the lower value for α_2/α_1 will increase the false-alarm rate. From equation (14.13) with $M = 24$, the values $\alpha_1 = 0.5$ and $\alpha_2/\alpha_1 = 0.125$ give $n \geq 3$, as opposed to $n \geq 3.6$ with $\alpha_2/\alpha_1 = 0.15$. Furthermore, there is not much flexibility for tuning if α_1 and α_2 are restricted to negative powers of 2. Not implementing the multiply and compare operations with some form of extended precision arithmetic will thus reduce the robustness of the system.

The third validation test which is performed is to compare

$$PV_{ij} \leq \alpha_1 \|B\| \tag{14.16}$$

at each position (i, j) of the search window. The value of PV_{ij} can be anywhere from 0 to P, although, since $\alpha_1\|B\| \leq \alpha_1 P/2$, we only need to represent values up to $\alpha_1 P/2$. With $\alpha_1 = 0.5$ and $P = 576$, $\alpha_1 P/2 = 144$, which requires 8 bits to represent digitally. Precision issues with the representation of PV_{ij}, as well as $\alpha_1\|B\|$, are thus a concern primarily for analog implementations. It is certainly not necessary to require the circuit to discriminate a full 144 levels, however, we do need to ensure that the threshold test (14.16) is accurate to at least a fraction of $\sigma_{H_0,\|B\|}$, and we also need to ensure that the circuit can discriminate between different candidate minimum values of PV_{ij}.

Any difference between scores that is within one standard deviation of the expected minimum value cannot be considered significant. From equation (6.14), the mean and variance of PV_{ij} under the hypothesis H_1 that the match is correct is given by

$$E[PV_{ij}|H_1] = Pp_n \tag{14.17}$$

$$\text{Var}(PV_{ij}|H_1) = Pp_n(1-p_n) \tag{14.18}$$

where p_n is the probability of an edge pixel being turned on or off by noise. Suppose $p_n = 0.05$, with $M = 24$ we have

$$\sigma_{(PV_{ij}|H_1)} \equiv \sqrt{\text{Var}(PV_{ij}|H_1)} = M\sqrt{p_n(1-p_n)} = 5.23 \tag{14.19}$$

In order to have $\sigma_{(PV_{ij}|H_1)} \leq 1$, it would also be necessary to have $p_n \leq .00174$, which is much lower than can be reasonably expected. Being able to discriminate between scores that are within 3 or 4 votes of each other should thus be sufficient. From inequality (14.8), we also see that

$$\sigma_{H_0,\|B\|} \leq \frac{1}{2M} = \frac{1}{48} \tag{14.20}$$

and hence requiring that the circuit be able to discriminate more than 48 different values of $\alpha_1\|B\|$ will ensure that the threshold test (14.16) can be performed to an accuracy greater than $\sigma_{H_0,\|B\|}$.

Chapter 15

Case Study: A Purely Digital Design

The matching procedure can be implemented by two very different architectures, both involving fully parallel array processing. In the first method, pixels from the block being matched are stored at the nodes of the array while the pixels from the search window are shifted across it. A score is computed at each shift cycle and compared with the current minimum value. If it is smaller, the minimum value is updated and the position of the search window is recorded. Once the entire search window has been processed, if all of the validation tests have been passed, the offset corresponding to the minimum score is reported as the position of the best match.

In the second architecture, which has been used in some commercially available motion estimation chips[1], the entire search window is stored in a processor array where each node corresponds to a given offset. Each pixel from the base image block is broadcast to the entire array so that its difference with every pixel of the search window can be computed simultaneously and the results added to the current scores stored at each node. The pixels of the search window are then shifted so that their offset relative to the next base pixel to be processed corresponds to the offset assigned to their new array location. Once all the pixels from the base image block have been processed, the validation tests are performed and the scores stored at each node are compared to find the one with the minimum value.

In the next two sections, I will discuss how the matching circuit could be designed, given the constraints presented in the last chapter, using each of these architectures.

[1]For example the STI3220 motion estimation processor from SGS Thomson, which was briefly discussed in Chapter 3

15.1 First Method: Moving the Search Window

The structure of the $M \times M$ processing array needed for the first architecture is shown in Figure 15-1 with the block diagram of the individual processing cells given in Figure 15-2. Each bit from the base image block is loaded and stored in a latch for the duration of the search. During the load phase, the edge pixels are counted and the value of $\alpha_1\|B\|$, as defined in the preceding chapter, is computed and stored in a register so that it can be used for the threshold test performed on each score. The validation test comparing $\alpha_1\|B\|$ to the external inputs $V_h = \alpha_1 P/2$ and $V_l = \alpha_2 P$ can be performed once the entire block from the base image is loaded to determine if it is necessary to start the the search procedure. Scoring begins as soon as the block corresponding to the first offset in the search window is moved into position by means of the shift register cells located at each processing node.

The complexity of this design is not in computing the score at each pixel, which is a simple XOR operation, but in tallying the scores from every node in the array. Counting the scores must be performed as quickly as possible as there are many offsets in the search window and many blocks in the base image to match. Unless prevented by space restrictions on the chip, the tally function should thus be implemented as a single combinational logic circuit to avoid wasting cycles. Furthermore, even though we only need to represent numbers up to the value of $\alpha_1\|B\|$, all of the votes must counted, as we cannot know in advance which nodes will have a 'high' output and which will not.

Building a tally circuit to count votes from M^2 nodes is expensive both in area and in delay. Figure 15-3 shows the construction procedure for building an $2^n - 1$ vote counter with an n-bit output. A full adder can tally the three votes from P_0, P_1, and P_2 giving the 2-bit output $\{t_1 t_0\}$, where

$$t_0 = P_0 P_1 P_2 + P_0 \bar{P}_1 \bar{P}_2 + \bar{P}_0 P_1 \bar{P}_2 + \bar{P}_0 \bar{P}_1 P_2 \tag{15.1}$$

and

$$t_1 = P_0 P_1 + P_0 P_2 + P_1 P_2 \tag{15.2}$$

A 7-vote tallier can be constructed by adding the results from two 3-vote talliers and connecting the seventh node to the carry-in input of the 2-bit adder. Generalizing this procedure, it is easily seen that a $2^n - 1$ vote counter can be built from two $2^{n-1} - 1$ talliers and one $(n-1)$-bit adder, as shown in Figure 15-3c.

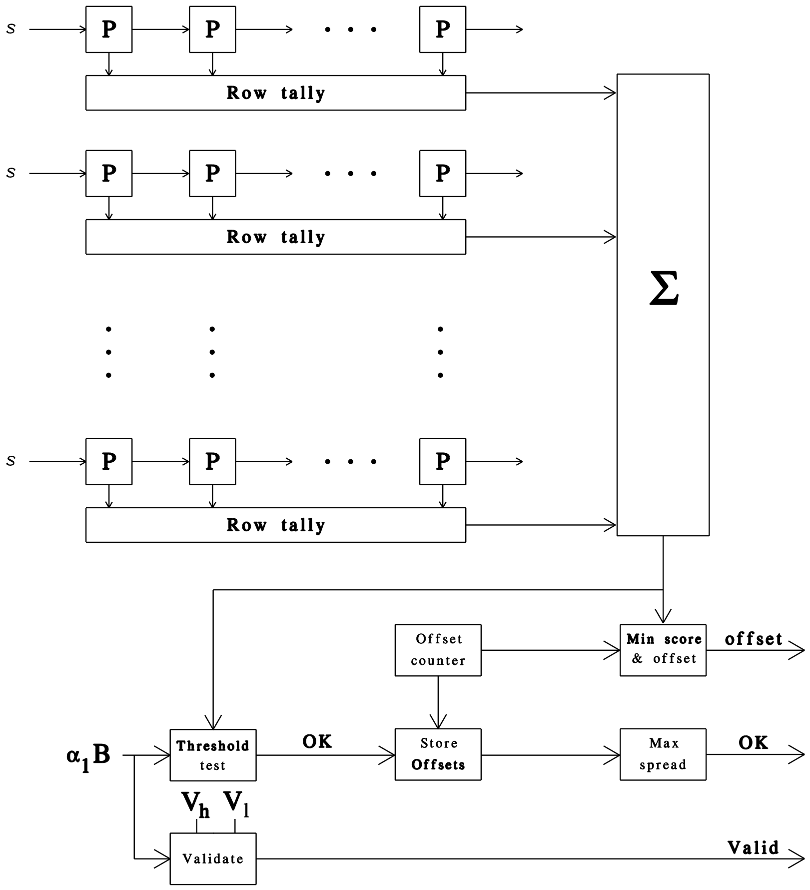


FIGURE 15-1: Processing array with base image block held in fixed position.

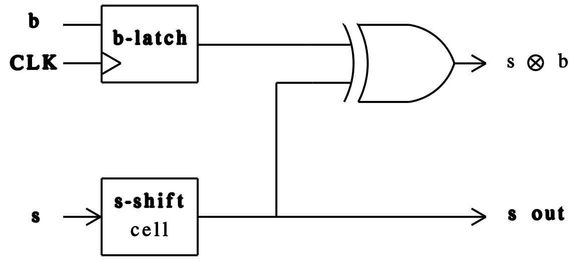


FIGURE 15-2: Individual processing cell.

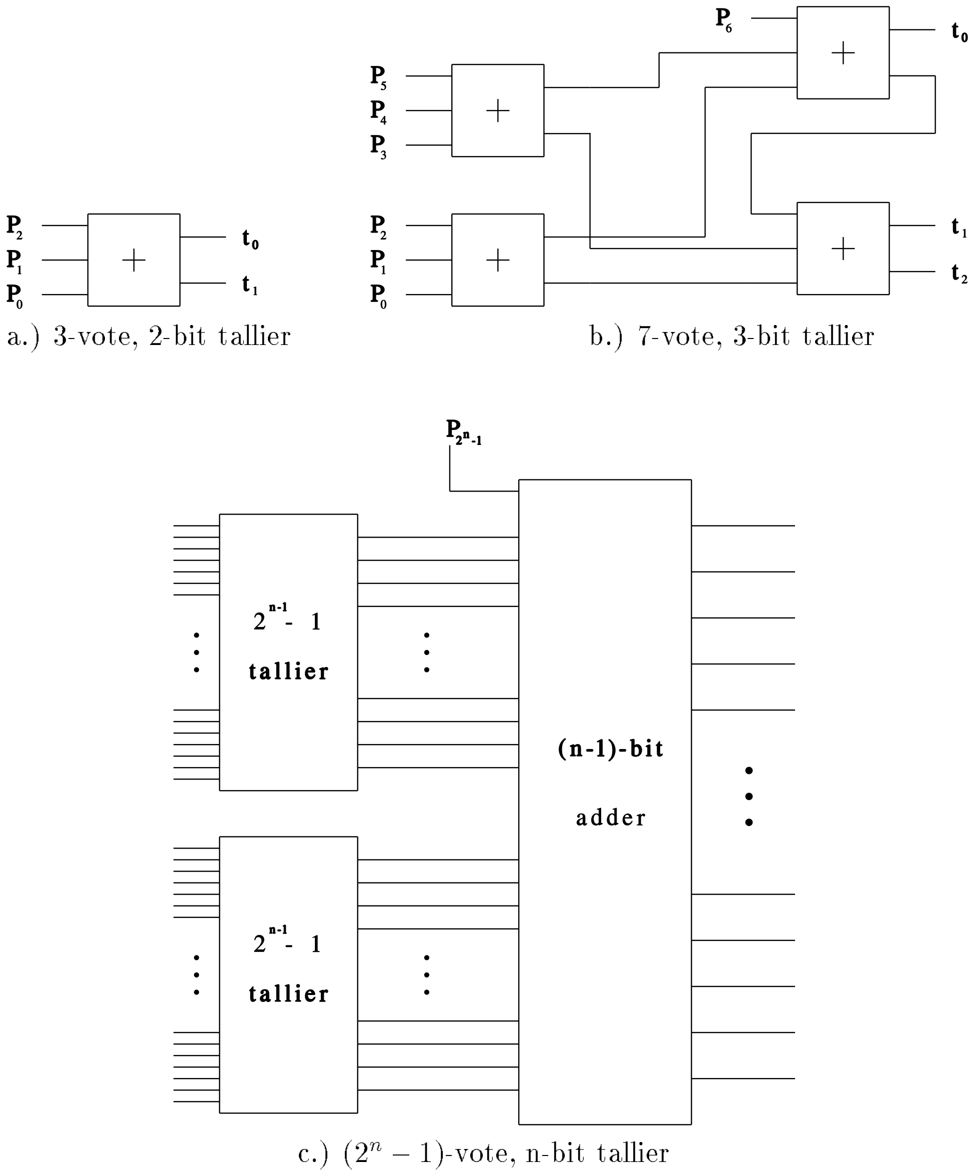


Figure 15-3: Tally circuit construction.

Let A_n denote the number of full adders required to implement a $2^n - 1$ vote counter. Clearly, A_n satisfies the recursive relation

$$A_n = 2A_{n-1} + n - 1 \tag{15.3}$$

and since $A_1 = 0$, it is easily verified that the solution to this recursion is given by

$$A_n = 2^n - n - 1 \tag{15.4}$$

The full $2^n - 1$ input tally circuit can be visualized as a tree with $n-1$ levels having $(n-i-1)\,2^i$ i-bit adders at each level. The total delay in the circuit is thus the sum of the worst case delays in the $n-1$ levels. Two choices with different worst case delays are possible for implementing the i-bit adders. The first is with *ripple-carry*, in which the carry bits propagate sequentially through the i full adders, while the other is with *carry-lookahead*, in which the carry bits are generated in parallel with combinational logic. If ripple-carry adders are used, the worst case delay for an i-bit adder is id, where d is the delay for a single full adder. The maximum delay for the full tally circuit would then be

$$\sum_{i=1}^{n-1} id = \frac{dn(n-1)}{2} \tag{15.5}$$

which increases as n^2. If carry-lookahead adders are used, the delay can be reduced to $O(n)$, but at the cost of more complexity in the adder circuits [102].

Equation (15.4) is most useful when the number of votes to count is one less than a power of 2. In the last chapter, it was determined that the minimum block size should be 24×24, meaning that 576 votes need to be counted with 10 output bits to represent the answer. Applying equation (15.4) with $n = 10$, we would need 1013 full adders to implement this circuit, which is clearly more than are actually necessary. The most efficient design, in terms of both area and interconnect requirements, is to build one 24-vote tallier per row and to sum the outputs from each row with an adder tree, as shown in the block diagram of Figure 15-1. We can build the row tallier with one 15-vote counter, requiring 11 adders; one 7-vote counter, requiring 4 adders; and one adder to count the remaining two nodes. One 3-bit adder and one 4-bit adder are then needed to combine the results for the entire row, for a total of 23 1-bit full adders. Summing the results from all 24 rows requires 12

5-bit adders, 6 6-bit adders, 3 7-bit adders, and 2 8-bit adders, giving a total of 133 1-bit adders. The entire circuit thus consists of $24 \times 23 + 133 = 685$ 1-bit adders.

One possible layout for a single node of the array is shown in Figure 15-4, with the equivalent circuit diagram in Figure 15-5. The top half of the layout contains the latch which stores the bit b, the shift register where the bit from the search window is temporarily stored, and the XOR circuit which computes the score for the node. The bottom half contains a 1-bit full adder and is laid out so that its width matches that of the top half of the cell as closely as possible so as to minimize wasted space. A single row of the processor array, along with its row tallier, can thus be formed by abutting 24 of the cells shown in the diagram, with one missing the adder section. To implement the tally function, an additional twelve horizontal lines of metal interconnect must be placed beneath the cell so that the inputs and outputs of the adder circuits can be properly wired. As shown in the diagram, the cell layout measures $340\lambda(\text{h}) \times 260\lambda(\text{w})$. With the additional metal lines, it will measure at least $412\lambda(\text{h}) \times 260\lambda(\text{w})$.

The control lines carrying the clock signals for phasing the shift register and latching the bits from the base image block run vertically across the cell so that rows can be abutted to form the full array. The complete 24×24 array will thus measure at least $9888\lambda(\text{h}) \times 6240\lambda(\text{w})$. In addition to the area taken by the array, the summation circuit to add the results from all of the rows, will require a minimum of 133 times the area of a single full adder, which as shown, measures $134\lambda(\text{h}) \times 196\lambda(\text{w})$.

It should be noted that the tally circuit just described uses ripple carry, and will thus have worst-case propagation delays proportional to the square of the number of levels in the summation tree, which in this case is five. The circuit can be made to run faster using carry-lookahead adders. However, these require more area, and because their structure is not as regular as that of the ripple carry adder, it would not be as simple to construct a unit cell for the array such as the one in Figure 15-4.

15.2 Second Method: One Processor Per Offset

The second architecture for implementing the matching circuit is interesting both because it can operate much faster than the first method, and because it has been used in the design of some existing motion estimation chips currently on the market.

The basic idea of this design is illustrated in the block diagram of Figure 15-6. The array

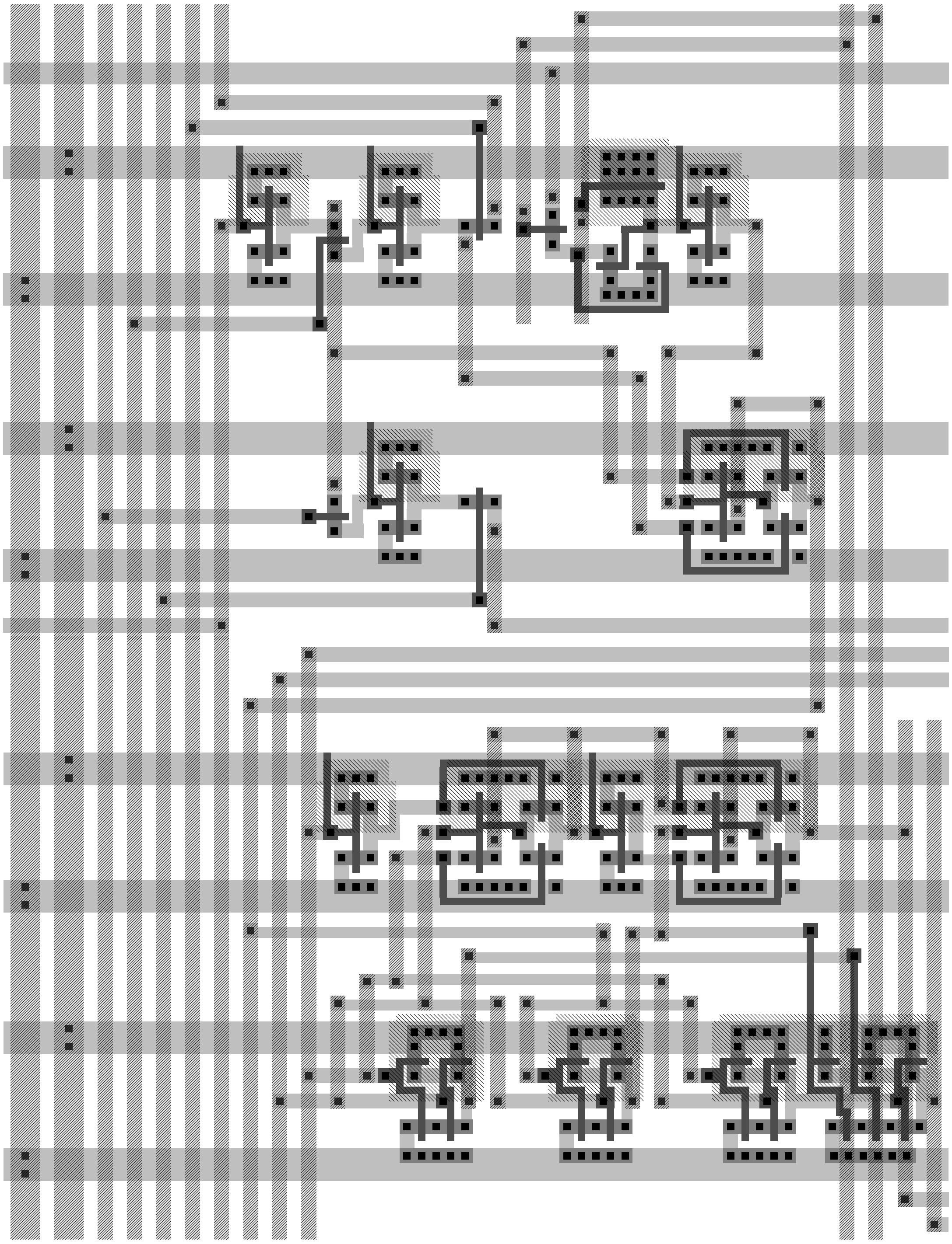


FIGURE 15-4: Layout of a unit cell including one full adder circuit.

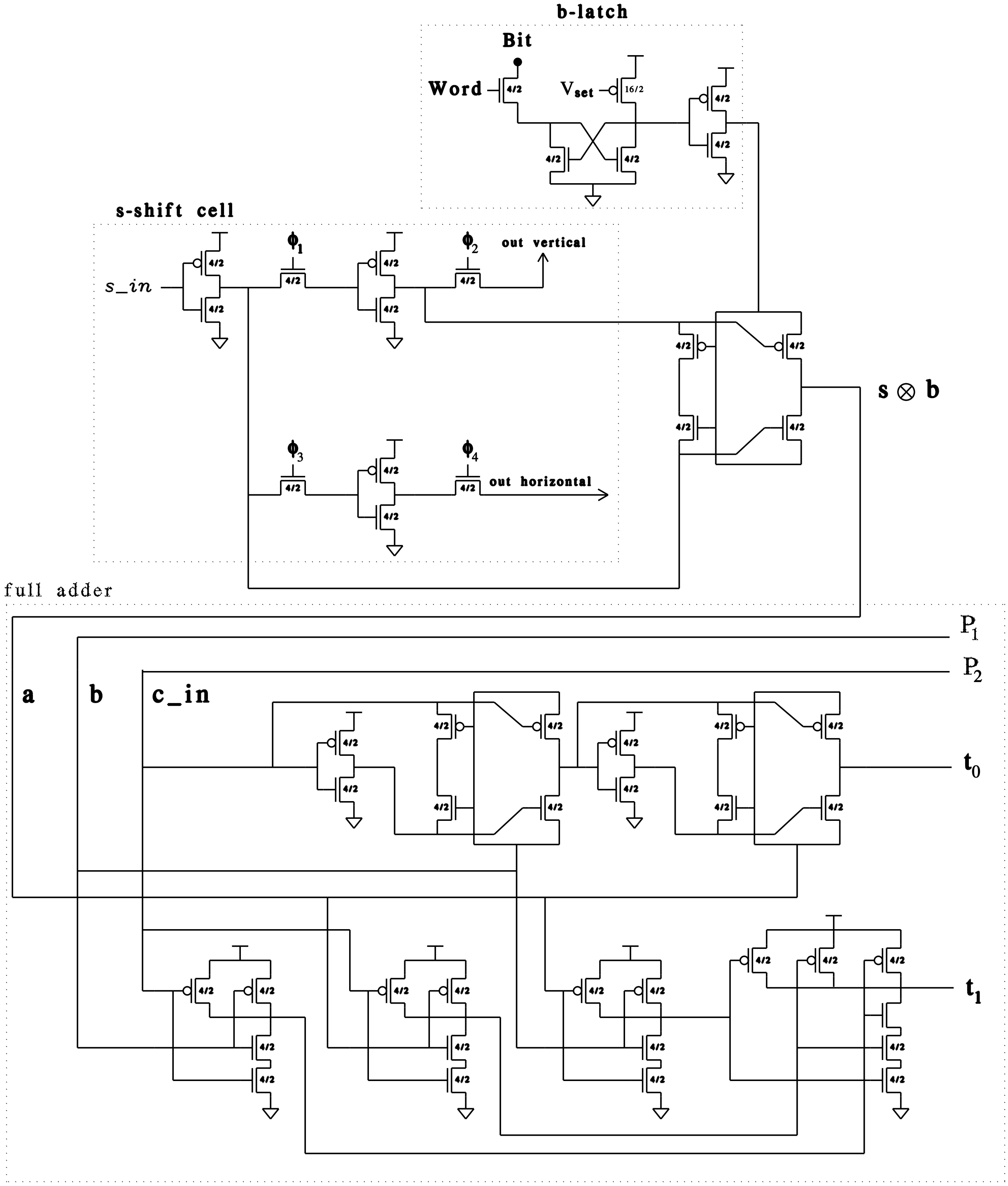


FIGURE 15-5: Circuit diagram for the layout of Figure 15-4.

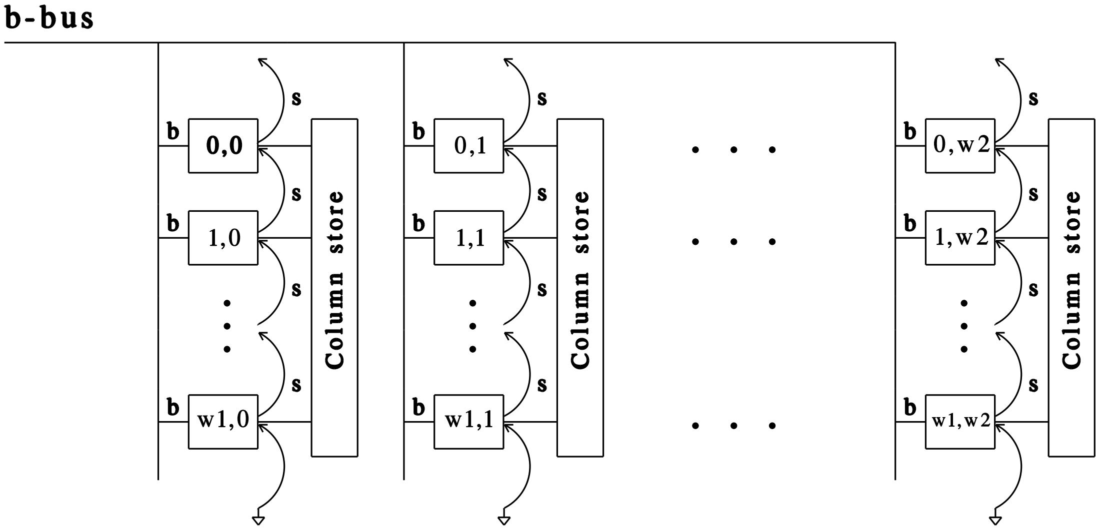


FIGURE 15-6: Offset processor array.

consists of a regular arrangement of processors, each corresponding to a given offset in the search window. Each processor consists of an XOR circuit to compute the score for one pixel in the base image block, a counter/accumulator to compute the total score, and a shift register cell to temporarily store one pixel from the search window as shown in Figure 15-7. The array is initialized by reseting all of the accumulators and loading the entire search window into the column store blocks placed between the columns of the processor array. In each processing cycle, one pixel from the base image block is broadcast on a global bus and its difference is computed simultaneously with every pixel from the search window.

The base image pixels are sequenced by column, such that pixels $(0, i)$ through $(M-1, i)$ from column i are processed in order, followed by pixel $(0, i+1)$ from the top of the next column. At the beginning of each column, the contents of the column stores, which each hold one column from the search window, are copied into the vertical shift registers linking the processor nodes. After each cycle, the search window pixels are shifted up one row so that their offset relative to the incoming base pixel corresponds to the offset assigned to their new location. Once the last pixel in the column from the base image has been processed, the contents of the column store blocks are shifted horizontally, and the procedure is repeated until the entire block has been processed. Assuming the processor array contains $W_1 \times W_2$ nodes, the minimum score and its offset can be found, once processing is completed, using

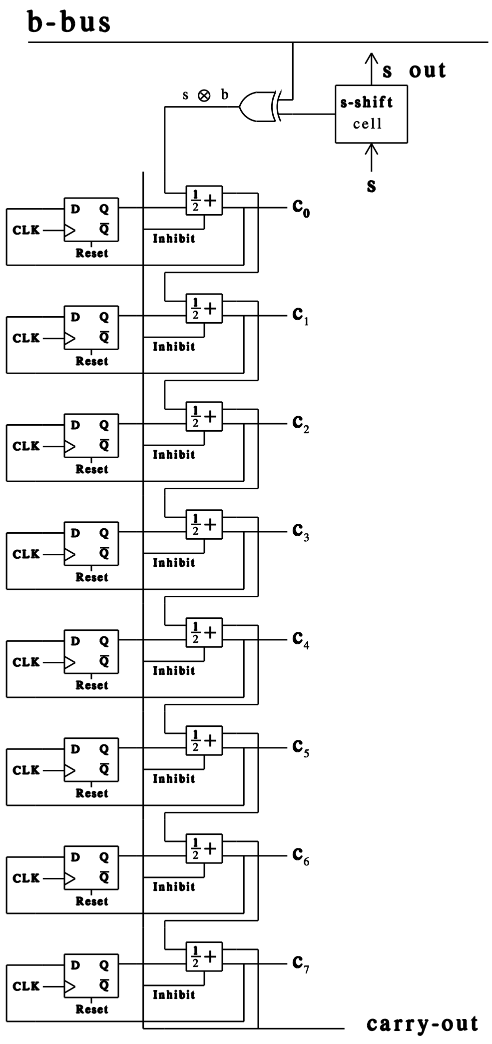


FIGURE 15-7: Block diagram of one processor node with 8-bit counter/accumulator cell.

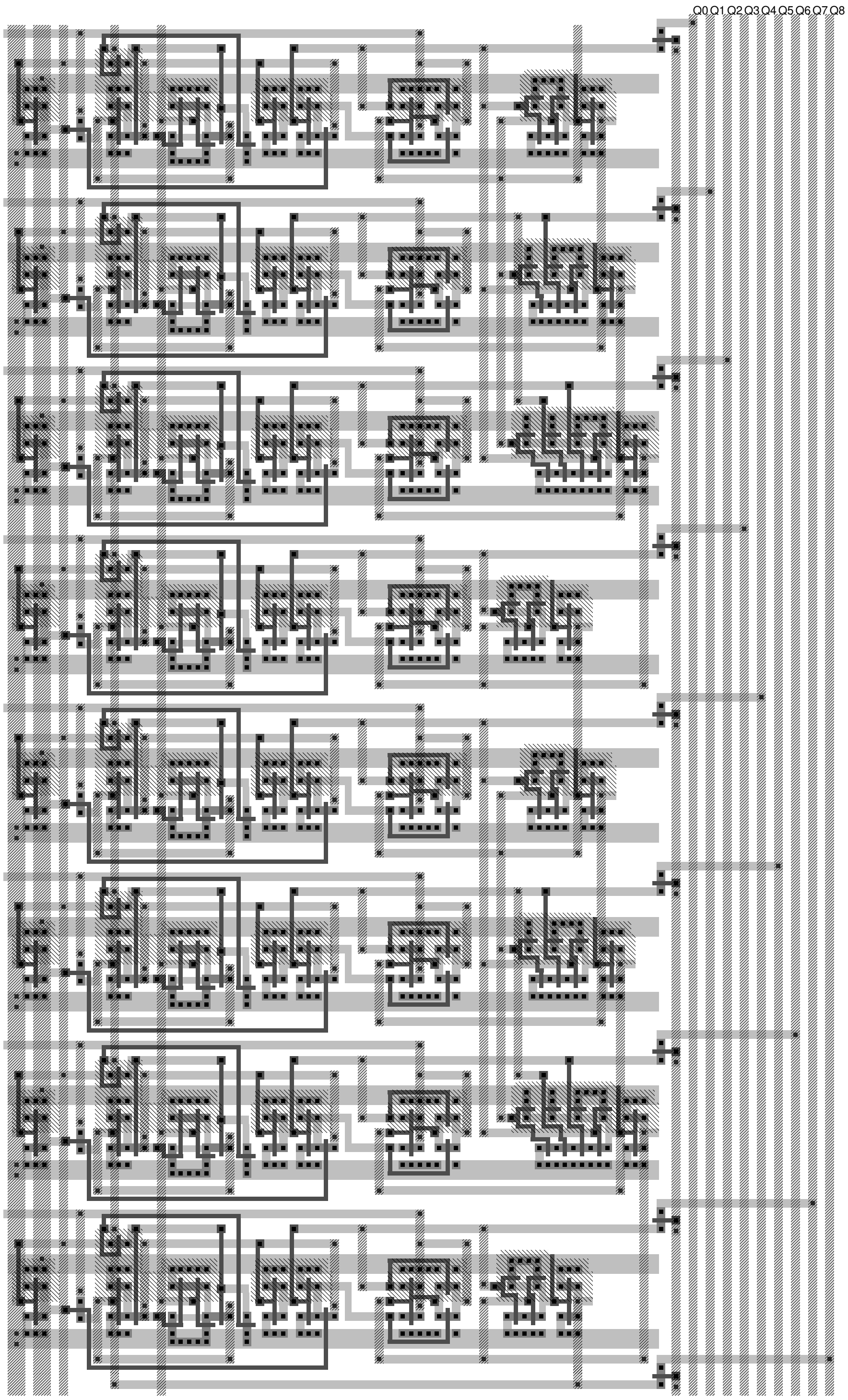


FIGURE 15-8: Layout of 8-bit counter and accumulator with inhibit on overflow.

a comparator tree with at most $2^n - 1$ comparators, where $n = \lceil \log_2(W_1 W_2) \rceil$.

This design is clearly much faster than the one discussed in the preceding section as the number of cycles needed to find the best offset is equal to the number of pixels in the base image block, instead of the much larger number of offsets within the search window. The price paid for this speed, however, is silicon area. As discussed in the last chapter, the accumulators at each node need to represent scores with 8 bits of precision, and the search window needs to be large enough to accomodate the typical displacements caused by the camera motion. It was estimated that the required size could be between 30×30 and 200×200 pixels.

A possible layout for an 8-bit counter and accumulator which could be used in the design is shown in Figure 15-8 and corresponds to the section of the block diagram in Figure 15-7 below the XOR circuit. This cell measures 647λ(h) $\times$ 387λ(w) and includes a circuit to inhibit counting if the carry-out bit of the accumulator goes high, signalling an overflow. In this design, the 8-bit counter is implemented with two 4-bit counters, each containing four 1-bit half-adders with carry-lookahead. Some area can be saved by removing the carry propagation circuits—which can be seen in the layout as the rightmost elements of the 1-bit subcells—and connecting the carry-out bit directly to the input of the next cell. This would decrease the width of the cell by 86λ, but will, of course, also reduce its operating speed. Based on the size of the counter/accumulator alone, ignoring the space needed by the column store and shift register subcells, we can see that a minimum size 30×30 array would require at least 19410λ(h) $\times$ 11610λ(w). Using a 0.8μm ($\lambda = 0.4\mu$m) or smaller process, we could conceivably build one 30×30 processor array on a single 1cm die.

Chapter 16

Mixing Analog and Digital

Of the two architectures discussed in the last chapter for a purely digital design, only the first one readily lends itself to analog processing. In the second architecture, in which the scores for each offset in the search window are computed simultaneously and accumulated as the base image block is read in, the principal processing element is the counter/accumulator. If designed with analog circuits, each node would need to include the following functions:

1. Scaling, to convert the binary output of the XOR circuit to a usably small unit voltage or current,
2. Addition of the new input to the previous value of the score, and
3. Storage of the result.

Since any mechanism to 'store' current also requires storing a control voltage, it is simpler to build an analog counter/accumulator which would operate entirely in the voltage domain. Adding voltages would require an opamp circuit with matched resistive elements, as well as low ouput impedance buffers to make the inputs appear as ideal voltage sources. Furthermore, care would need to be taken to ensure that the individual processors are matched to better than $\pm 3\%$ of their full scale range in order to meet the precision requirements outlined in Section 14.3. Even if it is possible to design the processors to these specifications, they will still require substantial silicon area, and will certainly be more expensive to fabricate than their digital equivalents.

In the first architecture, on the other hand, it is much less difficult to implement the tally function with the required precision using simple analog circuits. Votes can be counted

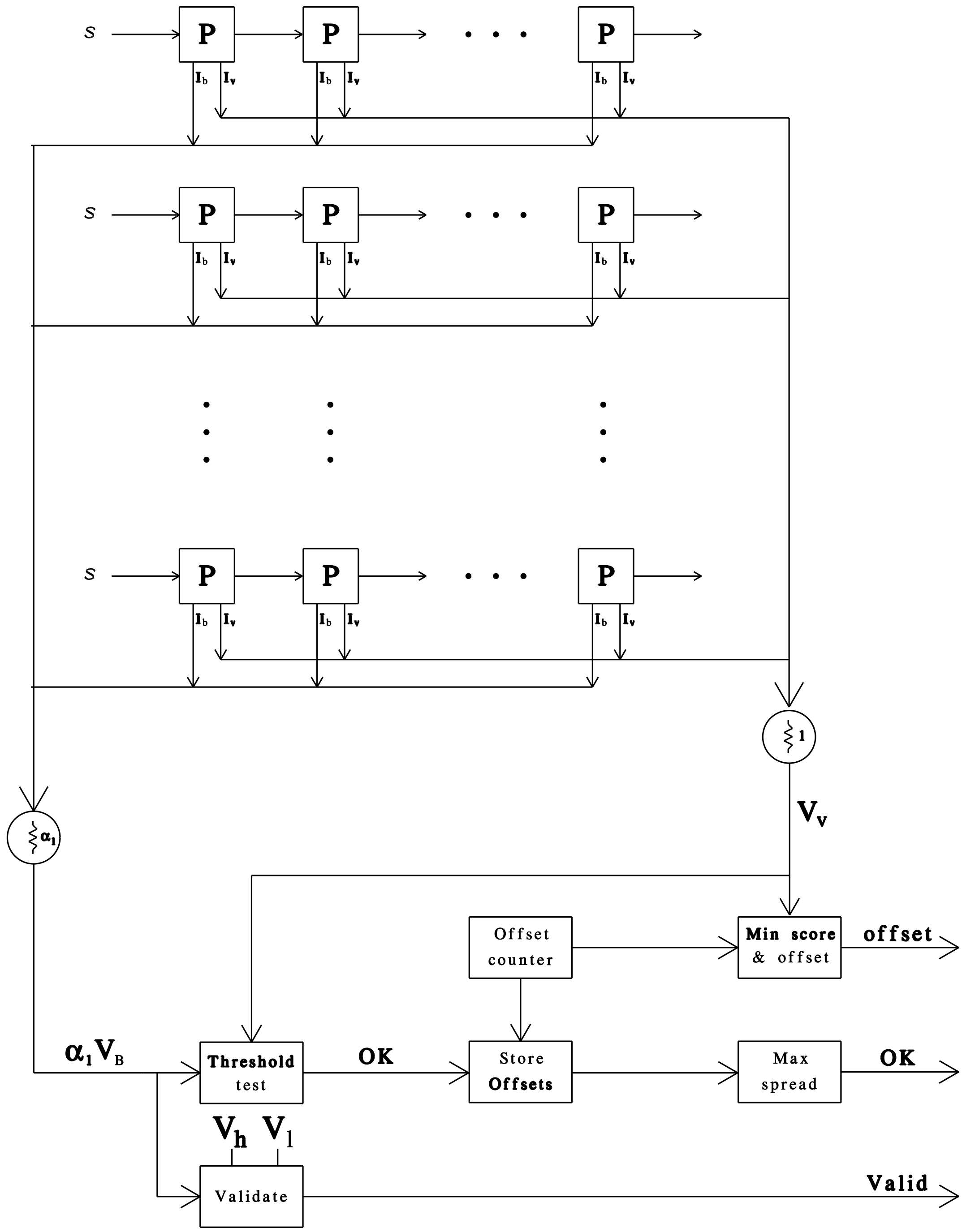


FIGURE 16-1: Processing array with both analog and digital elements.

by switching on current sources at each node when the output of the XOR circuit goes high. The currents from each node can then be summed on a single wire and the result converted to a voltage so that the total score can be compared with, and possibly replace, the stored minimum score.

The basic plan for a processor to implement the matching procedure using both analog and digital elements is outlined in the block diagram of Figure 16-1. This diagram is functionally identical to that of Figure 15-1. However, the row tally blocks and the summation circuit have been replaced by wires, and the comparison functions included in the blocks marked 'Min score & offset', 'Threshold test', and 'Validate' must now be implemented by analog circuits.

In this chapter, I will discuss the design of the principal elements in Figure 16-1 which involve analog processing and analyze their performance as predicted by simulations, using the device parameters for the HP CMOS26 process, also offered through MOSIS. Layouts were generated for each of these elements in order to compare total area requirements with those of the corresponding digital implementaion. Simulation results based on a circuit extraction from the layout of a 5×5 array are given in the last section to illustrate the ability of the mixed analog and digital processor to find an artificial test pattern in a 9×9 search window.

16.1 Unit Cell Design

The primary change to the unit cell is the addition of switched current sources to replace the full adder circuits used in the digital design. It was determined that better matching for the purposes of the threshold test could be achieved with less complexity if in fact two identical current sources were placed at each cell, one to be switched on when the value of $s \otimes b$ is high, and the other when the value of b is high. The resulting circuit diagram and layout for the unit cell are shown in Figures 16-2 and 16-3. The left three-quarters of the layout containing the s-shift cell, b-latch and XOR circuit are identical to the top half of the layout for the digital cell shown in Figure 15-4. The two current sources which occupy the right one-fourth of the cell add 80λ to its width so that the total cell measures 161λ(h) $\times$ 340λ(w), as opposed to the 412λ(h) $\times$ 260λ(w) used in the digital design. Including the analog current sources thus reduces the cell area by almost a factor of two. Given that the adder tree needed to sum the results from all of the rows is also no longer necessary, the

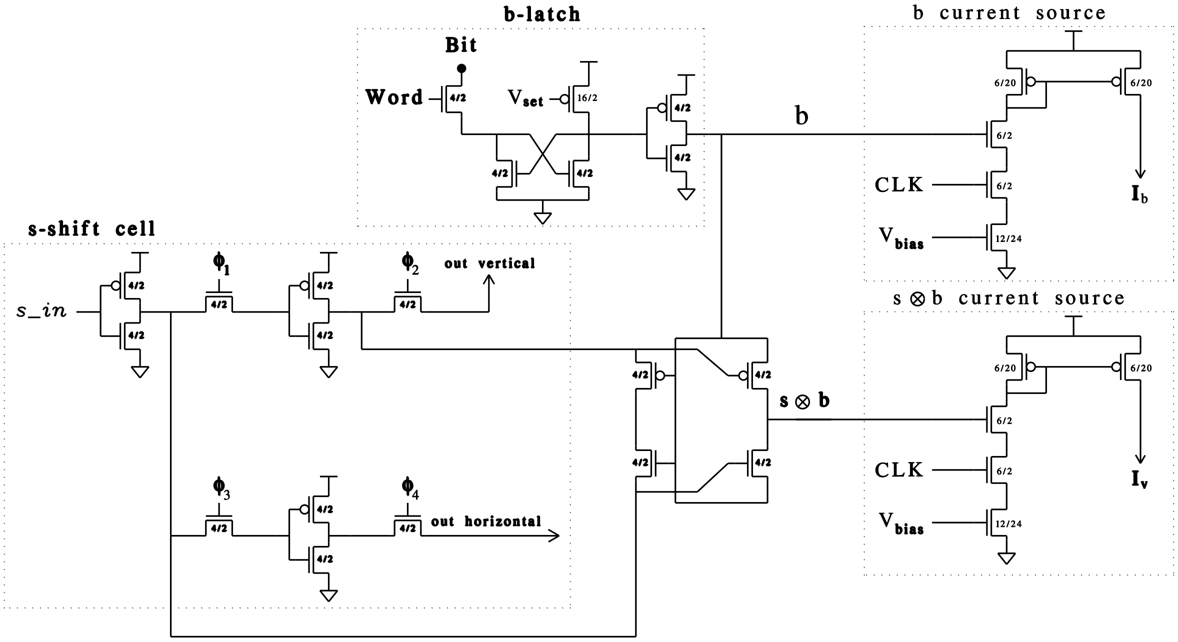


FIGURE 16-2: Circuit diagram of unit cell with switched current sources.

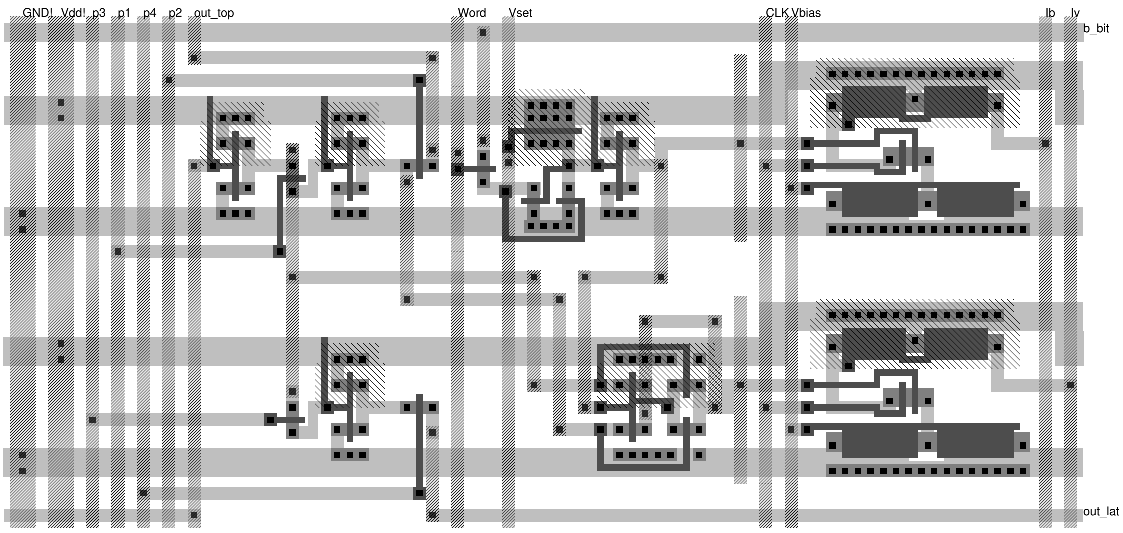


FIGURE 16-3: Unit cell layout.

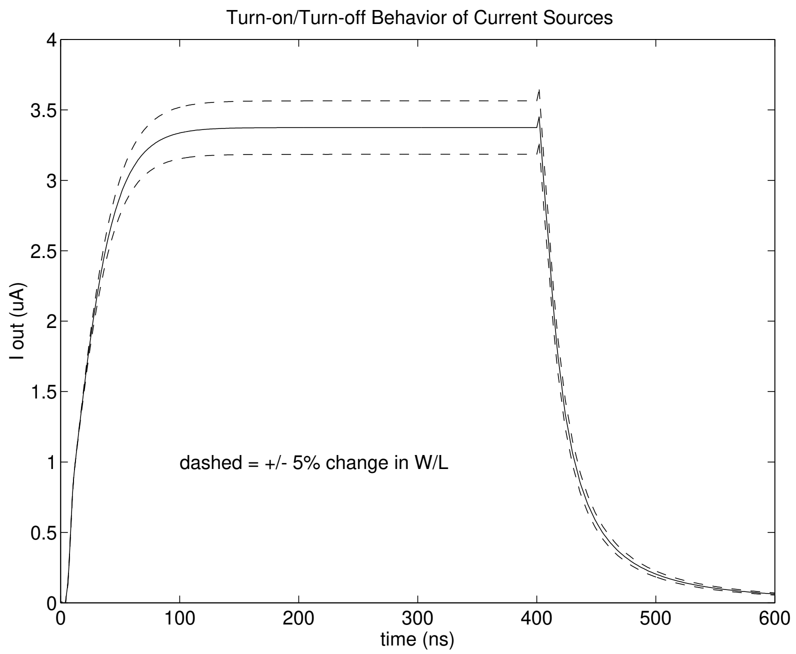


FIGURE 16-4: Transient behavior of switched current sources.

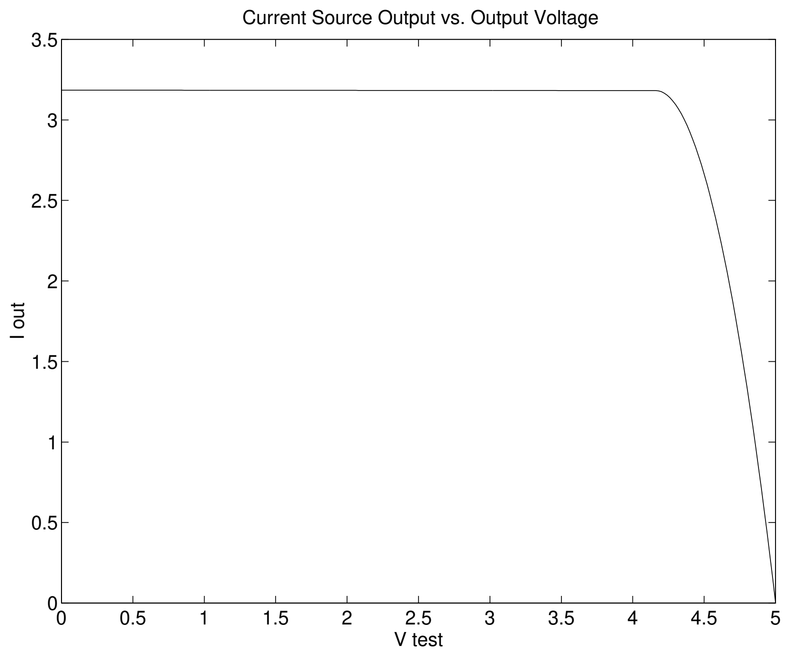


FIGURE 16-5: Output current vs. load voltage for unit cell current sources.

area required by the full array is only about one-third of that needed by the purely digital implementation.

Three considerations were important in determining the design of the current sources used in the unit cells. The first, of course, was to achieve good matching. The second goal was to maximize the range of load voltages over which the sources would behave as ideal elements. Finally, it was important to minimize the output rise time when the sources were switched on in order to increase operating speeds.

Variations in the W/L ratios of the transistors are the only source of mismatch which can be directly influenced by the layout. The width and length of all transistors were thus made as large as possible so that minor process variations in line widths would have a smaller percentage effect on the actual dimensions. Increasing L also helps improve the ideality of the current sources as it decreases channel length modulation and therefore increases the output resistance.

A simple p-type current mirror driven by a biased NMOS transistor was chosen for the design because it requires minimal area and allows a maximum load voltage of $V_{DD} - |V_{gs} - V_t|$. The bias transistor was sized at $W/L = 12\lambda/24\lambda$ to give 3.37μA of current for 1.2V input bias. This value was chosen so that if all 576 current sources of the same type in the 24×24 array were switched on, the maximum output current which would need to be handled would be approximately 2mA. Since each source dissipates 33μW of power when on, the maximum power dissipation in the full array with all current sources on is 38mW. The sources are turned on when the gate voltages on the two transistors connected in series between the bias transistor and the current mirror are brought high. The gate of the transistor closest to the p-fet is connected to the control input, *i.e.,* b or $s \otimes b$, while the gate of the other switch transistor is connected to a signal, labeled here as CLK, which periodically goes high.

The size of the current mirror transistors was determined both by the rise time requirements and by the need to maximize both the load voltage range and the output resistance. Small values of $|V_{gs} - V_t|$ are achieved by making W/L small while large output resistance requires large values of L. Fast rise times, however, are achieved by reducing the gate capacitance of the mirror transistors which needs to be charged when the current source is switched on. An appropriate compromise between these conflicting needs was obtained by choosing $W/L = 6\lambda/20\lambda$. As can be seen from the simulation results plotted in Figures 16-4 and 16-5, the current sources as designed behave ideally up to load voltages of 4.2V, while the output rises to within 1% of its final value in 160ns.

The dashed lines in Figure 16-4 indicate the change in output current for a change of $\pm 5\%$ in the W/L ratio of either the bias transistor or in one of the mirror transistors. The resultant variation in output current is also approximately $\pm 5\%$ from 3.55μA to 3.2μA. By using large values for both W and L in all of these transistors, however, it is hoped that the standard deviation of W/L variations will be much less 5%. Furthermore, to the extent that mismatches in the current sources are random and have zero mean value, summing the outputs from different sources will tend to cancel individual variations. The net effect of mismatch on the precision of the matching circuit should thus be minimal.

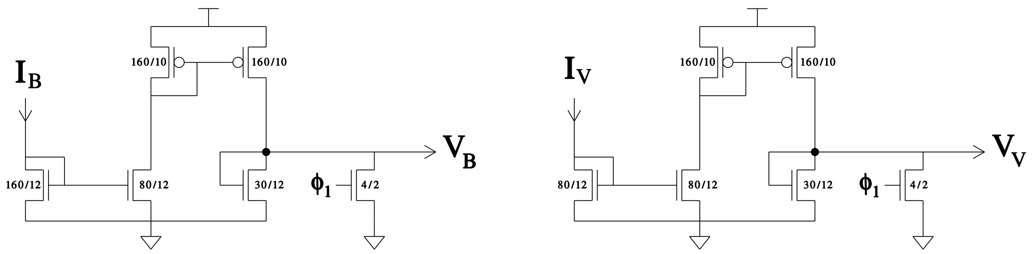
16.2 Global Test Circuits

To complete the matching procedure, the scores generated at each offset must be compared with both the threshold $\alpha_1\|B\|$ and the current minimum score. Since the minimum score must be stored as a voltage, it is best to convert the currents to voltages at this point, as indicated by the circled resistive elements in Figure 16-1, at the same time scaling the total current, I_B, from the base edge pixels by a factor of α_1 with respect to total score current I_V. Representing $\alpha_1\|B\|$ by the voltage $\alpha_1 V_B$ also simplifies the block validation test (14.15), as the values of V_h and V_l can then be supplied as external voltages which can be adjusted as needed.

Analog circuits are thus required outside the unit cells for current scaling and conversion, as well as for comparing and storing voltages. Since the comparator outputs are necessarily binary signals and since the search window offset values must clearly be represented digitally, the remaining functions needed in the procedure of storing candidate offsets and computing the maximum spread must be performed with digital circuits.

16.2.1 Current scaling and conversion

The circuits used for converting the currents I_B and I_V to the voltages $\alpha_1 V_B$ and V_V are shown in Figures 16-6a and 16-6b. The only difference in these circuits is the size of the diode-connected input transistor, which is twice as wide for I_B as for I_V. Since numerous simulations of the matching procedure on test image sequences have shown that best results are obtained by setting α_1 to its maximum value of 0.5, it was chosen to hardwire this value



a.) $\sum I_b$ – voltage scale/conversion circuit. b.) $\sum I_v$ – voltage conversion circuit.

FIGURE 16-6: I–V conversion circuits.

into the circuit. The current into the opposing transistor of the n-type current mirror in Figure 16-6a is thus equal to $I_B/2$, to within the accuracy allowed by the fabrication.

In order for the output voltages to be within the range required by the comparator circuits, discussed in the following section, the currents $I_B/2$ and I_V are then pulled through a second p-type current mirror whose output branch feeds into a diode-connected n-fet serving as the 'resistor'. The fact that this resistor is nonlinear is of no importance to the design since comparing the values of $I_B/2$ and I_V only requires that it be monotonic. It is very important, however, for the two voltage conversion circuits to be well matched. Large geometry transistors were thus used to reduce the percentage mismatch due to line width variations in fabrication, and long channels, $L \geq 10\lambda$, were used on each of the transistors to reduce channel length modulation. Because of the large current-carrying capacity required to accomodate the maximum possible current of $\sim$ 2mA, however, it was necessary to use large W/L ratios, which reduces the output resistance r_o and thus increases the output nonlinearity.

The simulated current divider characteristic for the I_B circuit is shown in Figure 16-7. The curve is linear with slope of exactly 1/2 for input currents up to 1.2mA. Beyond this point, however, the slope decreases considerably as the transistor driving the p-current mirror is pushed into the triode region. Since I_B must be less than $I_{max}/2$, however, in order to pass the block validation test, we are only concerned with the behavior of the divider up to inputs of 1mA.

The I–V characteristic of the complete I_B circuit for inputs of 0 to 1mA is shown in Figure 16-8, while a similar curve for the I_V circuit is given in Figure 16-9, but with a different input axis. For the I_V circuit, the output voltage is plotted as a function of the

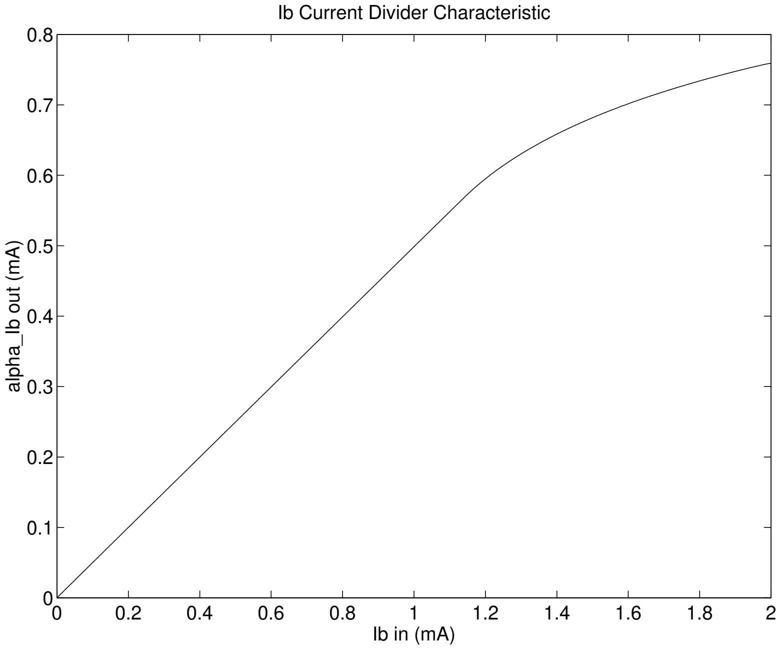


FIGURE 16-7: I_B current divider characteristic.

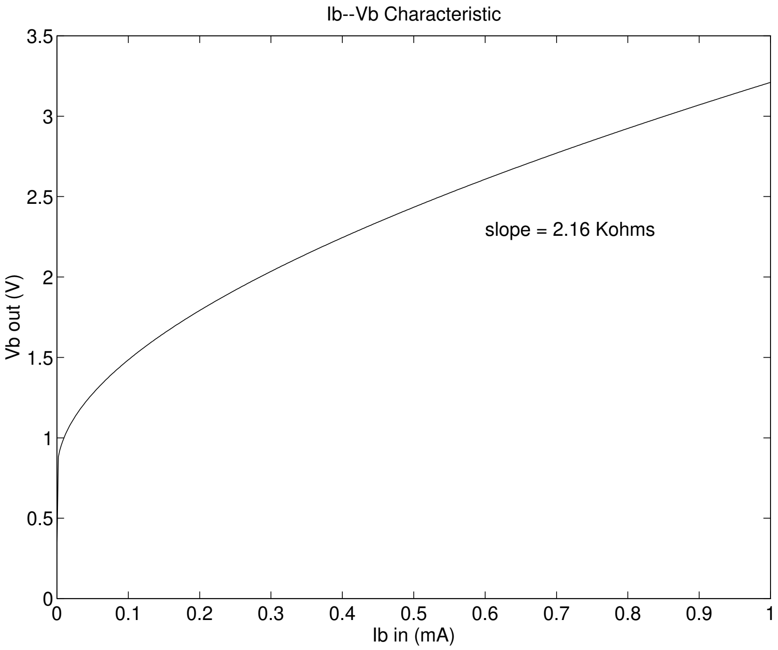


FIGURE 16-8: Simulated I–V characteristic for computing $I_B \rightarrow \alpha_1 V_B$.

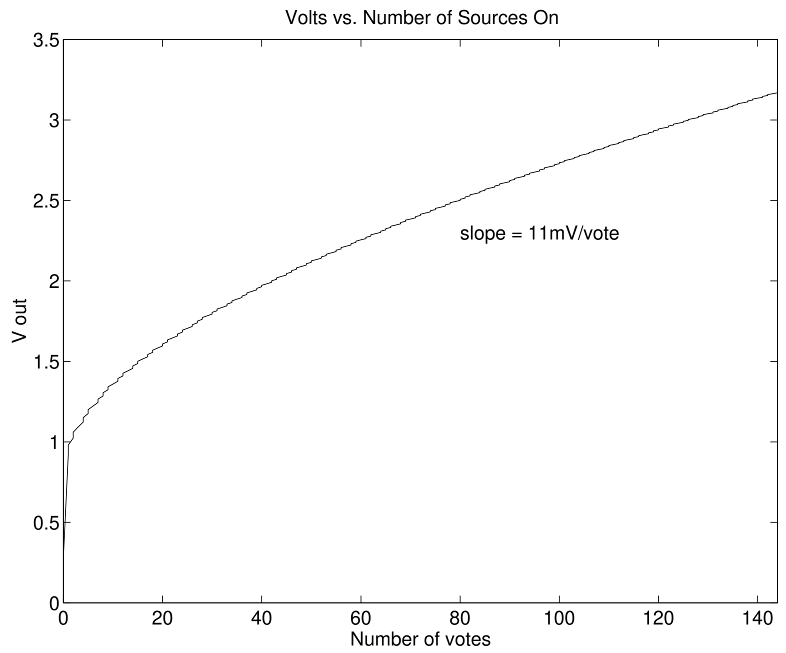


FIGURE 16-9: Output score voltage vs. number of processors with non-matching edge pixels.

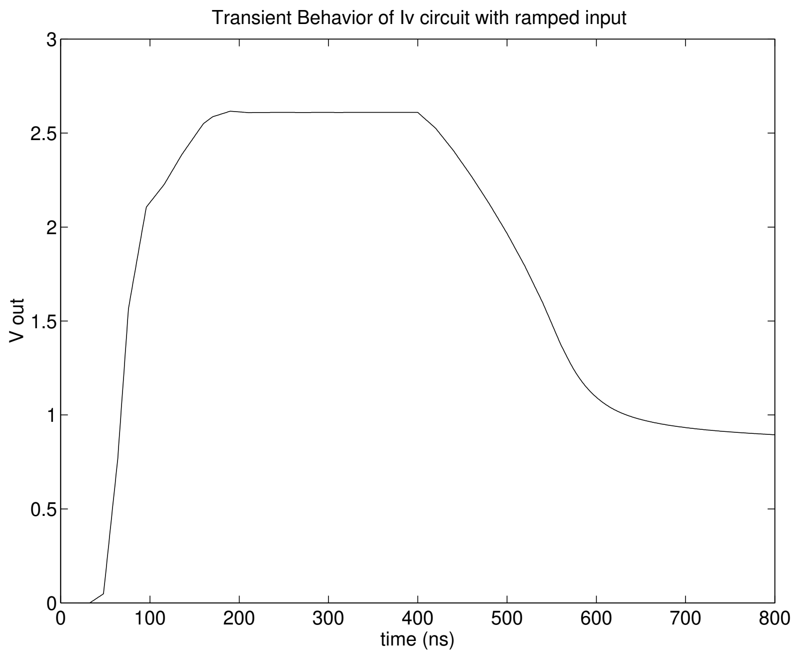


FIGURE 16-10: Rise and fall times of V_V for a 0.3mA input after switching on current sources and before turning on reset transistor.

number of processors responding by dividing the total current by $3.37\mu A$, which is the unit current from one source. For a score to pass the threshold test, it must be the case that $I_V < \alpha_1 I_B \leq P/4$, and hence the maximum number of responding nodes for the score to be a candidate is $576/4 = 144$.

As can be seen from the diagrams, the output voltages for both the I_B and I_V circuits are between 1V and 3.2V when the inputs are within the acceptable range. The slope of the curve for the I_V circuit in its least steep portion is 11mV/vote. Since, according to the precision requirements of Section 14.3, it is necessary to discriminate between scores that are different by more than 3 or 4 votes, the resolution of the comparator circuit should thus be better than 40mV.

The transient behavior of the I_V circuit for a 0.3mA ramped input, corresponding roughly to the rise time characteristic of the node sources, is shown in Figure 16-10. The output voltage is able to follow the rising input without an appreciable delay such that after 170ns, the output is stable at the final voltage. When the node sources are switched off, however, the fall time is much slower since, as the gate voltage on the diode-connected transistor decreases, there is less current to discharge the gate, and when the output voltage reaches the transistor threshold, the drain current becomes negligible. It is thus necessary to connect an additional reset transistor to bring the output back to zero at each cycle. In Figures 16-6a and 16-6b, this reset transistor is shown with its gate connected to the periodic waveform ϕ_1. Timing of the different clock signals used in the matching circuit is discussed in Section 16.3.

16.2.2 Validation and V_{min} tests

The remaining analog circuit needed is the voltage comparator for performing the validation tests and for finding the minimum score. For simplicity a single design, shown inside the dashed box of Figure 16-11, was used for all of the tests. The two input voltages, indicated as In_1 and In_2, are connected to the gates of two identical p-type source followers and 1.13pF capacitors when the signal CLK—which is the same as the one which switches on the node current sources—goes high. The reset transistors are turned on once at the very beginning of the matching procedure to charge the capacitors to an initial high value. This operation is necessary to initialize the comparator in the V_{min} circuit, but is irrelevant for the other validation tests.

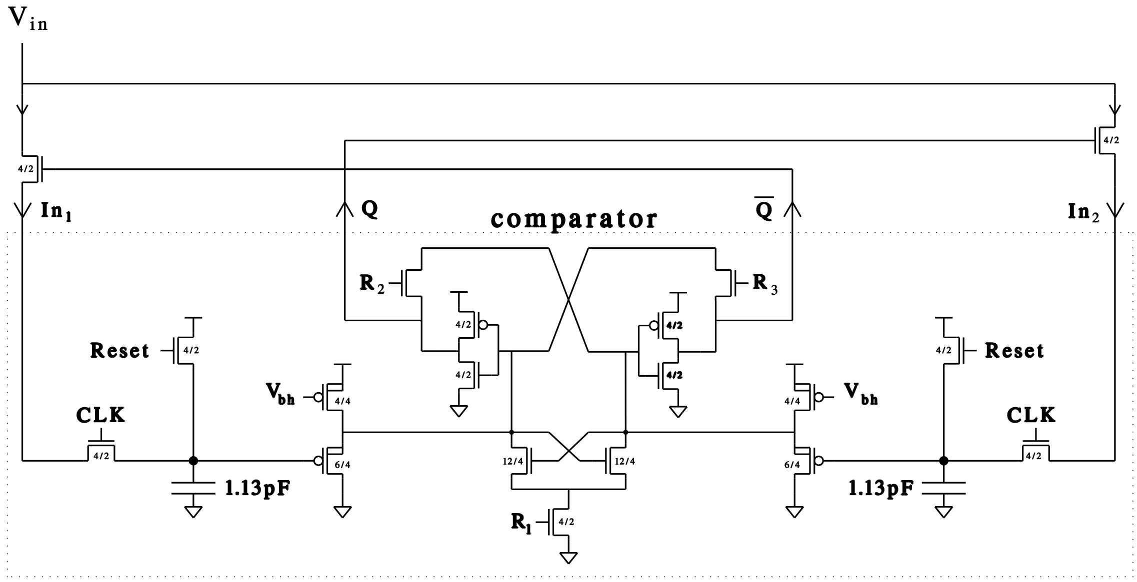


FIGURE 16-11: V_{min} circuit.

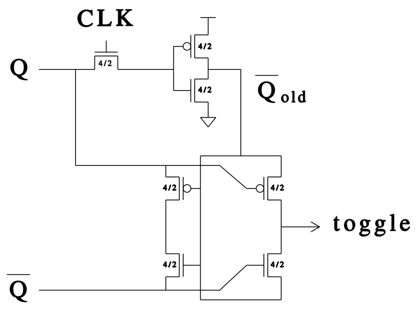


FIGURE 16-12: Toggle circuit for signalling update to V_{min}.

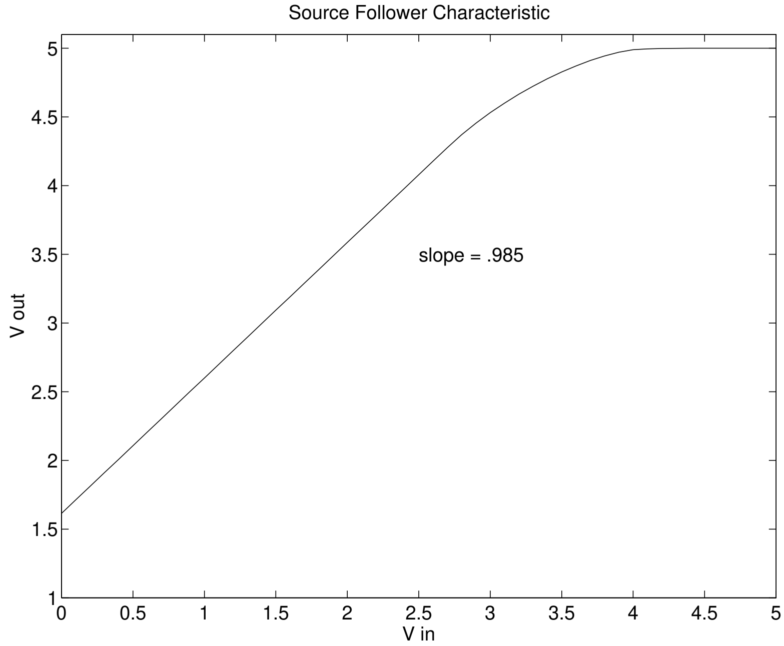


FIGURE 16-13: Input/output characteristic for source followers used in comparator circuits.

The comparator operation is driven by the three clock signals R_1, R_2, and R_3 whose timing relative to the CLK signal is shown in Figure 16-16. Initially, these signals are all high while the output voltages $\alpha_1 V_B$ and V_V rise to their final value. When CLK is brought low, the capacitors and source follower inputs are isolated from the outputs of the I_B and I_V conversion circuits. R_1, R_2, and R_3 are brought low at the same time as CLK so that the output of the source followers can charge the gates of the n-type half latch and the two inverters.

The simulated V_{in}–V_{out} characteristic of the source followers with a bias voltage, V_{bh}, of 3.2V is shown in Figure 16-13. The source followers serve two purposes in the comparator circuit. The first is to shift the input voltages upward so that the minimum input value is at least 1V above the threshold of the latch transistors. The second is to buffer the input values with a stable source that does not require intermediate switches. When R_1 is brought high, the two latch transistors are both turned on. The one with the higher gate voltage, however, will have the higher current, and thus its drain voltage will drop more quickly than that of the other transistor. When one of the drain voltages drops below $|V_{in} - V_t|$, where V_{in} is the input voltage on the source follower connected to that side, the lower p-fet of the source follower will turn off, and the bias transistor will supply the current needed to drive the drain all the way to ground. The opposing latch transistor, whose gate is connected to the drain which goes to 0V, will be turned off, and the output of the source follower connected to its drain will return to its value prior to bringing R_1 high. The capacitors holding the input voltages to the source followers are sized large enough so that any coupling through the gate-source capacitance, C_{gs}, of the lower p-fets will be negligible.

Once the gate voltages of the latch transistors are stable, the signals R_2 and R_3, which connect the outputs of the two inverters to the cross-coupled latch transistors, are brought high. In order to avoid metastable states, these two signals are staggered so that the comparator output cannot 'hang' if the two input voltages are identical. R_2 is brought high before R_3 so that if the drain voltage on the lefthand side is slightly above the inversion threshold, the righthand side will be brought to ground, while if it is below the inversion threshold, the opposite side will go to V_{DD}. Once R_3 is brought high, the comparison operation is complete, and the outputs Q and $\overline{Q}$ are binary signals such that $Q = V_{DD}$ if $In_2 > In_1$ and $\overline{Q} = V_{DD}$ if $In_1 \geq In_2$.

The design of the comparator was based primarily on the needs of the V_{min} circuit and is more than adequate for the validation tests comparing $\alpha_1 V_B$ to V_V, and to the upper and lower limits V_h and V_l. The V_{min} circuit is special, however, in that only one input voltage is supplied. The current minimum score is stored on one of the 1.13pF capacitors and is compared with each input value. If the new value is lower, it must then be stored as the minimum score, and the circuit must signal that the minimum has changed so that the new offset position can be latched.

The circuit configurations for performing these operations are shown in Figures 16-11 and 16-12. The input is gated to one side or the other of the comparator based on the results of the last test by connecting the outputs Q and $\overline{Q}$ to the gates of two pass transistors. If Q is high, the input on the right side was greater than that on the left in the last compare. The next score voltage, V_V, will thus be gated to the In_2 input and will be compared with the value still held on the capacitor on the In_1 side. If the new value is less than the old stored value, $\overline{Q}$ will go high, causing the next input to be gated to the In_1 side, while the new minimum value is stored at In_2.

The toggle circuit shown in Figure 16-12 is used to indicate a change in V_{min}. The output Q is connected to an inverter via a pass transistor controlled by the signal CLK.

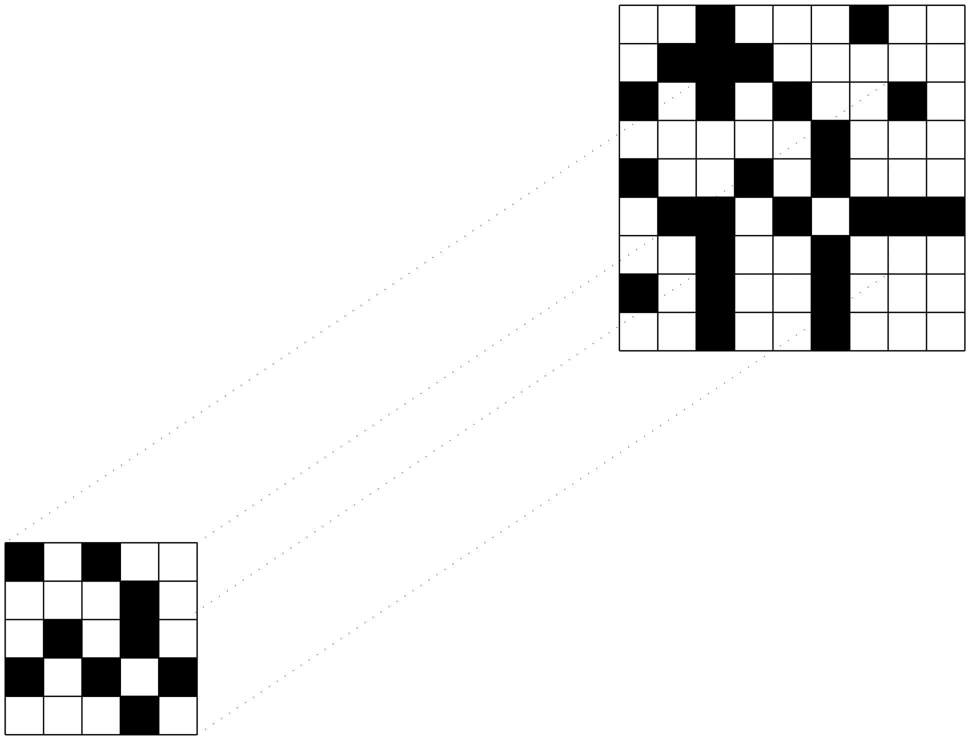


FIGURE 16-14: Test pattern for simulating the 5×5 array.

When CLK is low, which is the case at the end of a compare operation, the output of the inverter is the value of $\overline{Q}$ from the previous compare. Connecting the outputs Q and $\overline{Q}$ to the XOR circuit as shown thus makes $toggle = \overline{Q} \otimes \overline{Q}_{old}$ while CLK is low. If the minimum value changes, the output signal *toggle* will remain high until CLK again goes high.

16.3 Test Pattern Simulation

To test the operation of the complete matching circuit in finding the best offset for an actual edge pattern, a full simulation was conducted with a 5×5 array and a 9×9 search window. The test patterns used in the simulation for the base edge block and the search window are shown in Figure 16-14. At the correct offset, indicated by the dashed lines, the 5×5 edge pattern in the search window matches that of the test block exactly except for one pixel in the lower lefthand corner which is different. To find the best offset, however, the matching circuit must test each of the 25 different offset positions in which the 5×5

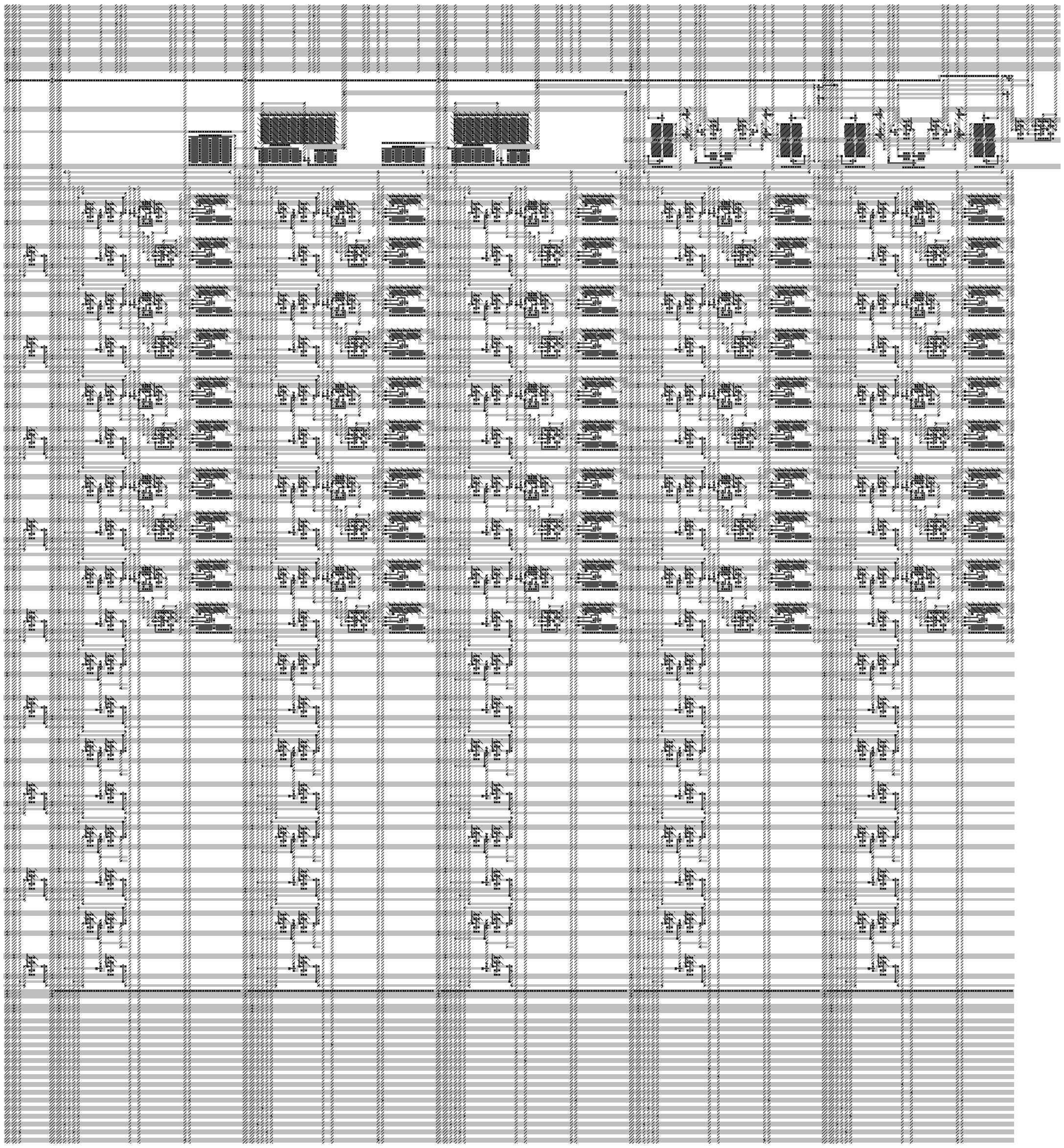


FIGURE 16-15: Layout of 5×5 array.

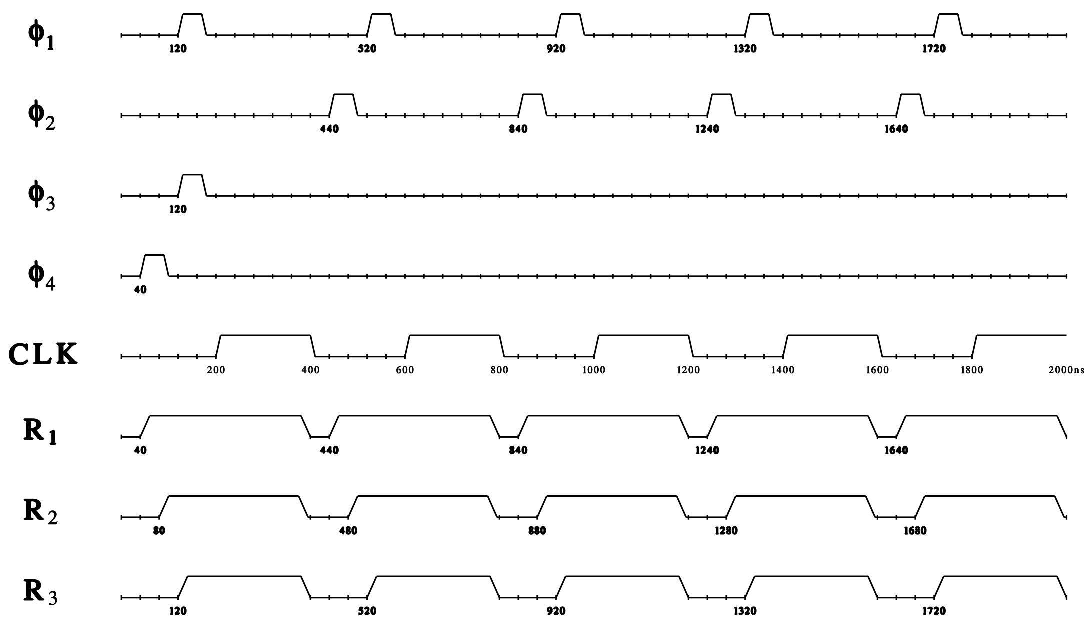


FIGURE 16-16: Clock waveforms.

block is entirely contained in the search window and correctly find the minimum score at the indicated location.

A layout for the full circuit is shown in Figure 16-15 and includes the 5×5 matching array, the I_B and I_V scaling and voltage conversion circuits, the $V_V{-}\alpha_1 V_B$ comparator, and the V_{min} circuit, as well as an additional four rows containing a 4×5 s-shift array so that an entire column of the search window can be loaded in one input cycle. The metal1 lines running horizontally at the base and top of the array carry the necessary control signals and bias voltages for operating the matching circuit. The layout shown measures 2007λ(h) × 1779λ(w) of which 608λ in height are occupied by the optional 4×5 s-shift array and 172λ in height are taken by the conversion and validation circuits.

The waveforms for the principal periodic control signals are shown in Figure 16-16. The simulation was based on a 40ns minimum pulse width with 400ns required to process each offset. The clock signals ϕ_1 and ϕ_2 control vertical movement on the shift register, while clocks ϕ_3 and ϕ_4 control horizontal movement. The shift sequence is such that the block is first aligned with a particular column in the search window, after which each row offset is processed sequentially. Once the last row offset has been processed, the entire search window

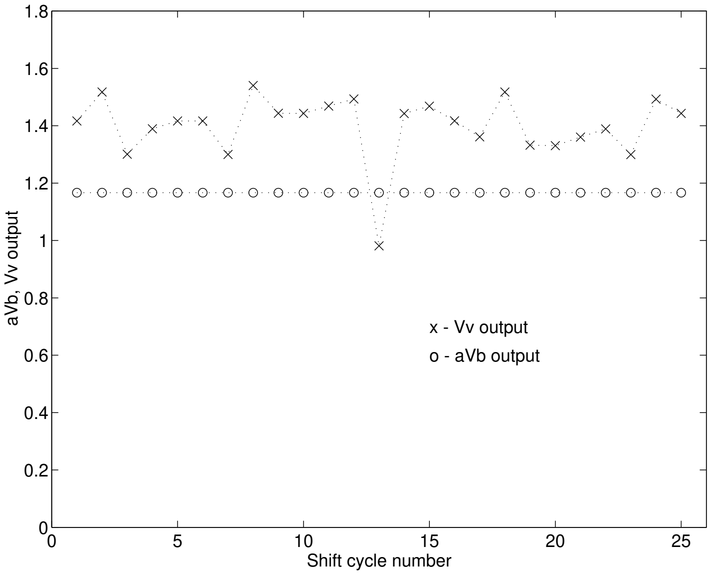


FIGURE 16-17: V_V and $\alpha_1 V_B$ outputs sampled at end of each cycle.

is shifted horizontally by one column with a new column being read in as the farthest one on the block is shifted out. For the present configuration, which has five row offsets per column, there are thus four vertical shifts for every one horizontal shift. The 400ns process time for each offset is due to the 160ns needed to complete each shift—since the clock phases cannot overlap, 200ns to allow adequate settling of the node current sources, and 40ns to reset the clock signals R_1, R_2, and R_3 used in the comparator circuits.

The results of the complete simulation are plotted in Figures 16-17 through 16-19. It is clear from the first diagram, which shows the peak values of $\alpha_1 V_B$ and V_V sampled at the end of each processing cycle, that shift position #13 corresponds to the minimum score. The minimum value of V_V is not only significantly less than all of the other scores, but it is also the only which is less than $\alpha_1 V_B$. The V_V–$\alpha_1 V_B$ comparator output, plotted in Figure 16-18, confirms this fact as offset #13 is the only one to produce a high output. The result of the V_{min} circuit, given by the *toggle* output plotted in Figure 16-19, shows that the minimum score fluctuates a few times at first, but then rises for the last time when the correct offset is reached.

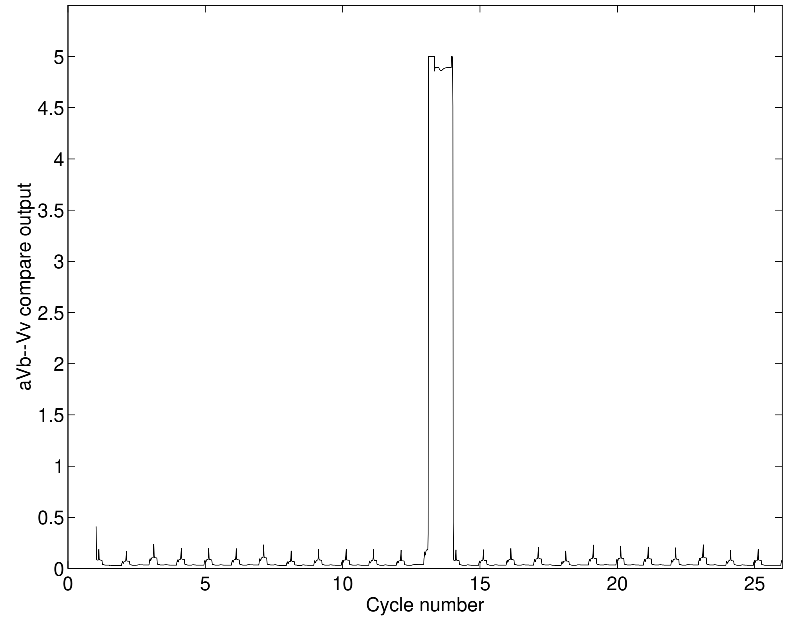


FIGURE 16-18: Response of threshold test circuit at each offset of test pattern.

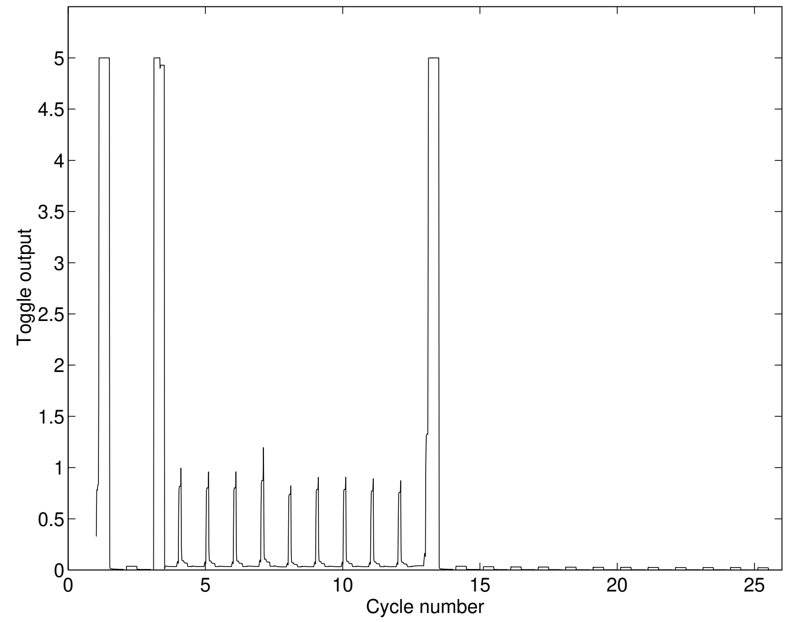


FIGURE 16-19: Response of V_{min} toggle circuit at each offset of test pattern.

16.4 Comparison with Digital Architectures

The simulation results from the test pattern indicate that the mixed analog and digital matching circuit functions correctly according to its design. It requires 400ns to process each offset and can thus search a 50×50 window in 1ms. A 24×24 array processor, including the comparator and V_{min} circuits, would occupy 4432λ(h) × 8160λ(w), based on the layouts shown for the unit cell and 5×5 array. In a 0.8μm ($\lambda = 0.4\mu$m) process, one matching circuit would require 1.77mm × 3.26mm, and thus eight individual circuits could comfortably fit on a single 1cm die.

In the last chapter, it was estimated that the corresponding 24×24 digital processor required 9888λ(h) × 6240λ(w) for the array alone, and at least another 400λ in width to accomodate the summation circuit which adds the results from each row. Digital versions of the comparator and V_{min} circuits were not designed or laid out for this processor, however, the area taken by these components should also be considered. In a 0.8μm process, the array and summation circuits together require at least 3.95mm × 2.66mm, and thus, at best, four processors might fit on a 1cm die. The time required per offset was not estimated for the digital processor. However, it is not at all clear that it could be operated faster than the mixed analog/digital design. Since the same time (200ns) is required by both circuits for shifting the search window, the difference in their speeds is determined by the time required to compute each score and compare it to the minimum value. For the fully digital processor to be faster, the tally circuit and V_{min} comparator would have to be designed to complete these operations in less than the 200ns required by the analog circuit.

The second digital architecture studied is much faster than the first one, as its total processing time is determined by the size of the block being matched and not that of the search window. Furthermore the computations for each edge pixel only require updating the 8-bit accumulators at each array node, and thus the delays are much shorter. The major disadvantages of this design are its large size and the fact that the dimensions of the array limit the maximum search area. It was estimated at the end of the last chapter that one could at best fit a single 30×30 array on a 1cm die.

In summary, the mixed analog/digital matching processor does appear to best meet the needs of the motion estimation system as they have been formulated. This processor has an 8-to-1 area advantage over the digital circuits used for motion estimation in commercial systems (*i.e.,* the second architecture) and does not restrict the search window size. It has, at least, a 2-to-1 area advantage over its direct digital equivalent and can be operated at comparable, or better, speeds. The maximum power dissipation of each 24×24 processor during the 200ns computation cycle is estimated at < 40mW in the worst possible case when all internal current sources are turned on.

Part IV

Conclusions

Chapter 17

Recommendations for Putting It All Together

It is appropriate to conclude by combining the results of the previous chapters into a plan for the complete system design. One possible configuration of a single 'board' motion estimation system, including the multi-scale veto processor and the analog/digital edge matching circuits developed in this thesis, is shown in Figure 17-1. From the analysis of Chapter 8, we know that the spatial resolution of the image sensor must be sufficiently fine so that, given the block size used in the matching procedure, each block can be reasonably approximated as a single point. From tests with real image sequences, it has been seen that the minimum sensor size needed to obtain accurate motion estimates is approximately 256×256 pixels.

Following the recommendations of Chapter 13, it should be possible, with a 0.8μm or better CCD/CMOS process, to build a 256×256 MSV focal-plane processor which will perform both imaging and edge detection. The advantage of the focal-plane processor is that the signal degradation incurred in loading the image one pixel at a time through the fill-and-spill input structure and transferring the charges over the length of the gate array is avoided. The improved design does not, however, offer any savings in total processing time, as the time not spent in loading the image is amply made up for in sequentially performing the differencing and threshold tests for each row and column. Assuming the processor is operated at 5MHz, and including a normal image acquisition time of 1ms, it should take approximately 5ms for the edge detector to process each frame and deliver the binary edge outputs to the memory buffers.

As discussed in Section 11.5.2, most of the power required to operate a CCD array is

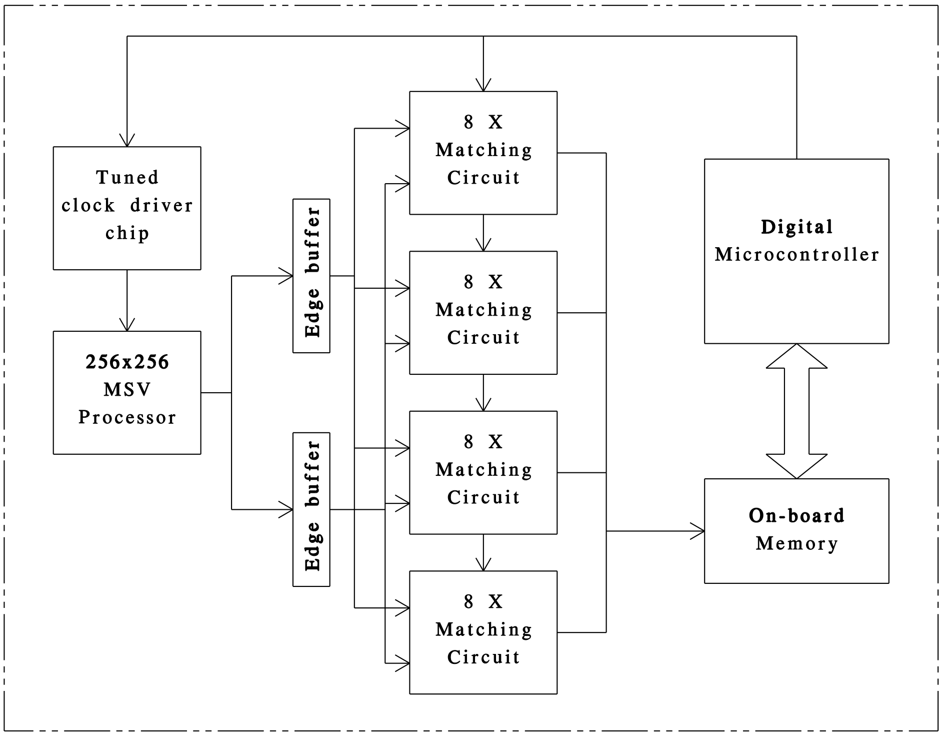


FIGURE 17-1: Single 'board' motion estimation system.

dissipated in the clock drivers. To minimize power requirements, a separate chip containing specially designed drivers tuned for the capacitive and inductive loads of the system, should be included to supply the clock waveforms for the MSV procesor. Tuned drivers can reduce power dissipation by a factor of $1/Q_f$, where Q_f is the quality factor of the oscillator.

To obtain accurate motion estimates, it is essential to try to match a large number of blocks in order to ensure that a sufficient number of correspondence points will be found. In the numerous tests performed with real image sequences, it was often observed that the hit rate, *i.e.,* the fraction of blocks having acceptable matches, was very low in ordinary scenes due to a combination of the lack of distinctive features and the frequent occurrence of repeating patterns which give multiple candidate matches. It was typical to obtain only 30–50 correspondences out of more than 400 blocks tested.

It would not be practical to include 400 matching circuits on the motion system board both for reasons of size and of power consumption—even if 8 individual circuits are contained in each chip. The proposed configuration of Figure 17-1 holds four chips, such that a total of 32 blocks can be matched simultaneously. By pipelining the matching operations, 448 blocks can be tested in 14 cycles. At the end of Chapter 16, it was calculated that a 50×50 window could be searched in 1ms. Taking this as the average window size, the entire matching process can thus be completed in 14ms plus some additional time to manage operational overhead. Assuming that imaging and edge detection require 5ms, and estimating that roughly 10ms are needed to both solve the motion equations and perform other housekeeping tasks, the complete system should be able to process each image pair in just over 29ms and thus achieve a throughput of slightly better than 30 frames/sec.

The final important component of the system is the micro-controller which sequences the operations on the board and also performs the motion computations. This component was hardly covered in this thesis as there are many commercially available digital processors which could be used for this purpose. It is important, however, to choose one which is adequate for the task but does not include unnecessary functions which will increase power consumption, and possibly the cost of the processor itself.

There are, of course, many issues left to be resolved to complete the motion system. The proposed new design of the MSV edge detector should be fabricated in a better-controlled and smaller scale CCD process, and then tested to evaluate its performance. Full-size matching circuits should also be fabricated to verify their actual performance with that predicted by simulation. The tuned clock driver chip for controlling the MSV processor must be designed, and an adequate, but minimal, microprocessor should be chosen to control the system.

Constructing the full system is outside the scope of any one thesis. The major contribution of this research has been to thoroughly analyze the theoretical and algorithmic constraints imposed on the system and, given these constraints, to carefully study the design of its two most important components. The results developed in this thesis have thus established the foundation which will serve as the basis of further work for building the complete motion estimation system according to the goals set for its design.

Bibliography

[1] S. Finsterwalder. Die Geometrischen Grundlagen der Photogrammetrie. *Jahresbericht Deutscher Mathematik*, 6:1–44, 1899.

[2] G. Hauck. Neue Konstruktionen der Perspektive und Photogrammetrie. *Crelle J. F. Math.*, pages 1–35, 1883.

[3] Berthold K. P. Horn. Relative orientation. *International Journal of Computer Vision*, 4(1):59–78, 1990.

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

[5] E. Kruppa. Zur Ermittlung eines Objektes aus zwei Perspectiven mit innerer Orientierung. *Sitz.-Ber. Akad. Wiss.*, Math. Naturew. Kl. Abt. IIa(122):1939–1948, 1913.

[6] Konstantinos Daniilidis and Hans Helmut Nagel. Analytical results on error sensitivity of motion estimation from two views. *Image and Vision Computing*, 8(4):297–303, November 1990.

[7] Olivier D. Faugeras and Steve Maybank. Motion from point matches: multiplicity of solutions. *International Journal of Computer Vision*, 4:225–246, 1990.

[8] H. C. Longuet-Higgins. Configurations that defeat the 8-point algorithm. In S. Ullman and W. Richards, editors, *Image Understanding 1984*. Ablex, Norwood, NJ, 1984.

[9] Toni Schenk, Jin-Cheng Li, and Charles Toth. Towards an autonomous system for orienting digital stereopairs. *Photogrammetric Engineering and Remote Sensing*, 57(8):1057–1064, August 1991.

[10] Ted J. Broida and Rama Chellapa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):497–513, June 1991.

[11] Joachim Heel. *Temporal Surface Reconstruction.* PhD thesis, Massachusetts Institute of Technology, 1991.

[12] Minas E. Spetsakis and John (Yiannis) Aloimonos. Optimal motion estimation. In *IEEE Workshop on Visual Motion*, pages 229–237, Irvine, CA, 1989.

[13] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London B*, 207:405–426, 1979.

[14] C. Tomasi and T. Kanade. Factoring image sequences into shape and motion. In *IEEE Workshop on Visual Motion*, pages 21–28, Nassau Inn, Princeton, NJ, October 1991.

[15] T. S. Huang and C. H. Lee. Motion and structure from orthographic projections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):536–540, 1989.

[16] Roger Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 364–374, Miami Beach, FL, June 1986.

[17] Reimar K. Lenz and Roger Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3-d machine vision metrology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):713–720, September 1988.

[18] Juyang Weng, Paul Cohen, and Marc Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, October 1992.

[19] Olivier D. Faugeras, Q.-T. Luong, and Steve J. Maybank. Camera self-calibration: Theory and experiments. In *Proceedings of the Second European Conference on Computer Vision*, pages 321–334, Santa Margherita Ligure, Italy, May 1992.

[20] Richard I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proceedings of the Second European Conference on Computer Vision*, pages 579–587, Santa Margherita Ligure, Italy, May 1992.

[21] Lisa Dron. Dynamic camera self-calibration from controlled motion sequences. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, New York, NY, 1993. also appears in the *Proceedings of the 1993 DARPA Image Understanding Workshop*.

[22] Homer H. Chen. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):530–541, June 1991.

[23] O. D. Faugeras, F. Lustman, and G. Toscani. Motion and structure from point and line matches. In *Proceedings of the International Conference on Computer Vision*, pages 25–34, London, England, 1987.

[24] Minas E. Spetsakis and John (Yiannis) Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4:171–183, 1990.

[25] B. L. Yen and T. S. Huang. Determining 3-D motion and structure of a rigid body using straight line correspondences. In T. S. Huang, editor, *Image Sequence Processing and Dynamic Scene Analysis*. Springer-Verlag, New York, NY, 1983.

[26] E. H. Thompson. A rational algebraic formulation of the problem of relative orientation. *Photogrammetric Record*, 3(14):152–159, October 1959.

[27] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[28] Roger Y. Tsai and Thomas S. Huang. Uniqueness and estimation of 3-D motion parameters and surface structures of rigid objects. In S. Ullman and W. Richards, editors, *Image Understanding 1984*. Ablex, Norwood, NJ, 1984.

[29] Jia-Qi Fang and Thomas S. Huang. Some experiments on estimating the 3-D motion parameters of a rigid body from two consecutive image frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(5):545–554, September 1984.

[30] Juyang Weng, Narendra Ahuja, and Thomas S. Huang. Matching two perspective views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):806–825, August 1992.

[31] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. In S. Ullman and W. Richards, editors, *Image Understanding 1984*. Ablex, Norwood, NJ, 1984.

[32] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.

[33] Brian G. Schunck. Image flow: Fundamentals and future research. In *Proceedings of IEEE Computer Society Conference on Pattern Recognition and Image Processing*, pages 560–571, 1985.

[34] D. W. Murray and B. F. Buxton. Scene segmentation from visual motion using global optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):147–163, March 1987.

[35] S. Geman and D. Geman. Stochastic relaxation, Gibbs Distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[36] D. J. Heeger, A. D. Jepson, and E. P. Simoncelli. Recovering observer translation with center-surround operators. In *IEEE Workshop on Visual Motion*, pages 95–100, Nassau Inn, Princeton, NJ, October 1991.

[37] M. Subbarao. Solution and uniqueness of image flow equations for rigid curved surfaces in motion. In *Proceedings of the International Conference on Computer Vision*, pages 687–692, London, England, 1987.

[38] A. M. Waxman and K. Wohn. Image flow theory: A framework for 3-D inference from time-varying imagery. In C. Brown, editor, *Advances in Computer Vision*. Erlbaum, Hillside, NJ, 1987.

[39] Shariar Negahdaripour. Ambiguities of a motion field. A.I. Memo 940, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, January 1987.

[40] Shariar Negahdaripour. Multiple interpretations of the shape and motion of objects from two perspective views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1025–1039, November 1990.

[41] B. K. P. Horn and E. J. Weldon Jr. Direct methods for recovering motion. *International Journal of Computer Vision*, 2:51–76, 1988.

[42] S. Negahdaripour and B. K. P. Horn. Direct passive navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):168–176, 1987.

[43] S. Negahdaripour and B. K. P. Horn. Direct passive navigation: Analytical solution for planes. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1157–1163, San Francisco, CA, 1986.

[44] S. Negahdaripour and B. K. P. Horn. Using depth-is-positive constraint to recover translational motion. In *IEEE Workshop on Computer Vision*, pages 138–144, Miami Beach, FL, November 30–December 2 1987.

[45] S. Negahdaripour and B. K. P. Horn. A direct method for locating the focus of expansion. A.I. Memo 939, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, January 1987.

[46] M. Ali Taalebinezhaad. Direct recovery of motion and shape in the general case by fixation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):847–853, August 1992.

[47] M. Ali Taalebinezhaad. *Robot Motion Vision by Fixation*. PhD thesis, Massachusetts Institute of Technology, 1992.

[48] Alessandro Verri and Tomaso Poggio. Against quantitative optical flow. In *Proceedings of the International Conference on Computer Vision*, pages 171–180, London, England, 1987.

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

[50] Michael A. Gennert and Shariar Negahdaripour. Relaxing the brightness constancy assumption in computing optical flow. A.I. Memo 975, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, June 1987.

[51] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, pages 283–310, 1989.

[52] B. K. P. Horn. Parallel networks for machine vision. A.I. Memo 1071, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, December 1988.

[53] John E. Tanner and Carver A. Mead. A correlating optical motion detector. In *1984 MIT Conference on Very Large Scale Integration*, pages 57–64, 1984.

[54] John E. Tanner. *Integrated Optical Motion Detection.* PhD thesis, California Institute of Technology, 1986.

[55] Carver A. Mead. *Analog VLSI and Neural Systems.* Addison-Wesley, Reading, MA, 1989.

[56] Timothy Horiuchi, Wyeth Bair, Brooks Bishofberger, Andrew Moore, and Christof Koch. Computing motion using analog VLSI vision chips: An experimental comparison among four approaches. *International Journal of Computer Vision*, 8(3):203–216, September 1992.

[57] J. L. Wyatt, Jr., C. L. Keast, M. Seidel, D. Standley, B. K. P. Horn, T. F. Knight, C. G. Sodini, H-S. Lee, and T. Poggio. Analog VLSI systems for image acquisition and fast early vision processing. *International Journal of Computer Vision*, 8(3):217–230, September 1992.

[58] M. Gottardi and Woodward Yang. A CCD/CMOS image motion sensor. In *IEEE International Solid State Circuits Conference*, San Fransisco, CA, February 1993.

[59] Ignacio S. McQuirk. Direct methods for estimating the focus of expansion in analog VLSI. Master's thesis, Massachusetts Institute of Technology, 1991.

[60] Michael A. Gennert. Brightness-based stereo matching. In *Proceedings of the International Conference on Computer Vision*, pages 139–143, Tampa, FL, 1988.

[61] Harry L. Van Trees. *Detection, Estimation, and Modulation Theory.* John Wiley & Sons, New York, NY., 1968.

[62] Daniel I. Barnea and Harvey F. Silverman. A class of algorithms for fast digital image registration. *IEEE Transactions on Computers*, 21(2):179–186, February 1972.

[63] William M. Silver. *Alignment and Gauging Using Normalized Correlation Search.* Cognex Corporation, Needham, MA. Internal Report.

[64] Didier Le Gall. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):47–58, April 1991.

[65] J. Mikko Hakkarainen. *A Real-Time Stereo Vision System in CCD/CMOS Technology*. PhD thesis, Massachusetts Institute of Technology, 1992.

[66] D. Marr. *Vision*. Freeman, San Francisco, CA, 1982.

[67] Leslie Novak. Correlation algorithms for radar map matching. *IEEE Transactions on Aerospace and Electronic Systems*, 14(4):641–648, July 1978.

[68] Robert Y. Wong. Sequential scene matching using edge features. *IEEE Transactions on Aerospace and Electronic Systems*, 14(1):128–140, January 1978.

[69] H. K. Nishihara. A practical real-time imaging stereo matcher. *Optical Engineering*, 23:536–545, 1984.

[70] D. Marr and E. C. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London*, B(207):187–217, 1980.

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

[72] Nicholas Ayache and Bernard Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *International Journal of Computer Vision*, pages 107–131, 1987.

[73] Margaret M. Fleck. A topological stereo matcher. *International Journal of Computer Vision*, 6:197–226, 1991.

[74] Lisa Dron. The multi-scale veto model: A two-stage analog network for edge detection and image reconstruction. *International Journal of Computer Vision*, 11(1):45–61, 1993.

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

[76] B. K. P. Horn. The Binford-Horn LINE-FINDER. A.I. Memo 285, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, July 1971.

[77] V. Torre and T. Poggio. On edge detection. A.I. Memo 768, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, August 1984.

[78] A. P. Witkin. Scale-space filtering. In *Proceedings of the 8th International Joint Conference on A.I.*, pages 1019–1022, Karlsruhe, West Germany, 1983.

[79] R. C. Gonzalez and P. Wintz. *Digital Image Processing.* Addison-Wesley, Reading, MA, 1987.

[80] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, San Francisco, CA, 1985.

[81] A. Blake and A. Zisserman. *Visual Reconstruction.* MIT Press, Cambridge, MA, 1987.

[82] J. Batali. A vision chip. A.I. Memo 869, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, May 1981.

[83] T. F. Knight Jr. *Design of an Integrated Optical Sensor with On-Chip Prepocessing.* PhD thesis, Massachusetts Institute of Technology, 1983.

[84] J. Harris, C. Koch, J. Luo, and J. L. Wyatt, Jr. Resistive fuses: Analog hardware for detecting discontinuities in early vision. In C. Mead and M. Ismail, editors, *Analog VLSI Implementation of Neural Systems*, pages 27–55. Kluwer Academic Publishers, 1989.

[85] P. Yu, S. J. Decker, H-S. Lee, C. G. Sodini, and J. L. Wyatt, Jr. CMOS resistive fuses for image smoothing and segmentation. *Journal of Solid State Circuits*, 27(4):545–553, 1992.

[86] Craig L. Keast. *An Integrated Image Acquisition, Smoothing and Segmentation Focal Plane Processor.* PhD thesis, Massachusetts Institute of Technology, 1992.

[87] M. H. DeGroot. *Probability and Statistics.* Addison-Wesley, Reading, MA, 1986.

[88] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4:629–642, April 1987.

[89] Berthold K. P. Horn. Motion fields are hardly ever ambiguous. *International Journal of Computer Vision*, 1:259–274, 1987.

[90] Juyang Weng, Narendra Ahuja, and Thomas S. Huang. Optimal motion and structure estimation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 144–152, San Diego, CA, June 1989.

[91] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 1989.

[92] Dieter K. Schroder. Advanced MOS devices. In Gerold W. Neudeck and Robert F. Pierret, editors, *Modular Series on Solid State Devices*. Addison-Wesley Publishing Co., 1987.

[93] D. F. Barbe and S. B. Campana. Imaging arrays using the charge-coupled concept. In *Advances in Image Pickup and Display*, volume 3, pages 171–296, New York, 1977. Academic Press.

[94] Craig L. Keast. A CCD/CMOS process for integrated image acquisition and early vision signal processing. Master's thesis, Massachusetts Institute of Technology, 1989.

[95] Hsing-San Lee and Lawrence G. Heller. Charge-control method of charge-coupled device transfer analysis. *IEEE Transactions on Electron Devices*, 19(12):1270–1279, 1972.

[96] Alan B. Grebene. *Bipolar and MOS Analog Integrated Circuit Design*. John Wiley & Sons, New York, 1984.

[97] J. D. E. Beynon and D. R. Lamb. *Charge-coupled devices and their applications*. McGraw-Hill (UK), London, 1980.

[98] Paul R. Gray and Robert G. Meyer. *Analysis and Design of Analog Integrated Circuits*. John Wiley & Sons, Inc., New York, third edition, 1993.

[99] Lance A. Glasser and Daniel W. Dobberpuhl. *The Design and Analysis of VLSI Circuits*. Addison-Wesley, Reading, MA, 1985.

[100] Loral Fairchild Imaging Sensors, Milpitas, CA. *1991 Loral Fairchild CCD Imaging Databook*, 1991.

[101] G. W. Taylor and A. F. Tasch, Jr. On the measurement of the channel potential in charge-coupled-device structures. *Solid-State Electronics*, 20:473–474, 1977.

[102] John P. Hayes. *Computer Architecture and Organization.* McGraw-Hill, New York, second edition, 1988.

Appendices

Appendix A

Quaternion Algebra

Quaternions are vectors in $\mathbb{R}^4$ which may be thought of as the composition of a scalar and 'vector' part [4].

$$\mathring{a} = (a_0, \mathbf{a}) \tag{A.1}$$

where $\mathbf{a} = (a_x, a_y, a_z)^{\mathrm{T}}$ is a vector in $\mathbb{R}^3$.
Conjugation is defined by

$$\mathring{a}^* = (a_0, -\mathbf{a}) \tag{A.2}$$

and the multiplication of two quaternions is by

$$\mathring{a}\mathring{b} = (a_0 b_0 - \mathbf{a} \cdot \mathbf{b}, a_0 \mathbf{b} + b_0 \mathbf{a} + \mathbf{a} \times \mathbf{b}) \tag{A.3}$$

Quaternion multiplication is associative, but not commutative. The identity with respect to multiplication is

$$\mathring{e} = (1, \mathbf{0}) \tag{A.4}$$

In all other respects, quaternions may be treated as ordinary vectors. The transpose, dot product, and multiplication by a matrix are defined in the usual manner. One way to express the same operation in (A.3) is by a matrix-vector multiplication using equivalent quaternion matrices. For example,

$$\mathring{a}\mathring{b} = \begin{pmatrix} a_0 & -a_x & -a_y & -a_z \\ a_x & a_0 & -a_z & a_y \\ a_y & a_z & a_0 & -a_x \\ a_z & -a_y & a_z & a_0 \end{pmatrix} \mathring{b} \equiv \mathcal{A}\mathring{b} \tag{A.5}$$

and

$$\mathring{a}\mathring{b} = \begin{pmatrix} b_0 & -b_x & -b_y & -b_z \\ b_x & b_0 & b_z & -b_y \\ b_y & -b_z & b_0 & b_x \\ b_z & b_y & -b_z & b_0 \end{pmatrix} \mathring{a} \equiv \underline{\mathcal{B}}\mathring{a} \tag{A.6}$$

$\mathcal{A}$ is referred to as the *left* quaternion matrix associated with $\mathring{a}$, and $\underline{\mathcal{B}}$ is referred to as the *right* quaternion matrix associated with $\mathring{b}$. Quaternion matrices are useful for re-arranging formulas into more convenient expressions. Using either (A.3) or the matrix formulations (A.5) and (A.6), the following identities can be easily shown

$$(\mathring{a}\mathring{a}^*) = (\mathring{a} \cdot \mathring{a})\mathring{e} \tag{A.7}$$

$$(\mathring{a}\mathring{b})^* = \mathring{b}^*\mathring{a}^* \tag{A.8}$$

$$(\mathring{a}\mathring{q}) \cdot (\mathring{b}\mathring{q}) = (\mathring{a} \cdot \mathring{b})(\mathring{q} \cdot \mathring{q}) \tag{A.9}$$

$$(\mathring{a}\mathring{q}) \cdot \mathring{b} = \mathring{a} \cdot (\mathring{b}\mathring{q}^*) \tag{A.10}$$

A vector in $\mathbb{R}^3$ can be represented by a quaternion with zero scalar part. If $\mathring{r}$, $\mathring{\ell}$, and $\mathring{b}$ are quaternions with zero scalar part then

$$\mathring{r}^* = -\mathring{r} \tag{A.11}$$

$$\mathring{r} \cdot \mathring{\ell} = \mathbf{r} \cdot \boldsymbol{\ell} \tag{A.12}$$

$$\mathring{r}\mathring{\ell} = (-\mathbf{r} \cdot \boldsymbol{\ell}, \mathbf{r} \times \boldsymbol{\ell}) \tag{A.13}$$

$$\mathring{b} \cdot (\mathring{r}\mathring{\ell}) = \mathbf{b} \cdot (\mathbf{r} \times \boldsymbol{\ell}) \tag{A.14}$$

The last identity above is the quaternion representation of the triple product.

The reason that quaternions are so useful is because of the simplicity with which rotation about an arbitrary axis can be represented. A unit quaternion is one whose magnitude, defined as the square root of its dot product with itself, is unity.

$$\mathring{q} \cdot \mathring{q} = 1 \tag{A.15}$$

Every unit quaternion represents a rotation in $\mathbb{R}^3$ of an angle θ about an axis $\hat{\boldsymbol{\omega}}$ in the sense that

$$\mathring{q} = \left(\cos\frac{\theta}{2}, \hat{\boldsymbol{\omega}} \sin\frac{\theta}{2}\right) \tag{A.16}$$

The rotational transformation of a vector $\boldsymbol{\ell}$ is found from

$$\mathring{\ell}' = \mathring{q}\mathring{\ell}\,\mathring{q}^* \tag{A.17}$$

where $\mathring{\ell}'$ is the quaternion representation of the rotated vector, $\boldsymbol{\ell}'$. Using quaternion matrices, we can also write (A.17) as

$$\begin{aligned} \mathring{\ell}' &= \boldsymbol{\mathcal{Q}}\underline{\boldsymbol{\mathcal{Q}}}^*\mathring{\ell} \\ &= \begin{pmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{R} \end{pmatrix} \end{aligned} \tag{A.18}$$

where $\mathbf{R}$ is an orthonormal rotation matrix. Expanding terms we find

$$\mathbf{R} = \begin{pmatrix} (q_0^2 + q_x^2 - q_y^2 - q_z^2) & 2(q_xq_y - q_0q_z) & 2(q_xq_z + q_0q_y) \\ 2(q_xq_y + q_0q_z) & (q_0^2 - q_x^2 + q_y^2 - q_z^2) & 2(q_yq_z - q_0q_x) \\ 2(q_xq_z - q_0q_y) & 2(q_yq_z + q_0q_x) & (q_0^2 - q_x^2 - q_y^2 + q_z^2) \end{pmatrix} \tag{A.19}$$

If we are given an orthonormal rotation matrix $\mathbf{R}$, the corresponding unit quaternion can be found by noting that

$$q_0^2 = \cos^2\frac{\theta}{2} = \frac{\mathrm{Tr}(\mathbf{R}) + 1}{4} \tag{A.20}$$

and by solving the eigenvector equation

$$\mathbf{R}\hat{\boldsymbol{\omega}} = \hat{\boldsymbol{\omega}} \tag{A.21}$$

Additional results on quaternions and their properties can be found in [88] and [4].

Appendix B

Special Integrals

In the analysis of the numerical stability and error sensitivity of the motion estimates, it is necessary to compute the following integrals:

$$\int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^2 \boldsymbol{\ell}' \, \xi \, d\xi \, d\alpha \tag{B.1}$$

$$\int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^4 \, \xi \, d\xi \, d\alpha \tag{B.2}$$

$$\int_0^D \int_0^{2\pi} \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha \tag{B.3}$$

$$\int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^2 \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha \tag{B.4}$$

$$\int_0^D \int_0^{2\pi} (\mathbf{u} \cdot \boldsymbol{\ell}') \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha \tag{B.5}$$

$$\int_0^D \int_0^{2\pi} (\mathbf{u}_1 \cdot \boldsymbol{\ell}')(\mathbf{u}_2 \cdot \boldsymbol{\ell}') \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha \tag{B.6}$$

where $\mathbf{u}$, $\mathbf{u}_1$, and $\mathbf{u}_2$ are arbitrary vectors and

$$\boldsymbol{\ell}' = \mathbf{R}\boldsymbol{\ell} \tag{B.7}$$

with $\boldsymbol{\ell}$ being the vector from the center of projection to a point $(\xi \cos \alpha, \xi \sin \alpha)$ on the image plane in the left camera system

$$\boldsymbol{\ell} = \begin{pmatrix} \xi \cos \alpha \\ \xi \sin \alpha \\ 1 \end{pmatrix} \tag{B.8}$$

as shown in Figure 8-1.

We assume that the vectors $\boldsymbol{\ell}$ are dense over the cone defined by the field of view so that we can replace summations by integrals and thereby obtain an analytical expression. In this Appendix, I will derive the solutions to the integrals (B.1)–(B.6) which are used in Chapter 8.

B.1 $\int\int |\ell'|^2 \ell' \, \xi \, d\xi \, d\alpha$

Since $|\boldsymbol{\ell}'|^2 = |\boldsymbol{\ell}|^2$ we have from (B.8)

$$|\boldsymbol{\ell}|^2 = \boldsymbol{\ell}^{\mathrm{T}} \boldsymbol{\ell} = 1 + \xi^2 \tag{B.9}$$

The rotation matrix $\mathbf{R}$ is constant and can therefore be taken outside the integral. We can thus write

$$\begin{aligned} \int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^2 \boldsymbol{\ell}' \, \xi \, d\xi \, d\alpha &= \mathbf{R} \int_0^D \int_0^{2\pi} (1 + \xi^2) \begin{pmatrix} \xi \cos \alpha \\ \xi \sin \alpha \\ 1 \end{pmatrix} \xi \, d\xi \, d\alpha \\ &= \mathbf{R} \left(2\pi \int_0^D (\xi + \xi^3) \hat{z} \, d\xi \right) \end{aligned} \tag{B.10}$$

The rotation is now taken into account by writing $\mathbf{R}$ as

$$\mathbf{R} = (\, \hat{v}_1 \, \hat{v}_2 \, \hat{v}_3 \,) \tag{B.11}$$

where the vectors $\hat{v}_1$, $\hat{v}_2$, and $\hat{v}_3$ form an orthonormal triad. It is useful to note that $\hat{v}_3 = \mathbf{R}\hat{z}$ represents the rotation of the optical axis in the left camera system. We thus have

$$\int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^2 \boldsymbol{\ell}' \, \xi \, d\xi \, d\alpha = \pi D^2 \left(1 + \frac{D^2}{2} \right) \hat{v}_3 \tag{B.12}$$

B.2 $\int\int |\ell'|^4 \, \xi \, d\xi \, d\alpha$

Direct multiplication gives

$$|\boldsymbol{\ell}'|^4 = |\boldsymbol{\ell}|^4 = 1 + 2\xi^2 + \xi^4 \tag{B.13}$$

and the integral is straightforward:

$$\begin{aligned} \int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^4 \, \xi \, d\xi \, d\alpha &= 2\pi \int_0^D (\xi + 2\xi^3 + \xi^5) d\xi \\ &= \pi D^2 \left(1 + D^2 + \frac{D^4}{3}\right) \end{aligned} \tag{B.14}$$

B.3 $\int\int \ell' \ell'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha$

We start by writing $\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}$ as

$$\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}} = \mathbf{R}\boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}} \tag{B.15}$$

with

$$\boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}} = \begin{pmatrix} \xi^2 \cos^2\alpha & \xi^2 \cos\alpha \sin\alpha & \xi \cos\alpha \\ \xi^2 \cos\alpha \sin\alpha & \xi^2 \sin^2\alpha & \xi \sin\alpha \\ \xi \cos\alpha & \xi \sin\alpha & 1 \end{pmatrix} \tag{B.16}$$

Moving the terms $\mathbf{R}$ and $\mathbf{R}^{\mathrm{T}}$ outside the integral and integrating over α we obtain

$$\int_0^{2\pi} \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}} \, d\alpha = \begin{pmatrix} \pi\xi^2 & 0 & 0 \\ 0 & \pi\xi^2 & 0 \\ 0 & 0 & 2\pi \end{pmatrix} \tag{B.17}$$

The integral over ξ then gives

$$\int_0^D \int_0^{2\pi} \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}} \, \xi \, d\xi \, d\alpha = \begin{pmatrix} \pi D^4/4 & 0 & 0 \\ 0 & \pi D^4/4 & 0 \\ 0 & 0 & \pi D^2 \end{pmatrix} \tag{B.18}$$

$$= \pi D^2 \left(\frac{D^2}{4}\left(\mathbf{I} - \hat{z}\hat{z}^{\mathrm{T}}\right) + \hat{z}\hat{z}^{\mathrm{T}}\right) \tag{B.19}$$

Combining (B.19) and (B.11) we obtain finally

$$\begin{aligned} \int_0^D \int_0^{2\pi} \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha &= \mathbf{R} \left(\int_0^D \int_0^{2\pi} \boldsymbol{\ell} \boldsymbol{\ell}^{\mathrm{T}} \, \xi \, d\xi \, d\alpha \right) \mathbf{R}^{\mathrm{T}} \\ &= \pi D^2 \left(\frac{D^2}{4} \left(\mathbf{I} - \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} \right) + \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} \right) \end{aligned} \tag{B.20}$$

B.4 $\int\int |\ell'|^2 \ell' \ell'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha$

The only difference between this integral and the previous one is the presence of the term $|\boldsymbol{\ell}'|^2 = \boldsymbol{\ell}^{\mathrm{T}} \boldsymbol{\ell} = 1 + \xi^2$. Since this term does not depend on α we can proceed as before to find

$$\int_0^{2\pi} |\boldsymbol{\ell}'|^2 \boldsymbol{\ell} \boldsymbol{\ell}^{\mathrm{T}} \, d\alpha = \begin{pmatrix} \pi(\xi^2 + \xi^4) & 0 & 0 \\ 0 & \pi(\xi^2 + \xi^4) & 0 \\ 0 & 0 & 2\pi(1 + \xi^2) \end{pmatrix} \tag{B.21}$$

Integrating over ξ we then have

$$\begin{aligned} \int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^2 \boldsymbol{\ell} \boldsymbol{\ell}^{\mathrm{T}} \, \xi \, d\xi \, d\alpha &= \begin{pmatrix} \pi(\frac{D^4}{4} + \frac{D^6}{6}) & 0 & 0 \\ 0 & \pi(\frac{D^4}{4} + \frac{D^6}{6}) & 0 \\ 0 & 0 & \pi D^2 (1 + \frac{D^2}{2}) \end{pmatrix} \\ &= \pi D^2 \left(\left(\frac{D^2}{4} + \frac{D^4}{6} \right) \mathbf{I} + \left(1 + \frac{D^2}{4} - \frac{D^4}{6} \right) \hat{z} \hat{z}^{\mathrm{T}} \right) \end{aligned} \tag{B.22}$$

Including the rotation, we obtain

$$\begin{aligned} \int_0^D \int_0^{2\pi} |\boldsymbol{\ell}'|^2 \boldsymbol{\ell}' \boldsymbol{\ell}'^{\mathrm{T}} \, \xi \, d\xi \, d\alpha &= \mathbf{R} \left(\int_0^D \int_0^{2\pi} |\boldsymbol{\ell}|^2 \boldsymbol{\ell} \boldsymbol{\ell}^{\mathrm{T}} \, \xi \, d\xi \, d\alpha \right) \mathbf{R}^{\mathrm{T}} \\ &= \pi D^2 \left(\left(\frac{D^2}{4} + \frac{D^4}{6} \right) \mathbf{I} + \left(1 + \frac{D^2}{4} - \frac{D^4}{6} \right) \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} \right) \end{aligned} \tag{B.23}$$

B.5 $\int\int (\mathbf{u}\cdot\ell')\ell'\ell'^T\,\xi\,d\xi\,d\alpha$

This integral is somewhat more tedious than the previous ones due to the $(\mathbf{u}\cdot\boldsymbol{\ell}')$ term which does depend on α. We start by defining

$$\mathbf{u}' = \begin{pmatrix} u'_x \\ u'_y \\ u'_z \end{pmatrix} = \mathbf{R}^{\mathrm{T}}\mathbf{u} \tag{B.24}$$

and note that from the definition (B.11)

$$u'_x = \mathbf{u}\cdot\hat{v}_1,\ \ u'_y = \mathbf{u}\cdot\hat{v}_2,\ \text{and}\ u'_z = \mathbf{u}\cdot\hat{v}_3 \tag{B.25}$$

Since $\mathbf{u}\cdot\boldsymbol{\ell}' = \mathbf{u}'\cdot\boldsymbol{\ell}$ we can write

$$\mathbf{u}\cdot\boldsymbol{\ell}' = u'_x\,\xi\cos\alpha + u'_y\,\xi\sin\alpha + u'_z \tag{B.26}$$

This scalar term multiplies each element of the array $\mathbf{R}\boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}$ producing many products of trigonometric terms. Fortunately, only a few of these are non-zero after integrating over α from 0 to 2π. We have

$$\int_0^{2\pi}(\mathbf{u}\cdot\boldsymbol{\ell}')\boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\,d\alpha = \pi\begin{pmatrix} u'_z\xi^2 & 0 & u'_x\xi^2 \\ 0 & u'_z\xi^2 & u'_y\xi^2 \\ u'_x\xi^2 & u'_y\xi^2 & 2u'_z \end{pmatrix} \tag{B.27}$$

and performing the integration over ξ, we obtain

$$\int_0^D\int_0^{2\pi}(\mathbf{u}\cdot\boldsymbol{\ell}')\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\,\xi\,d\xi\,d\alpha = \pi D^2\mathbf{R}\begin{pmatrix} u'_zD^2/4 & 0 & u'_xD^2/4 \\ 0 & u'_zD^2/4 & u'_yD^2/4 \\ u'_xD^2/4 & u'_yD^2/4 & u'_z \end{pmatrix}\mathbf{R}^{\mathrm{T}}$$

$$\begin{aligned} &= \pi D^2\mathbf{R}\left(u'_z\frac{D^2}{4}\mathbf{I} + u'_z\left(1-\frac{3D^2}{4}\right)\hat{z}\hat{z}^{\mathrm{T}} + \frac{D^2}{4}\left(\mathbf{u}'\hat{z}^{\mathrm{T}} + \hat{z}\mathbf{u}'^{\mathrm{T}}\right)\right)\mathbf{R}^{\mathrm{T}} \\ &= \pi D^2\left((\mathbf{u}\cdot\hat{v}_3)\frac{D^2}{4}\mathbf{I} + (\mathbf{u}\cdot\hat{v}_3)\left(1-\frac{3D^2}{4}\right)\hat{v}_3\hat{v}_3{}^{\mathrm{T}} + \frac{D^2}{4}\left(\mathbf{u}\hat{v}_3{}^{\mathrm{T}} + \hat{v}_3\mathbf{u}^{\mathrm{T}}\right)\right) \end{aligned} \tag{B.28}$$

There is a special case which is worth noting. If $\mathbf{u} = \hat{v}_3$, equation (B.28) becomes

$$\begin{aligned}\int_0^D \int_0^{2\pi} (\hat{v}_3 \cdot \boldsymbol{\ell}')\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\, \xi \, d\xi \, d\alpha &= \pi D^2 \left(\frac{D^2}{4}\mathbf{I} + \left(1 - \frac{D^2}{4}\right) \hat{v}_3 \hat{v}_3{}^{\mathrm{T}} \right) \\ &= \int_0^D \int_0^{2\pi} \boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\, \xi \, d\xi \, d\alpha \qquad \text{(B.29)}\end{aligned}$$

Multiplying the integrand by $(\hat{v}_3 \cdot \boldsymbol{\ell}')$ thus has no effect on the result.

B.6 $\int\int (\mathbf{u}_1 \cdot \ell')(\mathbf{u}_2 \cdot \ell')\ell' \ell'^T \, \xi \, d\xi \, d\alpha$

In this final integral even more complex trigonometric products are encountered. Expanding the term $(\mathbf{u}_1 \cdot \boldsymbol{\ell}')(\mathbf{u}_2 \cdot \boldsymbol{\ell}')$ we have

$$\begin{aligned}(\mathbf{u}_1 \cdot \boldsymbol{\ell}')(\mathbf{u}_2 \cdot \boldsymbol{\ell}') &= \mathbf{u}'^{\mathrm{T}}_1 \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}} \mathbf{u}'_2 \\ &= u'_{1x}u'_{2x}\xi^2 \cos^2\alpha + u'_{1y}u'_{2y}\xi^2 \sin^2\alpha + u'_{1z}u'_{2z} \\ &\quad + (u'_{1x}u'_{2y} + u'_{2x}u'_{1y})\, \xi^2 \cos\alpha \sin\alpha + (u'_{1x}u'_{2z} + u'_{2x}u'_{1z})\, \xi \cos\alpha \\ &\quad + (u'_{1y}u'_{2z} + u'_{2y}u'_{1z})\, \xi \sin\alpha \qquad \text{(B.30)}\end{aligned}$$

where $\mathbf{u}'_1 = \mathbf{R}^{\mathrm{T}}\mathbf{u}_1$ and $\mathbf{u}'_2 = \mathbf{R}^{\mathrm{T}}\mathbf{u}_2$. The integral is written as

$$\int_0^D \int_0^{2\pi} (\mathbf{u}_1 \cdot \boldsymbol{\ell}')(\mathbf{u}_2 \cdot \boldsymbol{\ell}')\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\, \xi \, d\xi \, d\alpha = \mathbf{R} \left(\int_0^D \int_0^{2\pi} (\mathbf{u}'^{\mathrm{T}}_1 \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}} \mathbf{u}'_2)\boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\, \xi \, d\xi \, d\alpha \right) \mathbf{R}^{\mathrm{T}} \qquad \text{(B.31)}$$

so that each element of the matrix $\boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}$ is multiplied by the scalar $\mathbf{u}'^{\mathrm{T}}_1 \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}} \mathbf{u}'_2$. The only trigonometric products which survive the integration over α, however, are the following:

$$\int_0^{2\pi} \cos^4\alpha \, d\alpha = \frac{3\pi}{4} \qquad \text{(B.32)}$$

$$\int_0^{2\pi} \sin^4\alpha \, d\alpha = \frac{3\pi}{4} \qquad \text{(B.33)}$$

$$\int_0^{2\pi} \cos^2\alpha \sin^2\alpha \, d\alpha = \frac{\pi}{4} \qquad \text{(B.34)}$$

$$\int_0^{2\pi} \cos^2\alpha \, d\alpha = \pi \qquad \text{(B.35)}$$

$$\int_0^{2\pi} \sin^2\alpha \, d\alpha = \pi \qquad \text{(B.36)}$$

$$\int_0^{2\pi} d\alpha = 2\pi \tag{B.37}$$

We now define the following integrals:

$$\begin{aligned} \mathbf{L}_1 &\equiv \int_0^D \int_0^{2\pi} \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\, \xi^2 \cos^2\alpha\; \xi\, d\xi\, d\alpha \\ &= \frac{\pi D^4}{4} \begin{pmatrix} D^2/2 & 0 & 0 \\ 0 & D^2/6 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{B.38}$$

$$\begin{aligned} \mathbf{L}_2 &\equiv \int_0^D \int_0^{2\pi} \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\, \xi^2 \sin^2\alpha\; \xi\, d\xi\, d\alpha \\ &= \frac{\pi D^4}{4} \begin{pmatrix} D^2/6 & 0 & 0 \\ 0 & D^2/2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{B.39}$$

$$\begin{aligned} \mathbf{L}_3 &\equiv \int_0^D \int_0^{2\pi} \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\, \xi\, d\xi\, d\alpha \\ &= \pi D^2 \begin{pmatrix} D^2/4 & 0 & 0 \\ 0 & D^2/4 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{B.40}$$

$$\begin{aligned} \mathbf{L}_4 &\equiv \int_0^D \int_0^{2\pi} \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\, \xi^2 \cos\alpha \sin\alpha\; \xi\, d\xi\, d\alpha \\ &= \frac{\pi D^6}{24} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{B.41}$$

$$\begin{aligned} \mathbf{L}_5 &\equiv \int_0^D \int_0^{2\pi} \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\, \xi \cos\alpha\; \xi\, d\xi\, d\alpha \\ &= \frac{\pi D^4}{4} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{aligned} \tag{B.42}$$

$$\mathbf{L}_6 \equiv \int_0^D \int_0^{2\pi} \boldsymbol{\ell}\boldsymbol{\ell}^{\mathrm{T}}\, \xi \sin\alpha\; \xi\, d\xi\, d\alpha$$

$$= \frac{\pi D^4}{4}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{B.43}$$

so that

$$\int_0^D \int_0^{2\pi} (\mathbf{u}_1 \cdot \boldsymbol{\ell}')(\mathbf{u}_2 \cdot \boldsymbol{\ell}')\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\, \xi \, d\xi \, d\alpha = \mathbf{R}\begin{pmatrix} \mathbf{u'}_1^{\mathrm{T}}\mathbf{L}_1\mathbf{u'}_2 & \mathbf{u'}_1^{\mathrm{T}}\mathbf{L}_4\mathbf{u'}_2 & \mathbf{u'}_1^{\mathrm{T}}\mathbf{L}_5\mathbf{u'}_2 \\ \mathbf{u'}_1^{\mathrm{T}}\mathbf{L}_4\mathbf{u'}_2 & \mathbf{u'}_1^{\mathrm{T}}\mathbf{L}_2\mathbf{u'}_2 & \mathbf{u'}_1^{\mathrm{T}}\mathbf{L}_6\mathbf{u'}_2 \\ \mathbf{u'}_1^{\mathrm{T}}\mathbf{L}_5\mathbf{u'}_2 & \mathbf{u'}_1^{\mathrm{T}}\mathbf{L}_6\mathbf{u'}_2 & \mathbf{u'}_1^{\mathrm{T}}\mathbf{L}_3\mathbf{u'}_2 \end{pmatrix}\mathbf{R}^{\mathrm{T}} \tag{B.44}$$

Using the definitions (B.38)–(B.43), we expand the elements of this matrix, grouping terms by powers of D, to arrive at

$$\begin{aligned} &\int_0^D \int_0^{2\pi} (\mathbf{u}_1 \cdot \boldsymbol{\ell}')(\mathbf{u}_2 \cdot \boldsymbol{\ell}')\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\, \xi \, d\xi \, d\alpha = \\ &\quad \frac{\pi D^6}{8}\mathbf{R}\begin{pmatrix} u'_{1x}u'_{2x} + u'_{1y}u'_{2y}/3 & (u'_{1x}u'_{2y} + u'_{2x}u'_{1y})/3 & 0 \\ (u'_{1x}u'_{2y} + u'_{2x}u'_{1y})/3 & u'_{1x}u'_{2x}/3 + u'_{1y}u'_{2y} & 0 \\ 0 & 0 & 0 \end{pmatrix}\mathbf{R}^{\mathrm{T}} \\ &\quad + \frac{\pi D^4}{4}\mathbf{R}\begin{pmatrix} u'_{1z}u'_{2z} & 0 & u'_{1x}u'_{2z} + u'_{2x}u'_{1z} \\ 0 & u'_{1z}u'_{2z} & u'_{1y}u'_{2z} + u'_{2y}u'_{1z} \\ u'_{1x}u'_{2z} + u'_{2x}u'_{1z} & u'_{1y}u'_{2z} + u'_{2y}u'_{1z} & \mathbf{u}'_1 \cdot \mathbf{u}'_2 \end{pmatrix}\mathbf{R}^{\mathrm{T}} \\ &\quad + \pi D^2\left(1 - \frac{D^2}{4}\right)(\mathbf{u}_1 \cdot \hat{v}_3)(\mathbf{u}_2 \cdot \hat{v}_3)\, \hat{v}_3\hat{v}_3{}^{\mathrm{T}} \end{aligned} \tag{B.45}$$

We can write the first of these matrices as

$$\begin{aligned} &\frac{\pi D^6}{8}\mathbf{R}\begin{pmatrix} u'_{1x}u'_{2x} + u'_{1y}u'_{2y}/3 & (u'_{1x}u'_{2y} + u'_{2x}u'_{1y})/3 & 0 \\ (u'_{1x}u'_{2y} + u'_{2x}u'_{1y})/3 & u'_{1x}u'_{2x}/3 + u'_{1y}u'_{2y} & 0 \\ 0 & 0 & 0 \end{pmatrix}\mathbf{R}^{\mathrm{T}} = \\ &\quad \frac{\pi D^6}{24}\mathbf{R}\left(\mathbf{I} - \hat{z}\hat{z}^{\mathrm{T}}\right)\left[\mathbf{u}'_1\mathbf{u}'^{\mathrm{T}}_2 + \mathbf{u}'_2\mathbf{u}'^{\mathrm{T}}_1 + ((\mathbf{u}'_1 \cdot \mathbf{u}'_2) - u'_{1z}u'_{2z})\mathbf{I}\right]\left(\mathbf{I} - \hat{z}\hat{z}^{\mathrm{T}}\right)\mathbf{R}^{\mathrm{T}} \\ &= \frac{\pi D^6}{24}\left(\mathbf{I} - \hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)\left[\mathbf{u}_1\mathbf{u}_2^{\mathrm{T}} + \mathbf{u}_2\mathbf{u}_1^{\mathrm{T}} + ((\mathbf{u}_1 \cdot \mathbf{u}_2) - (\mathbf{u}_1 \cdot \hat{v}_3)(\mathbf{u}_1 \cdot \hat{v}_3))\,\mathbf{I}\right]\left(\mathbf{I} - \hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right) \end{aligned} \tag{B.46}$$

To simplify notation, we define the vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ as

$$\mathbf{w}_1 \equiv (\hat{v}_3 \times \mathbf{u}_1) \times \hat{v}_3 = \left(\mathbf{I} - \hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)\mathbf{u}_1, \text{ and, } \mathbf{w}_2 \equiv (\hat{v}_3 \times \mathbf{u}_2) \times \hat{v}_3 = \left(\mathbf{I} - \hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)\mathbf{u}_2 \quad \text{(B.47)}$$

Since $\hat{v}_3 \cdot \hat{v}_3 = 1$, we also note that

$$\mathbf{w}_1 \cdot \mathbf{w}_2 = \mathbf{u}_1 \cdot \mathbf{u}_2 - (\mathbf{u}_1 \cdot \hat{v}_3)(\mathbf{u}_2 \cdot \hat{v}_3) \quad \text{(B.48)}$$

Equation (B.46) can thus be reduced to

$$\frac{\pi D^6}{8}\mathbf{R}\begin{pmatrix} u'_{1x}u'_{2x} + u'_{1y}u'_{2y}/3 & (u'_{1x}u'_{2y} + u'_{2x}u'_{1y})/3 & 0 \\ (u'_{1x}u'_{2y} + u'_{2x}u'_{1y})/3 & u'_{1x}u'_{2x}/3 + u'_{1y}u'_{2y} & 0 \\ 0 & 0 & 0 \end{pmatrix}\mathbf{R}^{\mathrm{T}} =$$
$$\frac{\pi D^6}{24}\left[\mathbf{w}_1\mathbf{w}_2^{\mathrm{T}} + \mathbf{w}_2\mathbf{w}_1^{\mathrm{T}} + (\mathbf{w}_1 \cdot \mathbf{w}_2)\left(\mathbf{I} - \hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)\right] \quad \text{(B.49)}$$

The second matrix of equation (B.45) can be written as

$$\frac{\pi D^4}{4}\mathbf{R}\begin{pmatrix} u'_{1z}u'_{2z} & 0 & u'_{1x}u'_{2z} + u'_{2x}u'_{1z} \\ 0 & u'_{1z}u'_{2z} & u'_{1y}u'_{2z} + u'_{2y}u'_{1z} \\ u'_{1x}u'_{2z} + u'_{2x}u'_{1z} & u'_{1y}u'_{2z} + u'_{2y}u'_{1z} & \mathbf{u'}_1 \cdot \mathbf{u'}_2 \end{pmatrix}\mathbf{R}^{\mathrm{T}} =$$
$$\frac{\pi D^4}{4}\mathbf{R}\left[u'_{2z}\left(\mathbf{u'}_1\hat{z}^{\mathrm{T}} + \hat{z}\mathbf{u'}^{\mathrm{T}}_1\right) + u'_{1z}\left(\mathbf{u'}_2\hat{z}^{\mathrm{T}} + \hat{z}\mathbf{u'}^{\mathrm{T}}_2\right)\right.$$
$$\left. - 4u'_{1z}u'_{2z}\hat{z}\hat{z}^{\mathrm{T}} + u'_{1z}u'_{2z}\left(\mathbf{I} - \hat{z}\hat{z}^{\mathrm{T}}\right) + (\mathbf{u'}_1 \cdot \mathbf{u'}_2)\hat{z}\hat{z}^{\mathrm{T}}\right]\mathbf{R}^{\mathrm{T}}$$

$$= \frac{\pi D^4}{4}\left[(\mathbf{u}_2 \cdot \hat{v}_3)\left(\mathbf{u}_1\hat{v}_3{}^{\mathrm{T}} + \hat{v}_3\mathbf{u}_1^{\mathrm{T}}\right) + (\mathbf{u}_1 \cdot \hat{v}_3)\left(\mathbf{u}_2\hat{v}_3{}^{\mathrm{T}} + \hat{v}_3\mathbf{u}_2^{\mathrm{T}}\right)\right.$$
$$- 4(\mathbf{u}_1 \cdot \hat{v}_3)(\mathbf{u}_2 \cdot \hat{v}_3)\hat{v}_3\hat{v}_3{}^{\mathrm{T}} + (\mathbf{u}_1 \cdot \hat{v}_3)(\mathbf{u}_2 \cdot \hat{v}_3)\left(\mathbf{I} - \hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)$$
$$\left. + (\mathbf{u}_1 \cdot \mathbf{u}_2)\hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right] \quad \text{(B.50)}$$

From (B.47) we can derive the following expression for $\mathbf{w}_1\mathbf{w}_2^{\mathrm{T}} + \mathbf{w}_2\mathbf{w}_1^{\mathrm{T}}$ as

$$\mathbf{w}_1\mathbf{w}_2^{\mathrm{T}} + \mathbf{w}_2\mathbf{w}_1^{\mathrm{T}} = \mathbf{u}_1\mathbf{u}_2^{\mathrm{T}} + \mathbf{u}_1\mathbf{u}_2^{\mathrm{T}} - (\mathbf{u}_2 \cdot \hat{v}_3)\left(\mathbf{u}_1\hat{v}_3{}^{\mathrm{T}} + \hat{v}_3\mathbf{u}_1^{\mathrm{T}}\right)$$
$$- (\mathbf{u}_1 \cdot \hat{v}_3)\left(\mathbf{u}_2\hat{v}_3{}^{\mathrm{T}} + \hat{v}_3\mathbf{u}_2^{\mathrm{T}}\right) + 2(\mathbf{u}_1 \cdot \hat{v}_3)(\mathbf{u}_2 \cdot \hat{v}_3)\hat{v}_3\hat{v}_3{}^{\mathrm{T}} \quad \text{(B.51)}$$

Then, using (B.48) and (B.51), we can rewrite equation (B.50) as:

$$\frac{\pi D^4}{4}\mathbf{R}\begin{pmatrix} u'_{1z}u'_{2z} & 0 & u'_{1x}u'_{2z}+u'_{2x}u'_{1z} \\ 0 & u'_{1z}u'_{2z} & u'_{1y}u'_{2z}+u'_{2y}u'_{1z} \\ u'_{1x}u'_{2z}+u'_{2x}u'_{1z} & u'_{1y}u'_{2z}+u'_{2y}u'_{1z} & \mathbf{u'}_1\cdot\mathbf{u'}_2 \end{pmatrix}\mathbf{R}^{\mathrm{T}} =$$

$$\frac{\pi D^4}{4}\Big[\mathbf{u}_1\mathbf{u}_2^{\mathrm{T}}+\mathbf{u}_2\mathbf{u}_1^{\mathrm{T}}-\mathbf{w}_1\mathbf{w}_2^{\mathrm{T}}-\mathbf{w}_2\mathbf{w}_1^{\mathrm{T}}-2(\mathbf{u}_1\cdot\hat{v}_3)(\mathbf{u}_2\cdot\hat{v}_3)\hat{v}_3\hat{v}_3{}^{\mathrm{T}} + (\mathbf{u}_1\cdot\mathbf{u}_2-\mathbf{w}_1\cdot\mathbf{w}_2)\left(\mathbf{I}-\hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)+(\mathbf{u}_1\cdot\mathbf{u}_2)\hat{v}_3\hat{v}_3{}^{\mathrm{T}}\Big]$$

$$= \frac{\pi D^4}{4}\Big[\mathbf{u}_1\mathbf{u}_2^{\mathrm{T}}+\mathbf{u}_2\mathbf{u}_1^{\mathrm{T}}+(\mathbf{u}_1\cdot\mathbf{u}_2)\,\mathbf{I}-2(\mathbf{u}_1\cdot\hat{v}_3)(\mathbf{u}_2\cdot\hat{v}_3)\hat{v}_3\hat{v}_3{}^{\mathrm{T}} - \left(\mathbf{w}_1\mathbf{w}_2^{\mathrm{T}}+\mathbf{w}_2\mathbf{w}_1^{\mathrm{T}}+(\mathbf{w}_1\cdot\mathbf{w}_2)\left(\mathbf{I}-\hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)\right)\Big] \tag{B.52}$$

Finally, we combine equations (B.49) and (B.52) with the third term of equation (B.45) to obtain the result:

$$\int_0^D\int_0^{2\pi}(\mathbf{u}_1\cdot\boldsymbol{\ell}')(\mathbf{u}_2\cdot\boldsymbol{\ell}')\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\,\xi\,d\xi\,d\alpha =$$

$$\frac{\pi D^4}{4}\left(\frac{D^2}{6}-1\right)\Big[\mathbf{w}_1\mathbf{w}_2^{\mathrm{T}}+\mathbf{w}_2\mathbf{w}_1^{\mathrm{T}}+(\mathbf{w}_1\cdot\mathbf{w}_2)\left(\mathbf{I}-\hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right)\Big] + \frac{\pi D^4}{4}\Big[\mathbf{u}_1\mathbf{u}_2^{\mathrm{T}}+\mathbf{u}_2\mathbf{u}_1^{\mathrm{T}}+(\mathbf{u}_1\cdot\mathbf{u}_2)\,\mathbf{I}\Big] + \pi D^2\left(1-\frac{3D^2}{4}\right)(\mathbf{u}_1\cdot\hat{v}_3)(\mathbf{u}_2\cdot\hat{v}_3)\,\hat{v}_3\hat{v}_3{}^{\mathrm{T}} \tag{B.53}$$

We note that a special case occurs if either $\mathbf{u}_1$ or $\mathbf{u}_2$ is equal to $\hat{v}_3$. The solution then becomes

$$\int_0^D\int_0^{2\pi}(\hat{v}_3\cdot\boldsymbol{\ell}')(\mathbf{u}\cdot\boldsymbol{\ell}')\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\,\xi\,d\xi\,d\alpha =$$

$$\pi D^2\left[\frac{D^2}{4}\left(\mathbf{u}\hat{v}_3{}^{\mathrm{T}}+\hat{v}_3\mathbf{u}^{\mathrm{T}}\right)+(\mathbf{u}\cdot\hat{v}_3)\frac{D^2}{4}\mathbf{I}+\left(1-\frac{3D^2}{4}\right)(\mathbf{u}\cdot\hat{v}_3)\,\hat{v}_3\hat{v}_3{}^{\mathrm{T}}\right]$$

$$= \int_0^D\int_0^{2\pi}(\mathbf{u}\cdot\boldsymbol{\ell}')\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathrm{T}}\,\xi\,d\xi\,d\alpha \tag{B.54}$$

Again, multiplying the integrand by $(\hat{v}_3\cdot\boldsymbol{\ell}')$ has no effect on the result.