ESTIMATION PROBLEMS IN DIGITAL VIDEO

Richard J. Radke

A DISSERTATION

PRESENTED TO THE FACULTY

OF PRINCETON UNIVERSITY

IN CANDIDACY FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE

BY THE DEPARTMENT OF

ELECTRICAL ENGINEERING

June 2001

Abstract

The swift evolution of technology over the past twenty years has brought digital video from the province of graduate school computer labs to the home of the average consumer. Furthermore, the wireless revolution will soon enable the display of realistic, interactive digital video on a new generation of sophisticated portable devices. Advances in video processing research will be required to meet the demands of new applications and exploit the characteristics of new communications systems. In this thesis, we pose, analyze, and solve several estimation problems in digital video that arise from images taken of the same scene by different cameras, and introduce some novel applications.

We first study the problem of estimating a projective transformation from noisy measurements. Though this common problem in computer vision is typically solved as an eight-parameter non-quadratic minimization, we present analytical and experimental evidence that the same solution can be more efficiently obtained by minimizing a related functional of two parameters.

We next turn to the estimation of correspondence in the general case. We fully characterize the class of point correspondences which can arise from a physical imaging system, and use this understanding to design an algorithm for estimating correspondence that is particularly suitable for wide-baseline camera configurations, where the assumptions of most existing correspondence algorithms are unsatisfied.

We demonstrate that dense correspondence between two images of the same scene is sufficient to synthesize realistic virtual images of the scene from new viewpoints. Prior work on still images is extended to the synthesis of compelling virtual video from two video sequences, by means of a novel and efficient algorithm for the recursive propagation of correspondence estimates between video frames.

Virtual video has applications not only in visualization and entertainment, but also in wireless multimedia applications. We show how virtual images can be used to interpolate low frame-rate video with minimal overhead and low computational complexity, by solving an estimation problem to construct virtual images which accurately represent real video frames.

Acknowledgements

My time at Princeton has been quite enjoyable, largely due to my excellent and supportive advisors, Peter Ramadge and Sanjeev Kulkarni. Peter and Sanj have been advisors in every sense of the word, and under their guidance I feel I've become a better researcher, writer, and teacher. Peter meticulously read innumerable drafts of papers over the past five years. After several pens' worth of red marks I now know all too well that the house on the corner is white. I am also grateful for the responsibility he granted me in running the Multimedia Lab and presenting our research at meetings. Working with Sanj on ELE 201 for the past two years has been a great learning experience from an excellent teacher, and he can be excused for his prediction that I could write this thesis "in about a week, if you really put your mind to it." Ha ha.

I have also benefited from excellent advice, academic and otherwise, given to me over the years by Bede Liu, Vince Poor, and Stuart Schwartz. Thanks to Adam Finkelstein for being the third reader of this thesis and providing insight from a computer graphics perspective.

This work was supported by generous grants from the IBM Tokyo Research Lab. Special thanks to Tomio Echigo, Kazuo Iwano, Yoichi Takao, and Jung-kook Hong for making our collaboration possible. Soccer images in this thesis and in the associated publications are printed with their kind permission. I was also supported at Princeton by two generous fellowships: a Wu Fellowship sponsored by Princeton University and Sir Gordon Wu, and a National Defense Science and Engineering Graduate Fellowship sponsored by the U.S. Department of Defense. Thanks also for funding from the New Jersey Center for Multimedia Research.

Several pieces of my computer code started out as revisions to software written by Yap-Peng Tan. Princeton Summer Interns Kevin Gold, Jung Kim, and Tim Cobb also wrote some useful video processing routines in the summer of 2000 that shaped my thinking on the last chapter. Ingrid Daubechies supplied the dragon wall pictures in Figure 4.2. Thanks to Scott Rickard for collaborating on the audio interpolation idea. Special thanks to Aaron Hertzmann for tossing hundreds of interesting papers my way and introducing me to the Siggraph proceedings. Finally, thanks to Steph Costantini, Tom Roddenberry, Karen Williams, and Jay Plett for administrative and computer

support.

Many friends, old and new, have supported me throughout my years at Princeton. Grad school might have been rough indeed without the following folks: Ted Bergman, Gilbert Collins, Julio Concha, Aaron Hertzmann, Yoshiko Jo, Rob Joyce, Toshi Kida, Shalinee Kishore, Phoebe Lam, Carl Nuzman, Shira Sand, Kama Sandilya, John Smee, Katerina Varsou, Catherine Witt, Audrey Wright, Min Wu, and Vitali Zagorodnov.

The dedication page is from *Snake 'N' Bacon's Cartoon Cabaret* by Michael Kupperman.


DEDICATED TO MOM AND DAD

"Ah, there's nothing more exciting than science. You get all the fun of sitting still, being quiet, writing down numbers, paying attention . . . science has it all."

– The Simpsons

Contents

Abstract **iii**

Acknowledgements **iv**

1 Introduction **1**

2 Preliminaries **8**

2.1 Cameras and Images 8

2.2 Homogeneous Coordinates 10

2.3 Two Cameras 11

2.4 The Fundamental Matrix 13

2.5 Epipolar Geometry 15

2.6 Affine Transformations 16

2.7 References 18

3 Basic Estimation Problems **19**

3.1 Estimating Affine Transformations 20

3.1.1 The Linear Least-Squares Problem 20

3.1.2 Estimating Translation 22

3.1.3 Estimating Rotation and Scale 24

3.2 Estimating the Fundamental Matrix 25

3.3 Estimating Rectifying Projective Transformations 26

3.4 Estimating Point Correspondences 31

3.5 Review of 2-D Correspondence Algorithms 33
3.5.1 Optical Flow 33
3.5.2 Layered Motion 35
3.5.3 Structure from Motion 37
3.5.4 Adaptive Meshes 37
3.5.5 Beier-Neely Morphing 38
3.6 References 38

4 Projective Transformations **42**
4.1 Origins of Projective Transformations 43
4.2 The Least Squares Estimate 48
4.3 Data Normalization 52
4.4 The Behavior of J on Singular Lines 53
4.5 Line-Search Descent 56
4.6 Second-Derivative Methods 58
4.7 Experimental Results 61
4.8 Conclusions 72
4.9 References 73

5 Correspondence **75**
5.1 Review of Epipolar Correspondence Algorithms 77
5.1.1 Basic Dynamic Programming: Ohta and Kanade 78
5.1.2 Bayesian Approach: Belheumer 79
5.1.3 Maximum Likelihood Approach: Cox et al. 80
5.1.4 Maximum-Flow Graph: Ishikawa and Geiger 80
5.1.5 Curve Matching: Tomasi and Manduchi 80
5.2 Non-Monotonicity 81
5.3 The Correspondence Graph 82
5.3.1 Constraints on Correspondence 82
5.3.2 The Correspondence Graph 86

5.3.3 Examples of Correspondence Graphs . . . 88
5.4 Estimating Correspondence Graphs . . . 91
5.4.1 Step 1: Segmentation and Correspondence of Foreground Objects . . . 92
5.4.2 Step 2: Estimating Initial Global Correspondence for Background Pixels . 94
5.4.3 Step 3: Generating the Basic Correspondence Graph . . . 96
5.4.4 Step 4: Refining each Monotonic Piece . . . 96
5.5 Experimental Results . . . 97
5.6 Conclusions . . . 104
5.7 References . . . 106

6 Virtual Video **109**
6.1 Review of View Morphing . . . 111
6.1.1 View Interpolation . . . 111
6.1.2 View Morphing . . . 114
6.2 Experimental Results: Virtual Images from Wide-Baseline Stills . . . 115
6.3 Virtual Video . . . 119
6.3.1 Notation . . . 119
6.3.2 Recursive Propagation . . . 121
6.3.3 Time Update . . . 122
6.3.4 Measurement Update . . . 123
6.3.5 Error Analysis . . . 125
6.4 Experimental Results: Virtual Video from Wide-Baseline Video . . . 129
6.5 Conclusions . . . 141
6.6 References . . . 142

7 View Morphing for Time-Domain Interpolation of Low-Bit-Rate Video **144**
7.1 Review of Video Compression Algorithms . . . 146
7.2 View Morphing as a Predictive Mechanism . . . 150
7.3 Correspondence Between Anchor Frames . . . 152
7.4 Aligning the Virtual and Actual Frames . . . 154

7.5 Experimental Results 155
7.6 Conclusions 163
7.7 References 165

8 Conclusions **167**

A Proof of Theorem 4.2 **171**

B Newton Methods **175**

C The Hessian of $J(c)$ **180**
C.1 The Hessian Itself 180
C.2 The Gauss-Newton Approximation 184

D Audio Interpolation **188**

List of Tables

4.1 Information and nominal parameters for the 6 data sets. 63
8.1 Thesis contributions. 169

List of Figures

2.1 Perspective projection. 9
2.2 Rigid motion of a camera. 12
2.3 Epipolar geometry. 15
2.4 Translation, rotation, scaling, and shear. 17
3.1 The effects of outliers. 21
3.2 Rectifying projective transformations. 27
3.3 Intersection of images with a plane. 29
3.4 Regions that are difficult to match correctly. 31
4.1 Perspective effects. 44
4.2 Images from a non-translating camera. 46
4.3 Images of a planar scene. 46
4.4 Singular lines. 47
4.5 The admissible region of the plane bounded by singular lines. 48
4.6 Two views of the cost function $J(c_1, c_2)$ for data points from actual images. 51
4.7 Proposed algorithm for minimizing $J(c)$. 61
4.8 Floating point operation counts for the Firestone 1-2 data set, purely Gaussian noise. 65
4.9 Floating point operation counts for the Firestone 2-1 data set, purely Gaussian noise. 65
4.10 Floating point operation counts for the B320 0-1 data set, purely Gaussian noise. . 66
4.11 Floating point operation counts for the Track data set, purely Gaussian noise. . . . 66
4.12 Floating point operation counts for the Atrium 1-2 data set, purely Gaussian noise. 67
4.13 Floating point operation counts for the Atrium 2-3 data set, purely Gaussian noise. 67
4.14 $\|\hat{H} - H\|/\|H\|$ as a function of noise variance for $\hat{N}$ and GNJ methods. 68

4.15 Floating point operation counts for the Firestone 1-2 data set, Gaussian noise with outliers. 68
4.16 Floating point operation counts for the Firestone 2-1 data set, Gaussian noise with outliers. 69
4.17 Floating point operation counts for the B320 0-1 data set, Gaussian noise with outliers. 69
4.18 Floating point operation counts for the Track data set, Gaussian noise with outliers. 70
4.19 Floating point operation counts for the Atrium 1-2 data set, Gaussian noise with outliers. 70
4.20 Floating point operation counts for the Atrium 2-3 data set, Gaussian noise with outliers. 71
5.1 Matching graph for conjugate epipolar lines. 77
5.2 Ohta and Kanade interval-matching cost function. 78
5.3 The “double nail illusion”. 81
5.4 Violations of monotonicity. 82
5.5 Epipolar geometry. 83
5.6 Mapping from (x, y)-space to (i, j)-space. 84
5.7 The set of points in front of both cameras. 85
5.8 The Southeasting operation. 87
5.9 Correspondence graph for Example 1 (simple occlusion). 89
5.10 Correspondence graph for Example 2 (double-nail illusion). 90
5.11 An N-piece and 2-piece input can both result in a 3-piece Southeast output. 91
5.12 Matching a “two-legged” object with a “one-legged” object. 93
5.13 Using the planar surface to estimate initial correspondence. 95
5.14 Original image pair $(\mathcal{I}_0, \mathcal{I}_1)$. 97
5.15 Image pair, with segmentation. 97
5.16 Image pair, with point correspondences used for estimation. 98
5.17 Image pair, with sample epipolar lines. 99
5.18 Rectified image pair, with sample epipolar lines. 100
5.19 Registration of planar surface in soccer images. 100

5.20 Correspondence graph (basic topology), line 71. 101
5.21 Correspondence graph (basic topology), line 105. 102
5.22 Correspondence graph (basic topology), line 120. 102
5.23 Correspondence graph (refined), line 71. 103
5.24 Correspondence graph (refined), line 105. 103
5.25 Correspondence graph (refined), line 120. 104
5.26 An unsatisfactory pair of rectified images. 105
6.1 View interpolation. 112
6.2 View morphing. 114
6.3 Original image pair $(\mathcal{I}_0, \mathcal{I}_1)$. 115
6.4 Synthesized virtual image $\hat{\mathcal{I}}_s$ at $s = 0.5$, with and without filling of occluded regions. 116
6.5 Interpolated virtual images $\hat{\mathcal{I}}_s$ at $s = 0.25$ and $s = 0.75$. 118
6.6 Extrapolated virtual images $\hat{\mathcal{I}}_s$ at $s = -0.5$ and $s = 1.5$. 118
6.7 Relationships between image planes. 120
6.8 The set $\mathcal{D}^i$ of rectangular domains searched by the correspondence operator C^i given basic correspondence graph topology for one epipolar line pair. 124
6.9 Measurement update by searching a local neighborhood $\mathcal{B}^i$ around the time-updated estimate. 125
6.10 Proof that $a \subset M^i(a)^{(\varepsilon)}$. 127
6.11 Filling in occluded regions in the time update. 130
6.12 Correspondence graphs, line 71, frames 415 and 417. 132
6.13 Correspondence graphs, line 105, frames 415 and 417. 133
6.14 Correspondence graphs, line 120, frames 415 and 417. 134
6.15 Virtual images, frame 415. 135
6.16 Virtual images, frame 433. 136
6.17 Virtual images, frame 447. 137
6.18 Virtual images, frame 465. 138
6.19 Virtual images, frame 487. 139
6.20 Virtual images, frame 499. 140

7.1 Schematic of block-based motion-compensation video compression algorithms. . . 147
7.2 Two frames of test video and their absolute luminance difference. 148
7.3 View morphing for interpolating video from a translating camera. 150
7.4 Correspondence for a scene composed of planar facets. 153
7.5 The two anchor frames. 156
7.6 Anchor frames, with feature points and control lines used to initialize correspondence. 156
7.7 Anchor frames, with feature points used to specify the left and right walls. 157
7.8 Estimates of the camera location s_t. 158
7.9 Original, interpolated, and error frames 30, 60, 90, 120, and 150 of the B320 sequence. 160
7.10 Peak signal to noise ratio over the test video sequence. 161
7.11 Detail of original, interpolated, and MPEG-compressed versions of frame 90. . . . 162
D.1 Microphone configuration. 189
D.2 Loci of sources that can be obtained with two microphones. 191

Chapter 1

Introduction

Twenty years ago, the storage and manipulation of digital video required unwieldy frame-buffering hardware and processing power not found outside of an advanced computer lab. Ten years later, a consumer with a high-end personal computer and CD-ROM drive might see short clips of digital video interspersed with video games. Today, the average college student interacts with digital video daily in the form of streaming news and entertainment broadcasts on the internet and movies in DVD format. The next ten years will bring video over wireless networks to laptop computers, personal digital assistants, cell phones, and even household appliances.

Many trends in technology and society drive the applications of digital video we will see in the coming years. Wireless devices are getting smaller, cheaper, and more popular among a wider segment of the public. On the other end of the spectrum, desktop computers have more processing power, bigger hard drives, and faster internet connections than ever before. Movies are slowly shifting to an all-digital pipeline, from shooting to post-production to projection to DVD release, enabling a new level of compelling digital special effects. At the convergence of these trends is a new breed of consumer who will make use of digital video at the workplace, on the commute, and in the home, and expects that this video will be realistic and interactive.

No matter how fast processors become, engineers will always be challenged with meeting the expectations of ever-more-demanding users in the face of limited time, restricted power, or low bandwidth constraints. To this end, we must continue to push the limits of our understanding of the relationships between cameras, images, and video, and exploit these relationships to store and manipulate digital video in efficient and flexible ways.

This thesis addresses some of the fundamental relationships between images taken of the same scene by multiple cameras. On one hand, we will show how new images and video of the scene can be robustly synthesized from the perspective of a camera not present in the original environment. On the other hand, we will show how the same techniques lend themselves to the elimination of redundant information in video so that it can be efficiently represented and transmitted. In a sense, all the problems we address rely on estimating a dense correspondence between images. That is, we want to estimate the image coordinates of every scene point that is visible in a pair of images. Of course, this is a classical problem in computer vision and has been the topic of intense research for over thirty years. However, much of the work in this area has been for stereoscopic or closely-spaced images that do not look "too different". Many applications of digital video in the future will require algorithms to process images from perspectives that differ substantially.

Much of this thesis is related in spirit to recent techniques in image-based rendering from the computer graphics community [1]-[12]. In contrast to the once prevalent school of thought that new images of a scene should be created by projecting a model of the scene in 3-D space onto a 2-D image plane, the central idea is that in certain circumstances, a new image of a scene can be created by processing a set of images of the scene, always staying in the 2-D domain. Hence, in image-based rendering, one of the most important quantities is image correspondence. Correspondence from multicamera video also has many useful applications in video coding, image understanding, and pattern recognition. While in some applications, an unstructured optical flow field may be an adequate representation of correspondence between an image pair, there are many practical situations in which a parametrized or structured correspondence is induced by the geometry of the cameras. In addition to coupling the correspondence to a physical modeling assumption, parametrized correspondence is generally more consistent, easier to manipulate, and contains more information about the relationships between images. Each chapter of this thesis poses and solves a parametric correspondence estimation problem for a different camera/scene configuration.

Chapter 2 introduces the notation and terminology that will be used throughout the text, as well as basic background material. Chapter 3 reviews standard methods for solving various estimation problems in digital video that we will use as building blocks in our algorithms.

We begin the main contributions of the thesis in Chapter 4 by discussing projective transformations. Projective transformations relate the coordinates of images that are taken by either a camera that undergoes only rotation while imaging an arbitrary scene, or one that rotates and translates while imaging a planar surface. Estimating the eight parameters of a projective transformation between a pair of image planes induces a global, dense correspondence between them. The estimated correspondence can be used to synthesize new views of the scene from different perspectives, e.g. by "rotating" the camera that took one of the original images, or by mosaicking many images into a large panorama. These applications have been well-studied, and what we address here are efficient algorithms for the initial step of estimating a projective transformation from noisy point correspondences. The contributions of the thesis in this regard include:

- **Reduction of dimensionality.** The projective transformation estimation problem is typically posed as the minimization of a nonlinear functional of eight parameters, and solved with an "off-the-shelf" numerical algorithm. We show that in fact, this minimization can be analytically reduced to a nonlinear problem in only two parameters.

- **Efficient algorithms.** We show that any descent algorithm to solve the eight-dimensional minimization can be modified to produce a more efficient algorithm for the two-dimensional problem. We propose several algorithms based on the two-dimensional problem and present results on data from real images to experimentally verify their superiority.

- **Robust algorithms.** We demonstrate that not only are the algorithms we propose efficient, but they are also robust to the types of measurement noise that could be introduced by a poorly calibrated sensor or outliers in the data sets.

We will use projective transformations extensively in our work with images and video, but the class of projective transformations is too restrictive to relate all the images we wish to consider. Unfortunately, in general, the correspondence between an image pair has no simple global parametrization. The only *a priori* restriction on the locations of corresponding points is the well-known epipolar constraint. However, the set of correspondences that are physically realizable is not entirely unconstrained and has a structure that we fully characterize in Chapter 5. The thesis

contributions here are:

- **Correspondence graphs.** We fully describe the class of sets of corresponding points that can arise from a real imaging system. Previous work on correspondence almost always begins with the monotonicity assumption, that the ordering of points along epipolar lines is an image invariant. Instead, we consider the correspondence induced by arbitrarily complicated scenes, and encapsulate this structure in the correspondence graph, the set of all points that are visible in two conjugate epipolar lines. Using the formalism of correspondence graphs, we can ensure that any estimated correspondence is consistent with a physical imaging system.

- **Efficient algorithms.** In addition to describing the topological structure of correspondence graphs, we provide an algorithm by which they can be efficiently constructed for real images. As a result, we can estimate dense, physically consistent correspondence between images taken by widely separated cameras, useful for applications where geometric accuracy is crucial.

The particular application of correspondence graphs that we address in Chapter 6 is the construction of "virtual video". This is physically consistent, synthetic video of a scene from the perspective of a moving camera not present in the original environment. This virtual video can contain perspectives and motion that would have been impractical or impossible to obtain with a real moving camera. Applications include sports video replays, computer games, video conferencing, and special effects. Our contributions here include:

- **Virtual images from wide-baseline stills.** Various techniques for synthesizing a virtual image from two still images have been proposed in the computer graphics community, and generally rely on dense image correspondence. This correspondence is typically estimated using algorithms that make the monotonicity assumption, thus limiting the class of input images that can be processed. The correspondence graph framework discussed in Chapter 5 allows us to create virtual images containing objects that violate the monotonicity assumption, yet still appear in the correct positions. This creates a heightened degree of realism, without resorting to 3-D modeling.

- **True virtual video.** To our knowledge, there were only two types of "virtual video" created before our work. The first consisted of moving a virtual camera around a static scene synthesized from still images. The second required expensive, specialized hardware, many cameras, and highly accurate scene modeling. In contrast, here we describe how to create virtual video from multicamera video, using geometric constraints and only two cameras. We create true virtual video, in the sense that the synthetic video evolves dynamically along with the scene.

- **Efficient estimation algorithms.** The virtual video is produced with an efficient recursive propagation algorithm that builds on the estimation algorithms derived in Chapters 3, 4, and 5, and requires minimal user interaction. We can update the correspondence estimate required to create each virtual frame in a fraction of the time it would take to estimate the correspondence anew for every frame pair, and prove that this fast algorithm is stable in a well-defined sense.

Virtual video can be used to create compelling graphical effects for visualization or entertainment. However, if the synthetic views are actually designed to be good approximations to real video frames, we can realize gains in rendering low frame-rate compressed video. This is especially applicable to the emerging domain of wireless multimedia. The wireless multimedia channel has limited bandwidth, so video data needs to be reduced in both frame size and frame rate. In addition, the multimedia client must reconstruct the video from the transmitted data at a small computational cost due to low power requirements. Virtual frames synthesized with the techniques of Chapter 6 are ideally suited for such requirements, since the information required to render the virtual frames is compact and the rendering algorithm is computationally inexpensive. We explore this idea in Chapter 7. The contributions of the thesis here include:

- **Virtual-view-based time-domain interpolation of video.** We propose a virtual-view-based scheme for video interpolation. We designate a small fraction of non-uniformly spaced frames as anchor frames, and transmit these with good fidelity. The remainder of the frames are reconstructed using a combination of virtual image synthesis and image registration techniques.

- **Low-bit-rate algorithms.** The anchor frames can be selected to be few and far between, and the side information required to synthesize images between them is amortized over potentially

hundreds of intermediate frames. This allows us to achieve bit rates on the order of 45 kbps or less, with better perceptual quality than MPEG-coded video at higher rates.

Taken together, the estimation algorithms introduced in this thesis provide a framework for creating realistic and interactive synthetic video, using algorithms that are well-founded, efficient, stable, and robust. Potential applications range from the high-bandwidth, wide-screen world of enhanced DVD to the low-bandwidth, small-screen domain of PDAs and wireless phones. In either case, the understanding of relationships between multicamera video enables us to create effects that were unheard of just five years ago.

References

[1] S.E. Chen and L. Williams. View Interpolation for Image Synthesis. *Computer Graphics (SIGGRAPH '93)*, pp. 279–288, July 1993.

[2] S.E. Chen. Quicktime VR - An Image-Based Approach to Virtual Environment Navigation. *Computer Graphics (SIGGRAPH '95)*,pp. 29–38, August 1995.

[3] L. McMillan and G. Bishop. Plenoptic Modeling: An Image-Based Rendering System. *Computer Graphics (SIGGRAPH '95)*, pp. 39–46, August 1995.

[4] P. Debevec, C. Taylor, and J. Malik. Modeling and Rendering Architecture from Photographs. *Computer Graphics (SIGGRAPH '96)*, pp. 11–20, August 1996.

[5] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The Lumigraph. *Computer Graphics (SIGGRAPH '96)*, pp. 43–54, August 1996.

[6] M. Levoy and P. Hanrahan. Light Field Rendering. *Computer Graphics (SIGGRAPH '96)*, pp. 31–42, August 1996.

[7] S.M. Seitz and C.R. Dyer. View Morphing. *Computer Graphics (SIGGRAPH '96)*, pp. 21–30, August 1996.

[8] J. Lengyel and J. Snyder. Rendering with Coherent Layers. *Computer Graphics (SIGGRAPH '97)*, pp. 233–242, August 1997.

[9] R. Szeliski and H. Shum. Creating Full View Panoramic Image Mosaics and Environment Maps. *Computer Graphics (SIGGRAPH '97)*, pp. 251–258, August 1997.

[10] D.N. Wood, A. Finkelstein, J.F. Hughes, C.E. Thayer, D.H. Salesin. Multiperspective Panoramas for Cel Animation. *Computer Graphics (SIGGRAPH '97)*, pp. 243–250, August 1997.

[11] J. Shade, S. Gortler, L. Hey, and R. Szeliski. Layered Depth Images. *Computer Graphics (SIGGRAPH '98)*, pp. 231–242, July 1998.

[12] H-Y. Shum and L-W. He. Rendering with Concentric Mosaics. *Computer Graphics (SIGGRAPH '99)*, pp. 299–306, August 1999.

Chapter 2

Preliminaries

In the following chapters, we will speak extensively about cameras, images, and correspondences between them. Here we introduce the notation and terminology that will be used throughout the text, as well as basic background material.

Section 2.1 gives the basic perspective image formation model we will use exclusively. While other, simpler models have been proposed (e.g. orthographic projection), this model accurately reflects the phenomena observed in images taken by real cameras. We introduce the notion of correspondence in Section 2.3 and discuss the fundamental matrix and epipolar geometry that relate image correspondences in the next two sections. Section 2.6 introduces affine transformations, which are special cases of the projective transformations we will discuss in Chapter 4.

We will discuss techniques for estimating several of the quantities introduced here from real data in Chapter 3.

2.1 Cameras and Images

In our analysis, we will consider idealized pinhole cameras. Such a camera $\mathcal{C}$ is described by:

1. A center of projection $O \in \mathbb{R}^3$
2. A focal length $f \in \mathbb{R}^+$
3. An orientation matrix $R \in SO(3)$

The camera's center and orientation are described with respect to a world coordinate system on $\mathbb{R}^3$. A point P expressed in the world coordinate system as $P = (X_o, Y_o, Z_o)$ can be expressed in the camera coordinate system of $\mathcal{C}$ as

$$P = \begin{bmatrix} X_{\mathcal{C}} \\ Y_{\mathcal{C}} \\ Z_{\mathcal{C}} \end{bmatrix} = R \left(\begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} - O \right)$$

The purpose of a camera is to capture a two-dimensional image of a three-dimensional scene $\mathcal{S}$, i.e. a collection of points in $\mathbb{R}^3$. This image is produced by perspective projection as follows.

Each camera $\mathcal{C}$ has an associated image plane $\mathcal{P}$, located in the camera coordinate system at $Z_{\mathcal{C}} = f$. The image plane inherits a natural orientation and two-dimensional coordinate system from the camera coordinate system's XY-plane (see Figure 2.1). At this point we note that the three-dimensional coordinate systems we consider are left-handed. This is a notational convenience, allowing us to make the assumptions that the image plane lies between the center of projection and the scene, and that scene points have positive $Z_{\mathcal{C}}$ coordinates.

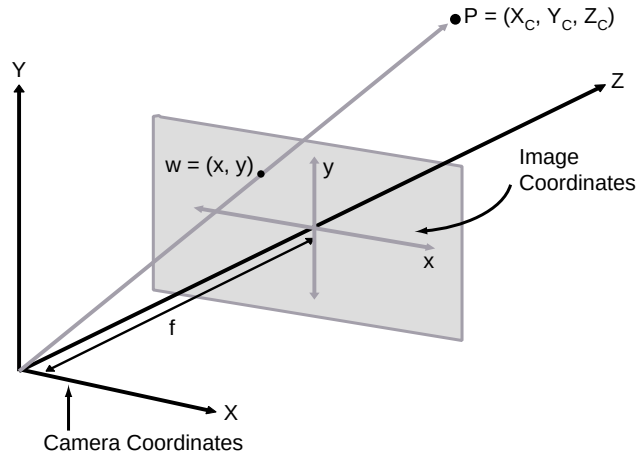


Figure 2.1: Perspective projection.

A scene point $P = (X_{\mathcal{C}}, Y_{\mathcal{C}}, Z_{\mathcal{C}})$ is projected onto the image plane $\mathcal{P}$ at the point $w = (x, y)$ by the perspective projection equations:

$$x = f\frac{X_{\mathcal{C}}}{Z_{\mathcal{C}}} \qquad y = f\frac{Y_{\mathcal{C}}}{Z_{\mathcal{C}}} \tag{2.1}$$

The image $\mathcal{I}$ that is produced is a map from $\mathcal{P}$ into a color space $\mathbf{C}$. The color of a point is typically a real-valued intensity or a triplet of RGB or YUV values. While the entire ray of scene points $\{\kappa(x, y, f)|\kappa > 0\}$ is projected to the image coordinate (x, y) by (2.1), the point on this ray that gives (x, y) its color in the image $\mathcal{I}$ is the one closest to the image plane (i.e. that point with minimal κ). This point is said to be visible; any scene point further along on the same ray is said to be occluded.

Technically, we should be careful about the relationship between the color of image points and the color of scene points. To simplify matters, we assume that scene points have the same color regardless of the viewing angle (this is called the Lambertian assumption), and that the color of an image point is the same as the color of a single corresponding scene point. In practice, the colors of corresponding image and scene points are different due to a host of factors in a real imaging system. These include the point spread function, color space, and dynamic range of the camera, as well as non-Lambertian or semi-transparent objects in the scene. For more detail on the issues involved in image formation, see [1, 2].

2.2 Homogeneous Coordinates

In certain situations it is advantageous to describe the image coordinate (x, y) by the homogeneous coordinate $\kappa(x, y, 1)$, where $\kappa \neq 0$. Clearly the image coordinate of a homogeneous coordinate (x, y, z) can be recovered as $(\frac{x}{z}, \frac{y}{z})$ when $z \neq 0$. Similarly, any scene point (X, Y, Z) can be represented in homogeneous coordinates as $\kappa(X, Y, Z, 1)$, where $\kappa \neq 0$. We use the symbol $\equiv$ to denote the equivalence between a homogeneous coordinate and a non-homogeneous one.

A camera $\mathcal{C}$ with parameters (O, f, R) can be represented by a 3×4 matrix $\Pi_{\mathcal{C}}$ that multiplies a scene point expressed as a homogeneous coordinate in $\mathbb{R}^4$ to produce an image point expressed as a homogeneous coordinate in $\mathbb{R}^3$. When the scene point is expressed in the world coordinate system, the matrix Π is given by

$$\Pi_{\mathcal{C}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} [R, \; -RO]$$

Then

$$
\begin{aligned}
\Pi_{\mathcal{C}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} &= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} [R,\ -RO] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} R \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - O \right) \\
&= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{\mathcal{C}} \\ Y_{\mathcal{C}} \\ Z_{\mathcal{C}} \end{bmatrix} \\
&= \begin{bmatrix} fX_{\mathcal{C}} \\ fY_{\mathcal{C}} \\ Z_{\mathcal{C}} \end{bmatrix} \\
&\equiv \begin{bmatrix} f\frac{X_{\mathcal{C}}}{Z_{\mathcal{C}}} \\ f\frac{X_{\mathcal{C}}}{Z_{\mathcal{C}}} \end{bmatrix}
\end{aligned}
$$

2.3 Two Cameras

Consider the situation in Figure 2.2, in which the same static scene is imaged by two cameras $\mathcal{C}_0$ and $\mathcal{C}_1$. These could be two physically separate cameras, or a single moving camera at different points in time. In the latter setting it is natural to say the induced images are related due to camera motion. Let the scene coordinates of a point P in the $\mathcal{C}_0$ coordinate system be (X, Y, Z), and in the $\mathcal{C}_1$ coordinate system be (X', Y', Z'). We denote the corresponding image coordinates of P in $\mathcal{P}_0$ and $\mathcal{P}_1$ by $w = (x, y)$ and $w' = (x', y')$, respectively. The points w and w' are said to be corresponding points, and the pair (w, w') is called a point correspondence.

The scene point P is projected onto the image points w and w' via the perspective projection equations (2.1):

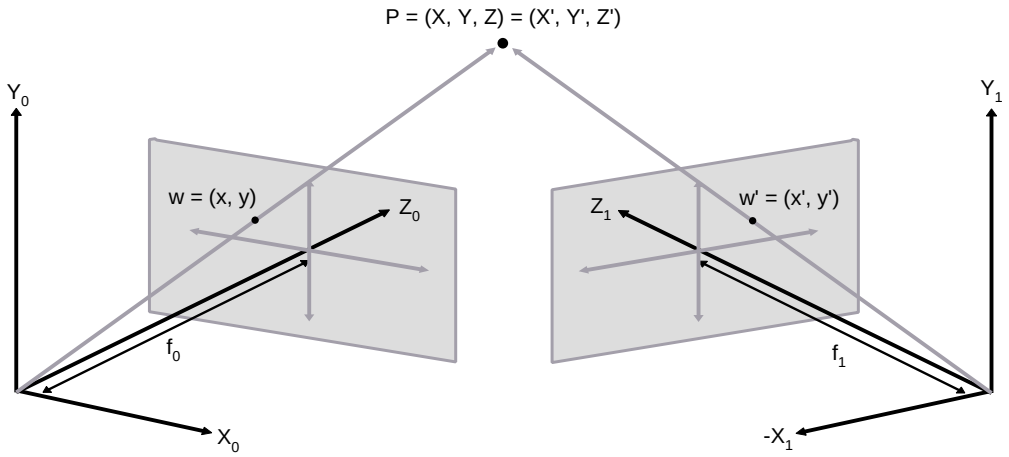


Figure 2.2: Rigid motion of a camera.

$$x = f_0 \frac{X}{Z} \qquad y = f_0 \frac{Y}{Z}$$
$$x' = f_1 \frac{X'}{Z'} \qquad y' = f_1 \frac{Y'}{Z'}$$

Here f_0 and f_1 are the focal lengths of $\mathcal{C}_0$ and $\mathcal{C}_1$, respectively. We assume that the two cameras are related by a rigid motion, which means that the $\mathcal{C}_1$ coordinate system can be expressed as a rotation R of the $\mathcal{C}_0$ coordinate system followed by a translation $[t_X\ t_Y\ t_Z]^T$. That is,

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix} \tag{2.2}$$

In terms of the parameters of the cameras, $R = R_1 R_0^{-1}$ and $t = R_1(O_0 - O_1)$. Alternately, we can write R as

$$R = \begin{pmatrix} \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & \cos\beta\sin\gamma & -\sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma \\ -\cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \cos\beta\cos\gamma & \sin\alpha\sin\gamma + \cos\alpha\sin\beta\cos\gamma \\ \sin\alpha\cos\beta & -\sin\beta & \cos\alpha\cos\beta \end{pmatrix} \tag{2.3}$$

where α, β and γ are rotation angles around the X, Y, and Z axes, respectively, of the $\mathcal{C}_0$ coordinate system.

By substituting equation (2.2) into the perspective projection equations (2.1), we obtain a relationship between the two sets of image coordinates:

$$\begin{aligned} x' &= f_1 \frac{X'}{Z'} \\ &= \frac{r_{11}\frac{f_1}{f_0}x + r_{12}\frac{f_1}{f_0}y + r_{13}f_1 + \frac{t_X f_1}{Z}}{\frac{r_{31}}{f_0}x + \frac{r_{32}}{f_0}y + r_{33} + \frac{t_Z}{Z}} \end{aligned} \tag{2.4}$$

$$\begin{aligned} y' &= f_1 \frac{Y'}{Z'} \\ &= \frac{r_{21}\frac{f_1}{f_0}x + r_{22}\frac{f_1}{f_0}y + r_{23}f_1 + \frac{t_Y f_1}{Z}}{\frac{r_{31}}{f_0}x + \frac{r_{32}}{f_0}y + r_{33} + \frac{t_Z}{Z}} \end{aligned} \tag{2.5}$$

Here the r_{ij} are the elements of the rotation matrix given in (2.3). In Section 2.6 and Chapter 4 we will consider some special cases of (2.4)-(2.5).

2.4 The Fundamental Matrix

The following theorem from [3] expresses an important fact about the correspondence between two images of the same scene:

Theorem 2.1: *For every pair of cameras $(\mathcal{C}_0, \mathcal{C}_1)$ in which the camera centers are separated by a non-zero translation, there exists a matrix F of rank two such that for all correspondences $(w, w') = ((x, y), (x', y')) \in \mathcal{P}_0 \times \mathcal{P}_1$,*

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T F \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Proof. Let the cameras $\mathcal{C}_0$ and $\mathcal{C}_1$ have parameters (O_0, f_0, R_0) and (O_1, f_1, R_1), respectively, with $O_1 \neq O_0$. Fix a correspondence $(w, w') \in \mathcal{P}_0 \times \mathcal{P}_1$, and let P be the associated scene point. Let P be expressed in the coordinate systems of $\mathcal{C}_0$ and $\mathcal{C}_1$ by (X, Y, Z) and (X', Y', Z'), respectively.

Let (R, t) relate the coordinate systems of $(\mathcal{C}_0, \mathcal{C}_1)$ so that:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \right) \tag{2.6}$$

Here $R = R_1 R_0^{-1}$ and $t = R_0(O_0 - O_1)$. From (2.6) it follows that the vectors $(X', Y', Z')^T$, $R(X, Y, Z)^T$, and Rt are linearly dependent (i.e. coplanar). Therefore, we can write:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \cdot R \left(t \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = 0 \tag{2.7}$$

We can rewrite (2.7) as

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}^T RT \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = 0 \tag{2.8}$$

where

$$T = \begin{bmatrix} 0 & -t_Z & t_Y \\ t_Z & 0 & -t_X \\ -t_Y & t_X & 0 \end{bmatrix}$$

Now, by perspective projection, the image coordinates are:

$$x = f_0 \frac{X}{Z} \quad y = f_0 \frac{Y}{Z}$$

$$x' = f_1 \frac{X'}{Z'} \quad y' = f_1 \frac{Y'}{Z'}$$

We can substitute into (2.8) to produce

$$(ZZ') \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T \begin{bmatrix} \frac{1}{f_1} & & \\ & \frac{1}{f_1} & \\ & & 1 \end{bmatrix} RT \begin{bmatrix} \frac{1}{f_0} & & \\ & \frac{1}{f_0} & \\ & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0 \tag{2.9}$$

Since neither of (Z, Z') is 0 by assumption, (2.9) implies

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T \begin{bmatrix} \frac{1}{f_1} & & \\ & \frac{1}{f_1} & \\ & & 1 \end{bmatrix} RT \begin{bmatrix} \frac{1}{f_0} & & \\ & \frac{1}{f_0} & \\ & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

This is the statement of the theorem, with $F = A_1^{-T} RT A_0^{-1}$, and $A_i = \mathrm{diag}(f_i, f_i, 1)$. The matrix F is of rank two since A_0, A_1, and R are always nonsingular, and T is rank 2 for any non-zero $t \in \mathbb{R}^3$. Since F depends only on the rigid motion relating the cameras, and not on the choice of correspondence, the theorem is proved. ∎

Any matrix F satisfying Theorem 2.1 for a camera pair $(\mathcal{C}_0, \mathcal{C}_1)$ is called a fundamental matrix for $(\mathcal{C}_0, \mathcal{C}_1)$. The matrix $E = RT$ is called the essential matrix for $(\mathcal{C}_0, \mathcal{C}_1)$.

The fundamental matrix is unique up to scale provided that there exists no quadric surface $\mathcal{Q}$ containing the line $\overline{O_0 O_1}$ and every point in $\mathcal{S}$ [4].

2.5 Epipolar Geometry

Given the fundamental matrix F for a camera pair $(\mathcal{C}_0, \mathcal{C}_1)$, a constraint on the possible locations of correspondences between the associated image pair $(\mathcal{I}_0, \mathcal{I}_1)$ can be obtained.

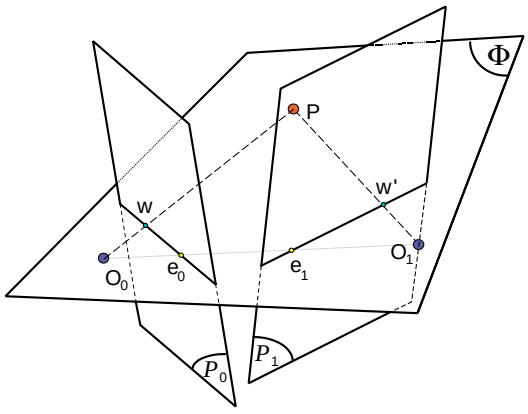


Figure 2.3: Epipolar geometry.

Definition. The epipolar line of a point $w \in \mathcal{P}_0$ is the set of points:

$$\ell_w = \left\{ w' = (x', y')^T \in \mathcal{P}_1 \left| \begin{bmatrix} w' \\ 1 \end{bmatrix}^T F \begin{bmatrix} w \\ 1 \end{bmatrix} = 0 \right. \right\}$$

If w is the image of scene point P in $\mathcal{P}_0$, the image of P in $\mathcal{P}_1$ is constrained to lie on the epipolar line ℓ_w. Epipolar lines for points in $\mathcal{P}_1$ can be defined accordingly. Hence, epipolar lines exist in conjugate pairs (ℓ_0, ℓ_1), such that the match to a point $w \in \ell_0$ must lie on ℓ_1, and vice versa. Conjugate epipolar lines are generated by intersecting any plane Φ containing the baseline $\overline{O_0 O_1}$ with the pair of image planes $(\mathcal{P}_0, \mathcal{P}_1)$ (see Figure 2.3). For a thorough review of epipolar geometry, see [5].

The epipoles $e_0 \in \mathcal{P}_0$ and $e_1 \in \mathcal{P}_1$ are the projections of the camera centers O_1 and O_0 onto $\mathcal{P}_0$ and $\mathcal{P}_1$, respectively. It can be seen from Figure 2.3 that the epipolar lines in each image all intersect at the epipole. In fact, the homogeneous coordinates of the epipoles e_0 and e_1 are the right and left eigenvectors of F, respectively, corresponding to the eigenvalue 0.

2.6 Affine Transformations

One of the fundamental relationships in image processing is the affine transformation. An affine transformation maps a point $w \in \mathbb{R}^2$ to $w' \in \mathbb{R}^2$ by:

$$w' = Aw + b$$

where $A \in GL(2)$ and $b \in \mathbb{R}^2$. Some important special cases are:

- **Translation:** $A = I$
- **Rotation:** $A = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, b = 0$
- **Scaling:** $A = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}, b = 0, \alpha, \beta \neq 0$
- **Shear:** $A = \begin{bmatrix} 1 & \kappa \\ 0 & 1 \end{bmatrix}, b = 0$

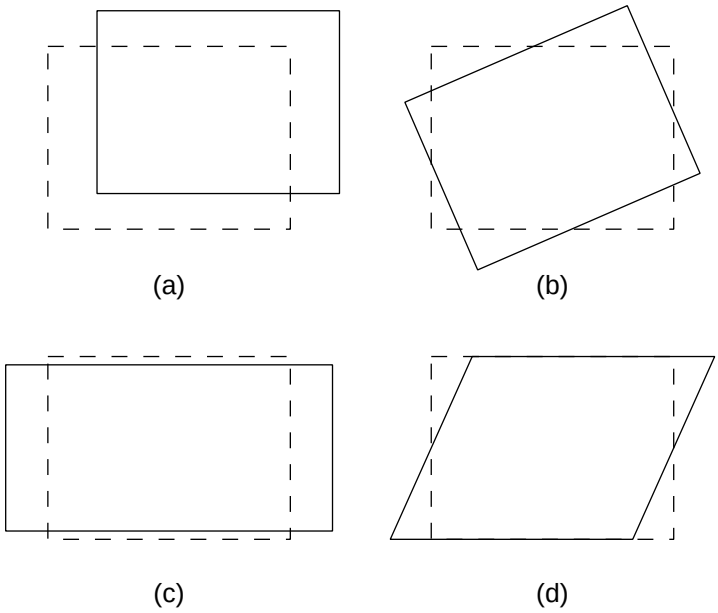


Figure 2.4: Special affine transformations. (a) Translation, (b) rotation, (c) scaling, (d) shear.

The effects of these operations on a rectangle centered at the origin are illustrated in Figure 2.4. In fact, any affine transformation can be expressed as a composition of these four special operations. Any nonsingular matrix A can be factored as $A = QR$, where Q is orthogonal (i.e. a rotation matrix) and R is upper triangular [6]. Then we can write R as a composition of a scaling and a shear:

$$R = \begin{bmatrix} a & b \\ 0 & c \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & c \end{bmatrix} \begin{bmatrix} 1 & \frac{b}{a} \\ 0 & 1 \end{bmatrix}$$

The relationship between the coordinates of two images of the same scene is often modeled as an affine transformation to be estimated. We pause to determine when this modeling assumption is well-founded. Reconsider equations (2.4) and (2.5) that relate the image coordinates of a point seen by two cameras $\mathcal{C}_0$ and $\mathcal{C}_1$:

$$\begin{aligned} x' &= \frac{r_{11}\frac{f_1}{f_0}x + r_{12}\frac{f_1}{f_0}y + r_{13}f_1 + \frac{t_X f_1}{Z}}{\frac{r_{31}}{f_0}x + \frac{r_{32}}{f_0}y + r_{33} + \frac{t_Z}{Z}} \\ y' &= \frac{r_{21}\frac{f_1}{f_0}x + r_{22}\frac{f_1}{f_0}y + r_{23}f_1 + \frac{t_Y f_1}{Z}}{\frac{r_{31}}{f_0}x + \frac{r_{32}}{f_0}y + r_{33} + \frac{t_Z}{Z}} \end{aligned}$$

For this to be an affine transformation that relates every image correspondence, we require:

- $r_{31} = 0$
- $r_{32} = 0$
- $\frac{t_X}{Z}, \frac{t_Y}{Z}, \frac{t_Z}{Z}$ constant for all scene points

From the form of the rotation matrix (2.3), the first two conditions imply that the rotation angles α and β are 0, i.e. the image planes are both parallel to the XY-plane. The third condition implies that either the translation vector t is identically 0, or that Z is constant for all points in the scene, i.e. the scene is a planar surface parallel to the image planes $\mathcal{P}_0$ and $\mathcal{P}_1$.

Therefore, an affine transformation is induced by the motion of a perspective camera only under somewhat restrictive conditions. However, the affine assumption is often made when the scene is far from the camera (Z is large) and the rotation angles α and β are very small. This assumption has the advantage that the affine parameters can be efficiently estimated. We discuss such estimation techniques in Section 3.1.

2.7 References

[1] B.K.P. Horn. *Robot Vision.* MIT Press, 1986.

[2] V.S. Nalwa. *A Guided Tour of Computer Vision.* Addison-Wesley, 1993.

[3] H.C. Longuet-Higgins. A Computer Algorithm for Reconstructing a Scene from Two Projections. *Nature*, Vol. 293, pp. 133–135, September 10, 1981.

[4] S.J. Maybank. The Angular Velocity Associated with the Optical Flowfield Arising from Motion Through a Rigid Environment. *Proc. Royal Soc. London A*, vol. 401, pp. 317–326, 1985.

[5] Z. Zhang. Determining the Epipolar Geometry and its Uncertainty: A Review. *International Journal of Computer Vision*, vol. 27, no. 2, pp. 161–195, 1998.

[6] G. Strang. *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, 1988.

Chapter 3

Basic Estimation Problems

In this section, we briefly review standard methods for solving various estimation problems in digital video that we will encounter in the text.

We begin in Section 3.1 with the problem of estimating an affine transformation. We present the classical linear least-squares solution from point matches; we will see in Chapter 4 how it is related to the more difficult nonlinear least-squares problem of estimating a projective transformation. In the case when point matches are not available, we briefly review correlation and Fourier-based methods for estimating translation, rotation, and scale parameters. A wide variety of techniques for the estimation of affine transformations is reviewed in [1].

As we saw in Chapter 2, the conjugate epipolar lines in an image pair are determined by the fundamental matrix. Since we make use of the epipolar geometry to analyze correspondence in Chapter 5, we discuss algorithms for estimating the fundamental matrix in Section 3.2. In Section 3.3 we discuss the related issue of estimating a pair of rectifying transformations, which facilitate working with epipolar lines in computer programs.

In Section 3.4 we turn to the issue of automatically obtaining a set of point matches between an image pair, a preliminary step for several of the estimation problems we will discuss. In Chapter 2 we defined a correspondence as a pair of points (w, w') that are the projections of some scene point P onto a pair of image planes $(\mathcal{P}_0, \mathcal{P}_1)$. Ideally, we would like each point match to be a correspondence, but in practice the point matches are correspondences corrupted by noise. We conclude in Section 3.5 with a discussion of algorithms for estimating dense correspondence over an image pair. These algorithms attempt to provide a match in $\mathcal{P}_1$ for every point in $\mathcal{P}_0$.

3.1 Estimating Affine Transformations

3.1.1 The Linear Least-Squares Problem

Suppose we possess a set of point matches $\{(w_j, w'_j) \in \mathbb{R}^2 \times \mathbb{R}^2, j = 1, \ldots, N\}$. We assume that these are noisy samples of a fixed but unknown affine transformation $M^* = (A, b)$, so that

$$w'_j = Aw_j + b + e_j \quad j = 1, \ldots, N$$

where the e_j are small errors. When the errors are modeled as zero-mean iid Gaussian random variables, the maximum likelihood estimate of the parameters (A, b) is the minimizer $(\hat{A}, \hat{b})$ of the least-squares functional

$$Q(A, b) = \frac{1}{2} \sum_{j=1}^{N} \left(w'_j - Aw_j - b\right)^T \left(w'_j - Aw_j - b\right) \tag{3.1}$$

This estimate is also the minimum-variance unbiased estimate of (A, b). The minimizer of the linear least squares functional (3.1) is well known and can be expressed as:

$$[\hat{A} \quad \hat{b}] = VW^{-1} \tag{3.2}$$

where $W \in \mathbb{R}^{3\times3}$, $V \in \mathbb{R}^{2\times3}$ are given by

$$W = \begin{bmatrix} \sum_{j=1}^{N} w_j w_j^T & \sum_{j=1}^{N} w_j \\ \\ \sum_{j=1}^{N} w_j^T & N \end{bmatrix}$$

$$V = \begin{bmatrix} \sum_{j=1}^{N} w'_j w_j^T & \sum_{j=1}^{N} w'_j \end{bmatrix}$$

If the contribution of the j^{th} data pair is to be weighted by λ_j,[1] the corresponding least-squares functional is

$$Q(A, b) = \frac{1}{2} \sum_{j=1}^{N} \lambda_j \left(w'_j - Aw_j - b\right)^T \left(w'_j - Aw_j - b\right)$$

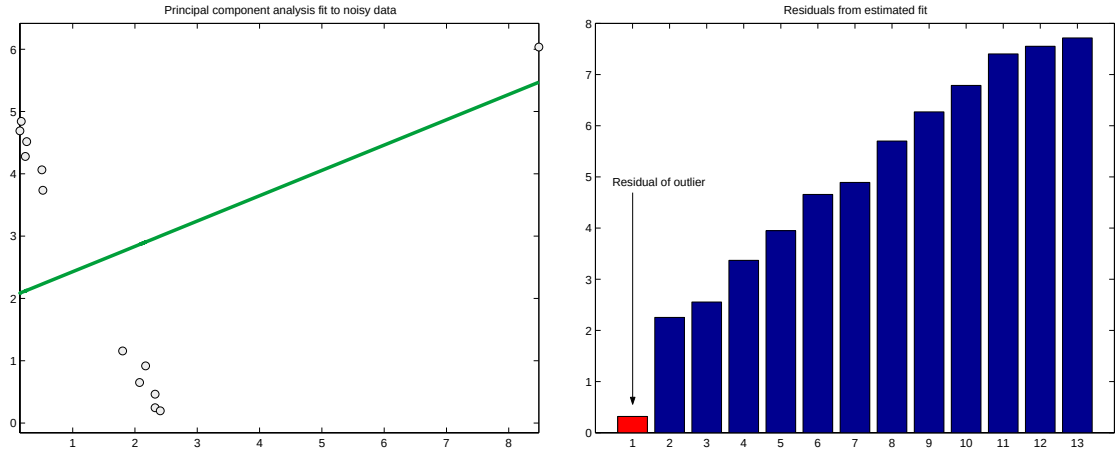
[1] For example, if the errors e_j are zero-mean independent Gaussian variables, with variances σ_j^2, then setting $\lambda_j = \sigma_j^{-2}$ yields the maximum likelihood estimate of (A, b).

and the solution is given by (3.2) with W and V replaced by

$$W = \begin{bmatrix} \sum_{j=1}^{N} \lambda_j w_j w_j^T & \sum_{j=1}^{N} \lambda_j w_j \\ \\ \sum_{j=1}^{N} \lambda_j w_j^T & N \sum_{j=1}^{N} \lambda_j \end{bmatrix}$$

$$V = \begin{bmatrix} \sum_{j=1}^{N} \lambda_j w'_j w_j^T & \sum_{j=1}^{N} \lambda_j w'_j \end{bmatrix}$$

This is a natural approach to estimation even when the errors are not Gaussian. However, a data point with a very large e_j can dominate the estimation and pull the minimizer away from the underlying set of parameters (see Figure 3.1a).



(a) (b)

Figure 3.1: Effects of outliers. (a) Data points and least-squares estimate, (b) Residuals $\hat{e}_j$

Unfortunately, the residual error $\hat{e}_j = w'_j - (\hat{A}w_j + \hat{b})$ of an outlying point can be quite small (see Figure 3.1b). Various techniques have been proposed for identifying and rejecting outliers [2, 3, 4]. In practice, we use the least-median-of-squares (or "X84") algorithm proposed by Rousseeuw and Levoy [5]. Specifically, if m is the median of $\{\hat{e}_j, j = 1, \ldots, N\}$, we compute the median absolute deviation

$$\text{median}_j \, |\hat{e}_j - m|$$

or the mean absolute deviation

$$\frac{1}{N}\sum_{j=1}^{N}|\hat{e}_j - m|$$

and reject points whose residual lies more than some number of deviations from the median. The advantage of this technique is its robustness to outliers in both x and y directions. Up to half of the data points can be outliers and still be correctly rejected.

The least-median-square estimator is one of a more general class of robust estimators called M-estimators [6].

3.1.2 Estimating Translation

Estimating the translation between a pair of images is a classical problem in computer vision. When we possess a set of noisy feature correspondences $\{(w_j, w'_j) \in \mathbb{R}^2 \times \mathbb{R}^2, j = 1, \ldots, N\}$, the maximum likelihood estimate of the translation is simply the mean difference $\hat{b} = \frac{1}{N}\sum_{j=1}^{N}(w'_j - w_j)$. Zagorodnov [7] discussed how to stabilize the variance of translation estimates between a sequence of images using multiple pairwise estimates.

In the absence of a set of matched feature correspondences, the problem may be posed as a least-squares problem over the entire image pair:

$$\min_{(b_1,b_2)\in\mathcal{N}} \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}\left(\mathcal{I}_1(x,y) - \mathcal{I}_0(x+b_1, y+b_2)\right)^2 \tag{3.3}$$

This expression assumes that the images always overlap in MN pixels, and that the search for the best translation vector (b_1, b_2) is conducted over a neighborhood $\mathcal{N}$ of $\mathbb{R}^2$.

When the search neighborhood $\mathcal{N}$ is large compared to the dimensions of the images, solving (3.3) by a brute-force correlation search is generally quite time-consuming. A standard approach is to subsample the images $\mathcal{I}_0$ and $\mathcal{I}_1$ to an extent such that a search over all the translation vectors in $\mathcal{N}$ is practical, and then propagate the translation estimates from each coarsely subsampled image pair to a more finely subsampled image pair. At the higher level, the search neighborhood is confined to lie near the previous translation estimate. This process continues until a translation estimate is obtained for the full-resolution images.

The following pseudocode illustrates a coarse-to-fine algorithm for estimating the translation between two $N \times N$ images with ℓ levels of subsampling.

Algorithm 3.1: *Estimating translation between an image pair.*

1. *Subsample the images* $(\mathcal{I}_0, \mathcal{I}_1)$ *by a factor of* 2^ℓ.
2. *Initialize the search neighborhood* $\mathcal{N} = [-\frac{N}{2^\ell}, \frac{N}{2^\ell}]^2$.
3. *For level=1 to* ℓ*:*
   (a) *Initialize* $C^* = \infty$.
   (b) *For* $(b_1, b_2) \in \mathcal{N}$*:*
      i. $x_{\min} = \max(1, -b_1)$, $x_{\max} = \min(N, N - b_1)$,
         $y_{\min} = \max(1, -b_2)$, $y_{\max} = \min(N, N - b_2)$.
      ii. $C = \frac{1}{(x_{\max} - x_{\min})(y_{\max} - y_{\min})} \sum_{x=x_{\min}}^{x_{\max}} \sum_{y=y_{\min}}^{y_{\max}} (\mathcal{I}_1(x, y) - \mathcal{I}_0(x + b_1, y + b_2))^2$.
      iii. *If* $C < C^*$*,*
         A. $C^* \leftarrow C$
         B. $(b_1^*, b_2^*) \leftarrow (b_1, b_2)$
   (c) *Upsample* $(\mathcal{I}_0, \mathcal{I}_1)$ *by a factor of 2.*
   (d) $\mathcal{N} \leftarrow [2b_1^* - 1, 2b_1^* + 1] \times [2b_2^* - 1, 2b_2^* + 1]$

This algorithm requires a factor of approximately $2^{4\ell}$ less arithmetic operations than a brute-force solution of (3.3). Since only a fraction of the translation vectors are tested, there is no guarantee that (b_1^*, b_2^*) will be the true minimizer. However, in practice, if the images are sufficiently smooth and the most coarsely subsampled images still resemble the originals, the estimate is good, and can be computed with fewer operations than it would take to detect and match features.

Another popular method for estimating the translation between an image pair operates in the frequency domain, using phase correlation [8]. Given two images $\mathcal{I}_0(x, y)$ and $\mathcal{I}_1(x, y)$, denote their Fourier transforms by

$$\begin{aligned} \mathcal{J}_0(\xi, \eta) &= \mathcal{F}(\mathcal{I}_0(x, y)) \\ \mathcal{J}_1(\xi, \eta) &= \mathcal{F}(\mathcal{I}_1(x, y)) \end{aligned}$$

Then writing the Fourier transforms in the magnitude-phase form

$$\begin{aligned} \mathcal{J}_0(\xi,\eta) &= M_0(\xi,\eta)e^{j\phi_0(\xi,\eta)} \\ \mathcal{J}_1(\xi,\eta) &= M_1(\xi,\eta)e^{j\phi_1(\xi,\eta)} \end{aligned}$$

we can define the phase correlation function $d(x,y)$ by

$$d(x,y) = \mathcal{F}^{-1}(e^{j(\phi_0(\xi,\eta)-\phi_1(\xi,\eta))}) \tag{3.4}$$

By the Fourier shift theorem, if $\mathcal{I}_1(x,y) = \mathcal{I}_0(x+x_0, y+y_0)$ for some shift (x_0,y_0), then the corresponding phase correlation function $d(x,y)$ is an impulse centered at (x_0,y_0). Hence, the translation between an image pair can be estimated by forming the phase correlation function (3.4) and searching for a global maximizer. One advantage of this technique is its efficient implementation via the Fast Fourier Transform. It is also robust to illumination scale or shift in the original image pair, or noise concentrated in a narrow band of the frequency domain.

3.1.3 Estimating Rotation and Scale

Translation estimation techniques can be extended to further estimate the rotation and scale difference between an image pair. Consider two images whose coordinates are related by rotation and an isotropic scaling:

$$\mathcal{I}_1(x,y) = \mathcal{I}_0(a(x\cos\gamma + y\sin\gamma), -a(x\sin\gamma - y\cos\gamma)) \tag{3.5}$$

Then by a change to so-called log-polar coordinates $\rho = \log\sqrt{x^2+y^2}$, $\theta = \tan^{-1}\frac{y}{x}$, (3.5) becomes

$$\mathcal{I}_1(\rho,\theta) = \mathcal{I}_0(\rho + \log a, \theta - \gamma)$$

The rotation angle and scaling factor can then be recovered by a translation estimation algorithm. Reddy and Chatterji [9] suggested using phase correlation, operating in the frequency domain and using properties of the Fourier transform. Wolberg and Zokai [10] proposed a coarse-to-fine multiresolution algorithm that operates in the spatial domain and simultaneously estimates the translation between the image pair.

3.2 Estimating the Fundamental Matrix

As in the projective transformation estimation problem, our objective is to select the matrix $F \in \mathbb{R}^{3\times 3}$ that best matches a given set of point mappings:

$$\{w_j \mapsto w'_j \in \mathbb{R}^2, j = 1, \ldots, N\}$$

A special case arises when the data consists of noisy samples of a fixed but unknown fundamental matrix F:

$$\begin{bmatrix} w'_j \\ 1 \end{bmatrix}^T F \begin{bmatrix} w_j \\ 1 \end{bmatrix} = e_j,\ j = 1 \ldots N$$

Therefore, it is natural to try to minimize a least-squares cost functional such as

$$J(F) = \sum_{j=1}^{N} \begin{bmatrix} w'_j \\ 1 \end{bmatrix}^T F \begin{bmatrix} w_j \\ 1 \end{bmatrix} \tag{3.6}$$

over the class of admissible fundamental matrices. We recall that F must have rank two (see Section 2.4). Furthermore, the fundamental matrix is unique up to scale, so we must fix some scaling (say, $\|F\| = 1$ for some appropriate norm) to ensure that J cannot become arbitrarily small. Hence, the class of admissible estimates has only seven degrees of freedom. Constrained minimizations of this type are problematic due to the difficulty in parameterizing the class of admissible F. Faugeras and Luong [11, 12, 13] proposed some solutions in this regard and analyzed various cost functionals for the estimation problem.

The approach we take in practice to estimating the fundamental matrix is due to Hartley [14]. Ignoring the rank-two constraint for the moment, we minimize (3.6) over the class of F with Frobenius norm 1.

Each correspondence (w_j, w'_j) produces a linear equation in the elements of F:

$$x_j x'_j f_{11} + x_j y'_j f_{21} + x_j f_{31} + y_j x'_j f_{21} + y_j y'_j f_{22} + y_j f_{23} + x'_j f_{31} + y'_j f_{32} + f_{33} = 0$$

The equations in all the data points can be collected into a linear system $Af = 0$, where A is an $N \times 9$ matrix involving the data, and $f = (f_{11}, f_{21}, f_{31}, f_{12}, f_{22}, f_{32}, f_{13}, f_{23}, f_{33})^T$ is the vector

of unknowns. The least-squares minimization problem is then

$$\begin{aligned} \min \quad & \|Af\|_2 \\ s.t. \quad & f^T f = 1 \end{aligned}$$

The solution to this is well known and the minimizer is the eigenvector $f \in \mathbb{R}^9$ of $A^T A$ corresponding to the minimal eigenvalue. This can be computed via the SVD. This eigenvector is then reassembled into a 3×3 matrix $\hat{F}$.

To account for the rank-two constraint, we replace the full-rank estimate $\hat{F}$ by $\hat{F}^*$, the minimizer of

$$\begin{aligned} \min \quad & \|\hat{F} - \hat{F}^*\|_F \\ s.t. \quad & rank(\hat{F}^*) = 2 \end{aligned} \tag{3.7}$$

Given the singular value decomposition $\hat{F} = UDV^T$, where $D = \mathrm{diag}(r, s, t)$ with $r > s > t$, the solution to (3.7) is

$$\hat{F}^* = U\hat{D}V^T$$

where $\hat{D} = \mathrm{diag}(r, s, 0)$.

Furthermore, the data is normalized by translation to the origin and isotropic scaling before the estimation to maintain numerical stability. If T and T' are the 3×3 normalizing transformations applied to the homogeneous coordinates of the w_j and w'_j, then the estimate of the fundamental matrix in the original coordinates is given by

$$F = T'^T \hat{F}^* T$$

3.3 Estimating Rectifying Projective Transformations

Since epipolar lines are generally not aligned with one of the coordinate axes of an image, or even parallel, the computer implementation of algorithms that work with epipolar lines can be complicated. To this end, it is common in computer vision algorithms to apply a technique called rectification to an image pair before processing, so that the epipolar lines are made parallel and horizontal.

Definition. An associated image plane pair $(\mathcal{P}_0, \mathcal{P}_1)$ is said to be rectified when the fundamental matrix for $(\mathcal{P}_0, \mathcal{P}_1)$ is the skew-symmetric matrix

$$F_* = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.8}$$

In homogeneous coordinates, the epipoles corresponding to F_* are $e_0 = e_1 = [1\ 0\ 0]^T$, which means the epipolar lines are horizontal and parallel. Furthermore, expanding the fundamental matrix equation for a correspondence $((x, y)^T, (x', y')^T) \in \mathcal{P}_0 \times \mathcal{P}_1$,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T F_* \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

which is equivalent to $y' - y = 0$. This implies that not only are the epipolar lines horizontal, they are aligned, so that the lines $y = \lambda$ in $\mathcal{P}_0$ and $y' = \lambda$ in $\mathcal{P}_1$ are conjugate epipolar lines.

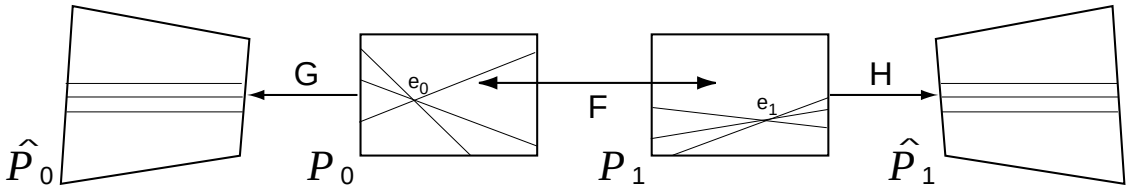


Figure 3.2: Rectifying projective transformations.

Definition. Two projective transformations (G, H) are called rectifying projective transformations for an associated image plane pair $(\mathcal{P}_0, \mathcal{P}_1)$ with fundamental matrix F if

$$H^{-T} F G^{-1} = F_* \tag{3.9}$$

By the above definition, if the projective transformations G and H are applied to $\mathcal{P}_0$ and $\mathcal{P}_1$ to produce warped image planes $\hat{\mathcal{P}}_0$ and $\hat{\mathcal{P}}_1$, respectively, then $(\hat{\mathcal{P}}_0, \hat{\mathcal{P}}_1)$ is a rectified pair (Figure 3.2). Efficient techniques for image warping are discussed by Wolberg [15].

Equation (3.9) may be rewritten and expanded to reflect its dependence on the entries of G and H:

$$\begin{aligned} F &= H^T F_* G \\ &= \begin{bmatrix} \bar{a}_{11} & \bar{a}_{12} & \bar{b}_1 \\ \bar{a}_{21} & \bar{a}_{22} & \bar{b}_2 \\ \bar{c}_1 & \bar{c}_2 & 1 \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ c_1 & c_2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \bar{a}_{21}c_1 - \bar{c}_1 a_{21} & \bar{a}_{21}c_2 - \bar{c}_1 a_{22} & \bar{a}_{21} - \bar{c}_1 b_2 \\ \bar{a}_{22}c_1 - \bar{c}_2 a_{21} & \bar{a}_{22}c_2 - \bar{c}_2 a_{22} & \bar{a}_{22} - \bar{c}_2 b_2 \\ \bar{b}_2 c_1 - a_{21} & \bar{b}_2 c_2 - a_{22} & \bar{b}_2 - b_2 \end{bmatrix} \end{aligned} \tag{3.10}$$

From (3.10) it is clear that the rectifying condition (3.9) may be expressed as 9 equations in the 16 unknowns of the two projective transformations G and H. Specifically, the parameters $(a_{11}, a_{12}, b_1, \bar{a}_{11}, \bar{a}_{12}, \bar{b}_1)$ are entirely unconstrained aside from the requirement that their choice not make $|G| = 0$ or $|H| = 0$. Hence, there are 7 degrees of freedom in the choice of a rectifying pair (G, H).

The importance of rectification has been known for many years in the field of photogrammetry [16]. However, older methods for obtaining rectifying projective transformations for an image plane pair generally relied on knowledge of camera parameters.

Seitz [17] and Hartley [18] described methods for deriving rectifying projective transformations from an estimate of the fundamental matrix relating an image pair. Isgrò and Trucco [19] observed that the rectifying transformations may be estimated without explicitly estimating the fundamental matrix as an intermediate step.

In practice, we use Seitz's method for obtaining a rectified image pair, though it has its drawbacks (see Section 5.6). First, an arbitrary plane E parallel to the baseline $\overline{O_0 O_1}$ is selected that intersects $\mathcal{P}_0$ and $\mathcal{P}_1$ in two lines d_0 and d_1, respectively (Figure 3.3). The two image planes may be made parallel by rotating image plane $\mathcal{P}_i$ through a certain angle θ_i about the line d_i (or any line parallel to d_i).

The epipoles $e_0 = (e_{0x}, e_{0y})$ and $e_1 = (e_{1x}, e_{1y})$ represent the projections of the camera centers O_1 and O_0 onto the image planes $\mathcal{P}_0$ and $\mathcal{P}_1$, respectively. Therefore, an image plane parallel to the baseline has its epipole at infinity (i.e. has third homogeneous coordinate equal to 0). The rotation

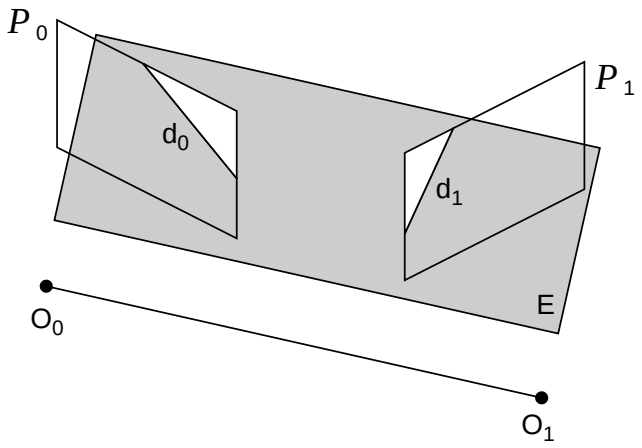


Figure 3.3: Intersection of images with a plane (from [17]).

$R_{\theta_i}^{d_i}$ of image plane $\mathcal{P}_i$ about the line $d_i = (d_{ix}, d_{iy}, d_{iz})$ so that the new epipole $\bar{e}_i$ has homogeneous coordinates $(\bar{e}_{ix}, \bar{e}_{iy}, 0)$ is given by

$$R_{\theta_i}^{d_i} = \begin{bmatrix} {d_{ix}}^2(1-\cos\theta_i)+\cos\theta_i & d_{ix}d_{iy}(1-\cos\theta_i) & d_{iy}\sin\theta_i \\ d_{ix}d_{iy}(1-\cos\theta_i) & {d_{iy}}^2(1-\cos\theta_i)+\cos\theta_i & -d_{ix}\sin\theta_i \\ -d_{iy}\sin\theta_i & d_{ix}\sin\theta_i & \cos\theta_i \end{bmatrix}$$

where

$$\theta_i = \tan^{-1}\left(\frac{1}{d_{iy}e_{ix} - d_{ix}e_{iy}}\right)$$

Seitz suggests choosing E implicitly by specifying $d_0 = (e_{0y}, -e_{0x}, 0)$; this choice of d_0 minimizes $|\theta_0|$. Then if $(x, y, z)^T = F(d_{0x}, d_{0y}, 0)^T$, it can be shown that $d_{1x} = \alpha y$ and $d_{1y} = -\alpha x$, where $\alpha = \sqrt{x^2 + y^2}^{-1}$.

An additional affine warp is required to align the conjugate epipolar lines. A rotation R_{ϕ_i} is first applied to each image to make the epipolar lines horizontal, i.e.

$$R_{\phi_i} = \begin{bmatrix} \cos\phi_i & -\sin\phi_i & 0 \\ \sin\phi_i & \cos\phi_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where

$$\phi_i = -\tan^{-1}\frac{\bar{e}_{iy}}{\bar{e}_{ix}}$$

Here, $\bar{e}_i$ are the epipoles in the parallel images $(R_{\theta_0}^{d_0}\mathcal{P}_0, R_{\theta_1}^{d_1}\mathcal{P}_1)$. After the ϕ rotations, the fundamental matrix of the transformed image plane pair $(R_{\phi_0}R_{\theta_0}^{d_0}\mathcal{P}_0, R_{\phi_1}R_{\theta_1}^{d_1}\mathcal{P}_1)$ has the form, up to a scale factor,

$$\bar{F} = R_{\phi_1}R_{\theta_1}^{d_1}FR_{-\theta_0}^{d_0}R_{-\phi_0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & 1 & b \end{bmatrix}$$

To bring the epipolar lines into alignment with those of the first image, i.e. to bring $\bar{F}$ to the form (3.8), the second image is vertically scaled and translated by a matrix T, given by

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -a & -b \\ 0 & 0 & 1 \end{bmatrix}$$

A pair of rectifying projective transformations that reduce F to the form in (3.8) are therefore given by:

$$\begin{aligned} G &= R_{\phi_0}R_{\theta_0}^{d_0} \\ H &= TR_{\phi_1}R_{\theta_1}^{d_1} \end{aligned}$$

It is desirable that the projective transformations computed by this method have positive diagonal elements, so that no reflections are involved in the transformations. The following algorithm can be applied:

1. Normalize $G(3,3) = H(3,3) = 1$ and check to see whether $a_{22}\bar{a}_{22} < 0$. If so, ϕ_1 should be incremented by π and H recomputed. Then $a_{22}\bar{a}_{22} > 0$.
2. At this point, the first and/or second rows of G and H can be multiplied by -1 if necessary to make the diagonal entries positive, without changing the relationship in (3.9).

Note that this method is an algorithm for selecting one pair of rectifying projective transformations from the entire family of rectifying projective transformation pairs, which has 7 degrees of freedom. There is no guarantee that the projective transformation pair that is estimated is optimal in any sense, and it can possibly distort the images in an undesirable way. However, a satisfactory method for selecting an optimal rectifying pair of projective transformations has not yet been proposed.

3.4 Estimating Point Correspondences

Many algorithms for image parameter estimation problems require a set of image correspondences as input. A common approach to obtaining such correspondences is to select a set of pixel regions (usually rectangular blocks) in $\mathcal{I}_0$ and to find matching regions in $\mathcal{I}_1$. The best match for a region of pixels $R_0 \in \mathcal{I}_0$ can be defined as the region $R_1 \in \mathcal{I}_1$ of the same size and shape as R_0 that minimizes the sum of squared intensity differences between pixels in the same position. In a sense, this is a very specific type of translation estimation problem.

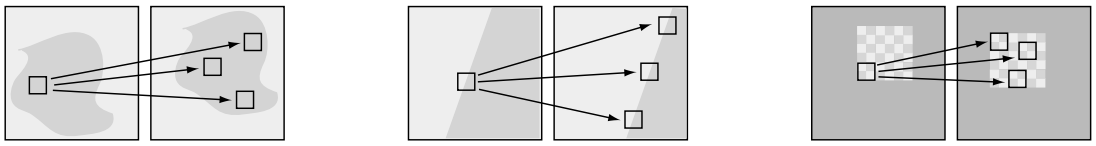


Figure 3.4: Regions that are difficult to match correctly.

However, some regions are worse than others for matching, in the sense that many minima to the matching problem may exist. Figure 3.4 illustrates blocks of pixels that lie in a constant intensity region, a region whose intensity along a linear profile is constant, and a regularly textured region. Each of the blocks in the left images can be matched with zero error by several blocks in the right images. Perturbations in intensity to these idealized image pairs create situations in which the global minimizer to an intensity-difference-minimizing functional is found in the bottom of a very shallow basin (i.e. the second derivative of the functional is very small at the minimizer).

To obtain robust results for estimation problems, it is important to obtain high-quality features in $\mathcal{I}_0$ that may be unambiguously matched in $\mathcal{I}_1$. Many indicators of pixel regions that constitute "good" features have been proposed, including line contours, corners, and junctions.

Often the definition of a "good" feature is made without reference to the estimation problem for which it is to be used. However, Tan et al. [20] presented an approach to finding good features for the feature correspondence problem by formulating it in a parameter estimation framework. After deriving the Cramér-Rao lower bound of an unbiased estimator for the parameter estimation problem, they were able to determine the types of features that minimized the variance of the estimator for various motion models (e.g. translation, scaling, rotation). They concluded that using features

with large intensity variations in both horizontal and vertical directions led to a low-variance estimator for the translation model. This is essentially the conclusion of one of a seminal series of papers by Tomasi and Kanade [21] on extracting scene shape and camera motion using feature correspondences in an image sequence.

When we refer to automatic feature detection and matching between two images $(\mathcal{I}_0, \mathcal{I}_1)$ in the text, we will use the following algorithm, proposed by Tan [22] and based on the Cramér-Rao lower-bound on the estimation of the translation of a block of pixels.

Algorithm 3.2: *Feature detection and matching.*

1. *Compute the gradients $S_x(x,y)$ and $S_y(x,y)$ for $\mathcal{I}_0$. That is,*

$$\begin{aligned} S_x(x,y) &= \mathcal{I}_0(x,y) - \mathcal{I}_0(x-1,y) \\ S_y(x,y) &= \mathcal{I}_0(x,y) - \mathcal{I}_0(x,y-1) \end{aligned}$$

2. *For every $M \times N$ block of pixels Γ,*
   (a) *Compute the covariance matrix*

$$I_\Gamma = \begin{bmatrix} \sum_{(x,y)\in\Gamma} S_x^2(x,y) & \sum_{(x,y)\in\Gamma} S_x S_y(x,y) \\ \sum_{(x,y)\in\Gamma} S_x S_y(x,y) & \sum_{(x,y)\in\Gamma} S_y^2(x,y) \end{bmatrix}$$

   (b) *Compute the feature quality measure*

$$q_\Gamma = \left(I^{-1}\right)_{11} + \left(I^{-1}\right)_{22}$$

   (c) *If q_Γ is less than some threshold τ, add Γ to the list of features.*
3. *Estimate the translation t between $\mathcal{I}_0$ and $\mathcal{I}_1$ (e.g. using one of the algorithms in Section 3.1.2).*
4. *For every block of pixels Γ in the list of features, find the $M \times N$ block of pixels in $\mathcal{I}_1$ that has the highest correlation. To save time, the search can be performed in a local neighborhood of the location of Γ in $\mathcal{I}_0$ translated by t.*

To counter problems with the deformation of features, Weng and Ahuja [23] analyzed feature attributes that are invariant to planar rigid motion. For their experiments, they used intensity and

definitions of "edgeness" and "cornerness" as descriptors for feature matching. Their algorithm was implemented using a coarse-to-fine methodology.

Block-matching correspondence approaches begin to fail when the motion of the camera or of scene objects induces too much of a change in the images. In this case, the assumption that a rectangular block of pixels in one image roughly matches a block of the same shape and size in the other breaks down. In the wide-baseline setting, where the neighborhood of a scene point can look very different from different points of view, this assumption may never be valid. We shall consider approaches to correspondence in this more difficult case in Chapter 5.

3.5 Review of 2-D Correspondence Algorithms

Here we review several classical approaches to the problem of establishing dense correspondence between an image pair. We call the algorithms in this section "2-D" because they make no use of the epipolar constraint. Hence, it is unlikely that the result of applying such an algorithm will be consistent with correspondence that could be obtained by a real imaging system. However, for many applications (e.g. video coding, computer graphics) this constraint is unimportant.

We first discuss the general problem of optical flow, a class of featureless methods that estimate a field of motion vectors for every pixel in an image pair. Next, we review layered motion algorithms, which can be viewed as an extension of parametrized motion models to regions of an image pair. Finally, we mention three algorithms from the computer graphics community based on interpolating a sparse set of feature correspondences. See [24, 25] for broad reviews of general correspondence techniques.

3.5.1 Optical Flow

A comprehensive discussion of optical flow computation is beyond the scope of this thesis; we only present a brief review here. Barron et al. [26] reviewed and compared the performance of many popular optical flow techniques, roughly categorized by differential techniques, region-based matching techniques, energy-based methods, and phase-based techniques. An older and broader reference is a survey by Aggarwal and Nandhakumar [27] of methods to compute motion from an

image sequence. The original reference for optical flow is Horn [28].

The motion field (or velocity field) M of $\mathcal{P}_0$ is a two-dimensional vector field. Let $(x, y) \in \mathcal{P}_0$ and $(x', y') \in \mathcal{P}_1$ be a visible correspondence. Then the motion vector $(u, v) = M(x, y)$ is defined as:

$$(u, v) = (x' - x, y' - y)$$

Due to occlusion, some points in the scene that are visible with respect to $\mathcal{C}_0$ are not visible with respect to $\mathcal{C}_1$. The motion field is not defined at such points of $\mathcal{P}_0$.

Optical flow is a somewhat poorly-defined term for a two-dimensional vector field on $\mathcal{P}_0$, generally described as the apparent or measurable motion of the intensity pattern from the image $\mathcal{I}_0$ to the image $\mathcal{I}_1$. One would like the optical flow field and the motion field to be identical, but even in ideal circumstances this is not the case. For example, any two images of an ideal Lambertian sphere that rotates under constant illumination are identical, implying zero optical flow, yet the motion field is non-zero. On the other hand, two images from the same camera of an ideal Lambertian sphere illuminated by a moving light source look different, implying non-zero optical flow, although the motion field is identically zero.

Optical flow methods are derived by assuming that the images $\mathcal{I}_0$ and $\mathcal{I}_1$ are slices of a function $g(x, y, t)$ for two nearby values of t. It is assumed that the intensity is conserved; that is,

$$g(x + \delta x, y + \delta y, t + \delta t) = g(x, y, t)$$

for some small δx, δy, and δt. Expanding $g(x, y, t)$ in a Taylor series about (x_0, y_0, t_0) gives

$$g(x_0 + \delta x, y_0 + \delta y, t_0 + \delta t) = g(x_0, y_0, t_0) + \delta x \frac{\partial g}{\partial x} + \delta y \frac{\partial g}{\partial y} + \delta t \frac{\partial g}{\partial t} + h.\ o.\ t. \tag{3.11}$$

Ignoring the higher order terms in (3.11), dividing through by δt and letting $\delta t \to 0$, we obtain the optical flow constraint

$$u \frac{\partial g}{\partial x} + v \frac{\partial g}{\partial y} + \frac{\partial g}{\partial t} = 0 \tag{3.12}$$

where

$$u = \frac{dx}{dt} \qquad v = \frac{dy}{dt}$$

are the components of the velocity associated with the point (x_0, y_0) in the x and y directions. The set of velocity vectors (u, v) evaluated at every point in $\mathcal{P}_0$ comprises the optical flow field. Optical

flow and correspondence are obviously related in that $(x, y) \in \mathcal{P}_0$ and $(x + u, y + v) \in \mathcal{P}_1$ are a corresponding pair. The partial derivatives of $g(x, y, t)$ are estimated from local derivatives of image intensity at each pixel.

The optical flow constraint (3.12) is one equation in two unknowns; hence, additional assumptions or constraints are required to uniquely specify the velocities u and v. Various typical assumptions include:

- Optical flow is smooth, and nearby pixels have similar velocities
- Optical flow is piecewise-constant
- Optical flow arises from a local or global motion model (e.g. translational, affine, or projective motion).

Optical flow techniques generally have difficulty resolving motions near depth discontinuities and occlusions, especially when smoothness constraints are imposed on the flow. The layered motion techniques addressed in the next section attempt to compensate for these problems. Also, many optical flow techniques make the implicit assumption that the motion between images is small (e.g. not more than a few pixels), which is generally not true for images taken by widely separated cameras, or even for video sequences generated by a briskly moving camera.

3.5.2 Layered Motion

The basic idea of layered motion is to posit the existence of L layers in the image, the motion of each of which is described by a parametric model. Then the formation of $\mathcal{I}_1$ is modeled by:

$$\mathcal{I}_1(w, \theta_k) = \mathcal{I}_0(w - m_k(w, \theta_k)) \quad \text{for } w \in \Gamma_k$$

where θ_k parameterizes the motion model m_k and Γ_k is the support set for the k^{th} model. It is generally assumed that $\{\Gamma_k, k = 1, \ldots, L\}$ is a nonoverlapping partition of the image, though sometimes this assumption is relaxed to allow for transparency. The parameters of the L models, as well as the number of models L, are to be estimated.

When almost all the pixels in the images move consistently with a single motion model (for example, in the case when only a few objects in the scene move independently of the camera),

this dominant motion can be estimated first, and subsequent layers and motion models estimated recursively. Sequential estimation is generally suboptimal, because a pixel may be irrevocably assigned to an incorrect layer in a preliminary stage. Sequential estimation methods also perform poorly when there is no dominant motion, or several strong motions.

Wang and Adelson [29] implemented a motion segmentation algorithm based on affine motion models for each layer:

$$m(w, \theta_1, \ldots, \theta_6) = \begin{bmatrix} \theta_1 + \theta_2 x + \theta_3 y \\ \theta_4 + \theta_5 x + \theta_6 y \end{bmatrix}$$

The algorithm alternates between two stages: hypothesis testing to assign pixels to one of a fixed set of motion layers, and a k-means clustering method to estimate the number of layers and the motion parameters for each layer. The algorithm terminates when only a few pixels are reassigned after each iteration. The output from a standard optical flow technique and a fixed set of non-overlapping layers are used to initialize the algorithm.

Hsu, Anandan, and Peleg [30] suggested applying optical flow methods to each parametrized motion layer to capture deviations in the modeled fit. The motion of the pixels is thus described by the parameters of a set of models, their regions of support, and a residual optical flow field.

Ayer and Sawhney [31] generalized the layered motion formulation to allow pixels to belong to different layers with some non-binary probability. The change of intensity of a pixel was modeled as an additive mixture of Gaussian densities. They used an expectation-maximization algorithm to obtain a maximum-likelihood estimate of the parameters of multiple models, their layers of support, and ownership probabilities, and a minimum-description-length principle to iteratively determine the appropriate number of models based on the space required to encode the model parameter values and motion residuals.

We note that an optical flow estimate is generally required by layered motion algorithms at a preliminary stage, and thus layered motion algorithms are difficult to apply in many of the same cases.

3.5.3 Structure from Motion

The correspondence problem is related to another classical computer vision problem of estimating structure from motion. Given a set of corresponding points and a pair of calibrated cameras, the 3-D locations of the points can be obtained simply by intersecting a pair of rays from the two cameras. Modeling assumptions can be used to construct a 3-D scene from the sparse set of 3-D features, which induces dense correspondence between the original image planes. This is a difficult problem; general reviews are given by Huang and Netravali [32] and Dhond and Aggarwal [33].

The canonical reference, using an orthographic camera, is Tomasi and Kanade [34], which was later extended to a para-perspective camera model [35]. Weng et al. [36] tested various nonlinear optimization algorithms for structure from motion using a perspective model, some of which included the epipolar constraint. We will go into more detail on epipolar-line-based methods in the next section.

3.5.4 Adaptive Meshes

Suppose we possess a set of point correspondences $\{w_j \mapsto w'_j \in \mathbb{R}^2, j = 1, \ldots, N\}$ that we wish to interpolate, without regard to parametric motion models. The feature points in each image can be connected to form the vertices of a set of triangles using an algorithm called Delaunay triangulation [37]. Points in the triangles' interiors are associated using trilinear interpolation. That is, if point $w \in \mathcal{P}_0$ lies within the triangle formed by $w_{j_1}, w_{j_2}, w_{j_3}$, then w may be uniquely written:

$$w = \alpha_1 w_{j_1} + \alpha_2 w_{j_2} + \alpha_3 w_{j_3} \quad \text{where } \sum_{i=1}^{3} \alpha_i = 1$$

This is known as writing w in barycentric coordinates. The correspondence w' in $\mathcal{P}_1$ of w is estimated to be

$$w = \alpha_1 w'_{j_1} + \alpha_2 w'_{j_2} + \alpha_3 w'_{j_3}$$

Generally this technique produces useful results only when the triangle mesh is very fine, and the triangles are chosen to coincide with roughly planar facets of the scene. One approach towards adaptively selecting a good mesh is described in [38].

3.5.5 Beier-Neely Morphing

Beier-Neely morphing [39] is a standard technique used in computer graphics to create special effects in which an object appears to continuously metamorphose from one shape and position to another. An intermediate step in the morphing process is the construction of a continuous, nonlinear mapping from the coordinate system of $\mathcal{P}_0$ to the coordinate system of $\mathcal{P}_1$.

In this method, the fixed correspondence data are directed line segments. The mapping from a pixel $w \in \mathcal{P}_0$ to $w' \in \mathcal{P}_1$ is computed by means of a parametrized weighted average of the distances from w to the control line segments in $\mathcal{P}_0$. Constructing a good morph is rarely automatic and typically requires some back-and-forth human interaction.

Since the weighting scheme depends on a choice of parameters and is relatively ad-hoc, it is very unlikely that any given choice of control lines and parameters will give rise to a physically valid correspondence. Morphing is more frequently used to generate a fine level of correspondence between different but similar objects (e.g. faces) than to estimate correspondence between views of the same scene. Seitz [40] combined morphing with the view interpolation technique described in Section 6.1 to create the effect of simultaneously and continuously interpolating between the shape and pose of two different objects.

3.6 References

[1] L. Brown. A Survey of Image Registration Techniques. *ACM Computing Surveys*, vol. 24, no. 4, pp. 325–376, 1992.

[2] V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley and Sons, 1984.

[3] R.D. Cook and S. Weisberg. *Residuals and Influence in Regression.* Chapman and Hall, 1982.

[4] D.M. Hawkins. *Identification of Outliers.* Chapman and Hall, 1980.

[5] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection.* John Wiley and Sons, 1987.

[6] F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel. *Robust Statistics: The Approach Based on Influence Functions.* John Wiley and Sons, 1986.

[7] V. Zagorodnov and P. Ramadge. Error Stabilization in Successive Estimation of Registration Parameters. In *Proc. ICIP 2000*, Vancouver, Canada, September 2000.

[8] C.D. Kuglin and D.C. Hines. The Phase Correlation Image Alignment Method. *Proc. Int. Conf. on Cybernetics and Society*, pp. 163–165, 1975.

[9] B.S. Reddy and B.N. Chatterji. An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration. *IEEE Transactions on Image Processing*, vol. 5, no. 8, pp. 1266–1271, August 1996.

[10] G. Wolberg. *Digital Image Warping.* IEEE Computer Society Press, 1990.

[11] O.D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, 1993.

[12] O.D. Faugeras, Q-T. Luong, and T. Papadopoulo. *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications.* MIT Press, 2001.

[13] Q.-T. Luong and O.D. Faugeras. The Fundamental Matrix: Theory, Algorithms, and Stability Analysis. *International Journal of Computer Vision*, Vol. 17, No. 1, pp. 43—76, 1996.

[14] R. Hartley. In Defence of the 8-Point Algorithm. *Proc. ICCV '95*, pp. 1064–1070, June 1995.

[15] G. Wolberg. *Digital Image Warping.* IEEE Computer Society Press, 1990.

[16] C.C. Slama, editor. *Manual of Photogrammetry*, 4th ed. American Society of Photogrammetry, Falls Church, VA, 1980.

[17] S.M. Seitz. *Image-Based Transformation of Viewpoint and Scene Appearance.* Ph.D. Thesis, Department of Computer Science, University of Wisconsin at Madison, 1997.

[18] R.I. Hartley. Theory and Practice of Projective Rectification. *International Journal of Computer Vision*, Vol. 35, No. 2, pp. 115–127, November 1999.

[19] F. Isgrò and E. Trucco. Projective Rectification without Epipolar Geometry. In *Proc. CVPR '99*, June 1999.

[20] Y.P. Tan, S. Kulkarni, and P. Ramadge. Extracting Good Features for Motion Estimation. *Proc. ICIP 1996*, vol. 1, pp. 117–120, 1996.

[21] C. Tomasi and T. Kanade. Detection and Tracking of Point Features (Shape and Motion from

Image Streams: a Factorization Method – Part 3). Carnegie Mellon University Department of Computer Science Technical Report CMU-CS-91-132, April 1991.

[22] Y.P. Tan. *Digital Video Analysis and Manipulation.* Ph.D. Thesis, Department of Electrical Engineering, Princeton University, November 1997.

[23] J. Weng, N. Ahuja, and T.S. Huang. Matching Two Perspective Views. *IEEE PAMI*, Vol. 14, No. 8, pp. 806–825, 1992.

[24] A. Redert, E. Hendriks, and J. Biemond. Correspondence Estimation in Image Pairs. *IEEE Signal Processing*, vol. 16, no. 3, pp. 29–46, May 1999.

[25] A.M. Tekalp. *Digital Video Processing.* Prentice Hall, 1995.

[26] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of Optical Flow Techniques. *International Journal of Computer Vision*, Vol. 12, No. 1, pp. 43–77, 1994.

[27] J.K. Aggarwal and N. Nandhakumar. On the Computation of Motion from Sequences of Images- A Review. *Proceedings of the IEEE*, Vol. 76, No. 8, pp. 917–935, August 1988.

[28] B.K.P. Horn. *Robot Vision.* MIT Press, 1986.

[29] J.Y.A. Wang and E.H. Adelson. Representing Moving Objects with Layers. *IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, Vol. 3, No. 5, pp. 625–638, September 1994.

[30] S. Hsu, P. Anandan, and S. Peleg. Accurate Computation of Optical Flow by Using Layered Motion Representations. *Proc. ICPR '94*, pp. 743–746, October 1994.

[31] S. Ayer and H.S. Sawhney. Layered Representation of Motion Video using Robust Maximum-Likelihood Estimation of Mixture Models and MDL Encoding. In *Proc. ICCV '95*, June 1995.

[32] T.S. Huang and A.N. Netravali. Motion and Structure from Feature Correspondences: A Review. *Proceedings of the IEEE*, Vol. 82, No. 2, pp. 251–268, February 1994.

[33] U.R. Dhond and J.K. Aggarwal. Structure from Stereo - A Review. *IEEE Trans. on Systems, Man, and Cybernetics.*, Vol. 19, No. 6, pp. 1489–1510, November 1989.

[34] C. Tomasi and T. Kanade. Shape and Motion from Image Streams under Orthography: a Factorization Method. *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[35] C. Poelman and T. Kanade. A Paraperspective Factorization Method for Shape and Motion Recovery. *Proc. ECCV '94*, Stockholm, vol. 2, pages 97–108, 1994.

[36] J. Weng, N. Ahuja, and T.S. Huang. Optimal Motion and Structure Estimation. *IEEE PAMI*, Vol. 15, No. 9, pp. 864–884, 1993.

[37] D.T. Lee and B.J. Schachter. Two Algorithms for Constructing a Delaunay Triangulation. *Int. J. Comput. Inform. Sci.*, Vol. 9, pp. 219–242, 1980.

[38] Y. Wang and O. Lee. Active Mesh—A Feature Seeking and Tracking Image Sequence Representation Scheme. *IEEE Trans. Image Processing*, vol. 3, pp. 610–624, Sept. 1994.

[39] T. Beier and S. Neely. Feature-Based Image Metamorphosis. *Computer Graphics (SIGGRAPH '92)*, pp. 35–42, July 1992.

[40] S.M. Seitz and C.R. Dyer. View Morphing. *Computer Graphics (SIGGRAPH '96)*, pp. 21–30, August 1996.

Chapter 4

Projective Transformations

The estimation of the parameters of a two-dimensional projective transformation is a standard problem that arises in image and video processing. A projective transformation maps a point $w \in \mathbb{R}^2$ to $w' \in \mathbb{R}^2$ by:

$$w' = \frac{Aw + b}{c^T w + d} \tag{4.1}$$

where $A \in \mathbb{R}^{2\times 2}$, $b, c \in \mathbb{R}^2$, and $d \in \mathbb{R}$. An affine transformation is a special case of a projective transformation.

One typical application is the recovery of a global motion model for points in images of a stationary scene taken by a rotating and zooming camera [1]. The motion model can be used to synthesize panoramic image mosaics [2, 3, 4, 5, 6]. A second application is the registration of images of a planar surface taken by multiple separated cameras [7, 8]. These effects are illustrated in Section 4.1.

As in Section 3.1.1, we can pose the projective transformation estimation problem as a least squares minimization based on a finite set of noisy point samples of the underlying transformation. However, in contrast to the affine case, this generally results in an eight-dimensional nonquadratic minimization problem. Such a problem is typically solved numerically using an 'off-the-shelf' procedure such as the Gauss-Newton or Levenberg-Marquardt algorithm [9].

Within this context, we show in Section 4.2 that the general least squares problem for estimating a projective transformation can be analytically reduced to a two-dimensional nonquadratic minimization problem. Some properties of the two-dimensional cost function are discussed in Section 4.4. In Sections 4.5 and 4.6 we discuss issues involved with the practical minimization of the

cost function by analyzing its gradient and Hessian, and show that any descent algorithm for the eight-dimensional problem can be modified to produce a more effective descent algorithm for the two-dimensional problem. Of course, we are also concerned with real implementations of the minimizations on a computer, and we provide experimental results in Section 4.7 to show that Newton methods based on the two-dimensional problem outperform analogous methods applied to the eight-dimensional problem. Furthermore, we propose an approximate second-derivative method that is quite robust to measurement noise. A brief summary of some of our results originally appeared in [10].

Though here we concentrate exclusively on the minimization of the nonlinear least-squares cost functional introduced in Section 4.2, other methods for approaching the projective transformation estimation problem exist. Tan [11] introduced an approximation to make the least-squares problem linear, which is valid when the c parameters are very close to 0. Kanatani [1] proposed a tensor-based approximation that reduces the estimation problem to an eigendecomposition. However, the mapping (4.1) is very sensitive to changes in the c parameters, and as these parameters deviate from 0, the above approximations quickly diverge from the solution to the nonlinear problem.

Instead of using a set of point correspondences as a basis for estimating a projective transformation, Mann and Picard [2] proposed an iterative technique for simultaneously estimating the transformation parameters and optical flow over an entire image pair. However, they used bilinear approximations to the projective transformations in each step in order to simplify the estimation.

4.1 Origins of Projective Transformations

For $M \in \mathbb{R}^{3\times 3}$ with $\det(M) \neq 0$, i.e., $M \in GL(3)$, write

$$M = \begin{pmatrix} A & b \\ c^T & d \end{pmatrix}$$

with $A \in \mathbb{R}^{2\times 2}$, b and $c \in \mathbb{R}^{2\times 1}$, and $d \in \mathbb{R}$. Then the transformation g_M of the plane defined by

$$g_M(w) = \frac{Aw + b}{c^T w + d} \tag{4.2}$$

is called a projective transformation[1] with homogeneous coordinates M.

We state without proof the following well-known properties of projective transformations.

Proposition 4.1: *The family of projective transformations of the plane has the following properties:*

1. *The composition of $g_M \cdot g_N$ is the projective transformation g_{MN}.*
2. *The identity transformation of the plane is the projective transformation g_I, where I is the identity matrix in $\mathbb{R}^3$.*
3. *g_M has the inverse projective transformation $(g_M)^{-1} = g_{M^{-1}}$.*
4. *The homogeneous coordinates of g_M are unique to within a scalar multiple.*

We see from the above that the set of projective transformations of the plane forms a group $\mathcal{G}$ under function composition. In the remainder of the development, we will normalize $d = 1$, so that a projective transformation is uniquely characterized by eight parameters $M = (A, b, c)$. This excludes the set of transformations with $d = 0$. However, this subset of transformations is not usually of interest. We note that the set of affine transformations $\mathcal{A}$ is a subgroup of $\mathcal{G}$. The two "projective" parameters of c account for the keystoning effects of perspective projection (see Figure 4.1).

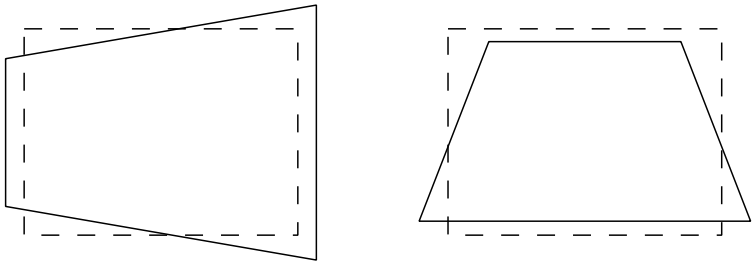


Figure 4.1: Perspective effects. (a) Effect of varying c_1, (b) Effect of varying c_2.

Again, we pause to determine when the assumption that two images are related by a projective transformation is well-founded. Reconsider equations (2.4) and (2.5) from Chapter 2 that relate the

[1] A projective transformation is sometimes also known as a collineation or a homography.

image coordinates of a point seen by two cameras $\mathcal{C}_0$ and $\mathcal{C}_1$:

$$x' = \frac{r_{11}\frac{f_1}{f_0}x + r_{12}\frac{f_1}{f_0}y + r_{13}f_1 + \frac{t_X f_1}{Z}}{\frac{r_{31}}{f_0}x + \frac{r_{32}}{f_0}y + r_{33} + \frac{t_Z}{Z}} \tag{4.3}$$

$$y' = \frac{r_{21}\frac{f_1}{f_0}x + r_{22}\frac{f_1}{f_0}y + r_{23}f_1 + \frac{t_Y f_1}{Z}}{\frac{r_{31}}{f_0}x + \frac{r_{32}}{f_0}y + r_{33} + \frac{t_Z}{Z}} \tag{4.4}$$

For this to be a projective transformation that globally relates the image coordinates, for every scene point (X, Y, Z) we require:

$$\begin{aligned} \frac{t_X}{Z} &= \alpha_1 x + \beta_1 y + \gamma_1 \\ \frac{t_Y}{Z} &= \alpha_2 x + \beta_2 y + \gamma_2 \\ \frac{t_Z}{Z} &= \alpha_3 x + \beta_3 y + \gamma_3 \end{aligned}$$

for some constant scalars $\alpha_i, \beta_i, \gamma_i$. These conditions are satisfied when either:

1. $t_X = t_Y = t_Z = 0$ or
2. $k_1 X + k_2 Y + k_3 Z = 1$

In the first case, corresponding to a camera whose optical center undergoes no translation, we obtain

$$\begin{aligned} x' &= \frac{r_{11}\frac{f_1}{f_0}x + r_{12}\frac{f_1}{f_0}y + r_{13}f_1}{\frac{r_{31}}{f_0}x + \frac{r_{32}}{f_0}y + r_{33}} \\ y' &= \frac{r_{21}\frac{f_1}{f_0}x + r_{22}\frac{f_1}{f_0}y + r_{23}f_1}{\frac{r_{31}}{f_0}x + \frac{r_{32}}{f_0}y + r_{33}} \end{aligned}$$

An example of three such images composed into the same frame of reference with appropriate projective transformations is illustrated in Figure 4.2. Note the nonlinear warping of the images.

In the second case, corresponding to a planar scene, (4.3)-(4.4) become:

$$\begin{aligned} x' &= \frac{(r_{11}\frac{f_1}{f_0} + t_X f_1 k_1)x + (r_{12}\frac{f_1}{f_0} + t_X f_1 k_2)y + (r_{13}f_1 + t_X f_1 k_3)}{(\frac{r_{31}}{f_0} + t_Z k_1)x + (\frac{r_{32}}{f_0} + t_Z k_2)y + (r_{33} + t_Z k_3)} \\ y' &= \frac{(r_{21}\frac{f_1}{f_0} + t_Y f_1 k_1)x + (r_{22}\frac{f_1}{f_0} + t_Y f_1 k_2)y + (r_{23}f_1 + t_Y f_1 k_3)}{(\frac{r_{31}}{f_0} + t_Z k_1)x + (\frac{r_{32}}{f_0} + t_Z k_2)y + (r_{33} + t_Z k_3)} \end{aligned}$$

Figure 4.2: Images from a non-translating camera.

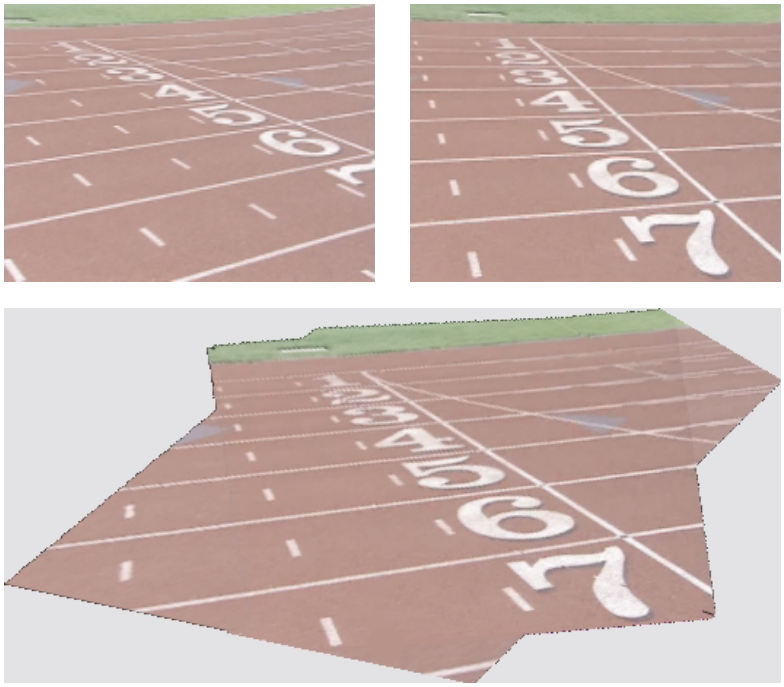


Figure 4.3: Images of a planar scene.

An example of a pair of images of a planar surface, registered by an appropriate projective transformation, is illustrated in Figure 4.3.

Note that g_M in (4.2) is defined at all points of $\mathbb{R}^2$ except those on the line $c^T w + 1 = 0$, which is called the singular line of the transformation g_M. Along this line $Aw + b \neq 0$, since the matrix $M \in GL(3)$.

In the two cases above, singular lines have a geometric interpretation. The singular line is simply the intersection of the image plane $\mathcal{P}_0$ with the plane $Z' = 0$ corresponding to the parallel transport of the image plane $\mathcal{P}_1$ to the center of projection O_1. This is illustrated in Figure 4.4. In practical situations (e.g. when the image planes are of finite extent, and one camera is not in the field of view of the other) all the points in $\mathcal{P}_0$ lie to one side of the singular line.

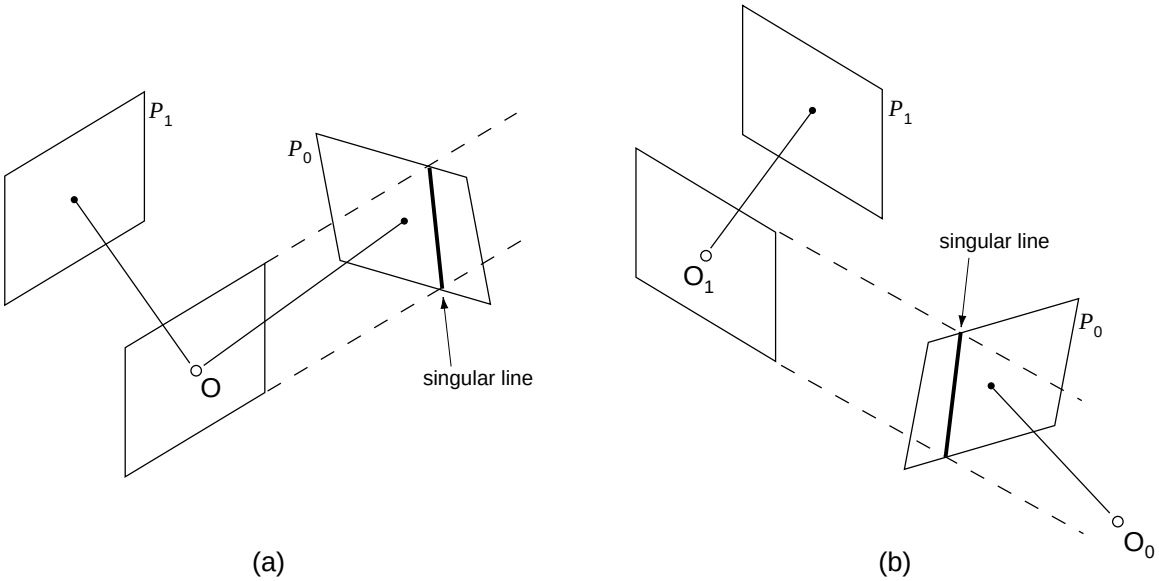


Figure 4.4: (a) Singular line in fixed-center case, (b) Singular line in translated case.

For a fixed projective transformation, hence for a fixed c, there is a line of w in $\mathcal{P}_0$ that lie on the corresponding singular line. Conversely, for a fixed $w \in \mathcal{P}_0$, there is a singular line of c in $\mathbb{R}^2$.

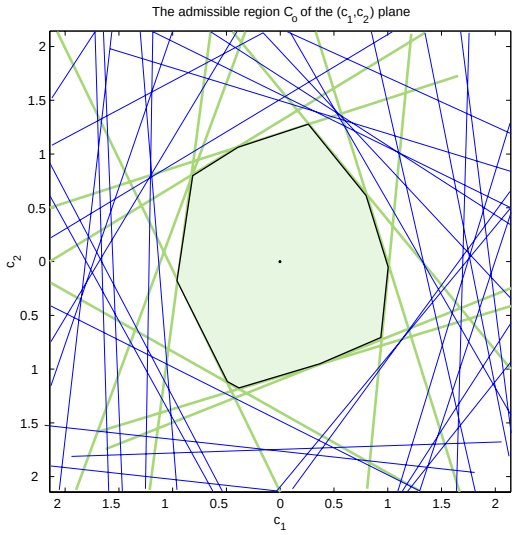


Figure 4.5: The admissible region C_o of the (c_1, c_2) plane generated by data points from actual images. Thin lines represent singular lines; thick lines are singular lines that actively bound the admissible region.

4.2 The Least Squares Estimate

Our objective is to select the parameters $M = (A, b, c)$ so that g_M best fits a given set of point matches:

$$\{w_j \mapsto w'_j \in \mathbb{R}^2, j = 1, \ldots, N\}$$

A case of special interest arises when the data consists of noisy samples of a fixed but unknown projective transformation g_{M^*}:

$$w'_j = g_{M^*}(w_j) + e_j, \; j = 1 \ldots N$$

Here $e_j \in \mathbb{R}^2$ is the error in the measurement of $g_{M^*}(w_j)$. In this case we seek an estimate M of M^*. As discussed in Section 3.4, in practice, the noisy point samples originate from automatically generated or manually selected feature correspondences in an image pair such as similar blocks of pixels, intersections of lines, or corners.

An estimate M is, by definition, *admissible* if the singular line of g_M does not intersect the convex hull W of 0 and w_j, $j = 1, \ldots, N$. Since $0 \in W$, M is admissible if and only if $c^T w + 1 > 0$ for all $w \in W$. This is equivalent to the requirement that $c^T w_j + 1 > 0$, $j = 1, \ldots, N$. This defines

an open convex set $C_o \subset \mathbb{R}^2$ of allowed values for c, and M is admissible if and only if $c \in C_o$. The set of admissible estimates is the open set $\{(A, b, c)\colon A \in \mathbb{R}^{2\times 2}, b \in \mathbb{R}^2, c \in C_o\}$. Note that admissibility does not require $M \in GL(3)$. Figure 4.5 illustrates the admissible region C_o generated by data points from an actual image pair.

The least squares estimate $\hat{M} = (\hat{A}, \hat{b}, \hat{c})$ consists of those values of A, b and c that globally minimize:

$$Q(M) = \frac{1}{2}\sum_{j=1}^{N}\left(w_j' - \frac{Aw_j + b}{c^T w_j + 1}\right)^T \left(w_j' - \frac{Aw_j + b}{c^T w_j + 1}\right) \tag{4.5}$$

over all admissible $M = (A, b, c)$. In general this estimate need not be an element of $GL(3)$ and hence need not itself be a projective transformation. However, for a wide range of reasonable models for the noise terms e_j, $j = 1, \ldots N$, $\hat{M}$ will generically be an element of the open set $GL(3)$. We defer the proof that Q has a global minimum within the set of admissible estimates to the end of Section 4.5.

For a fixed data set, obtaining the least squares estimate requires solving a nonlinear minimization problem over an open subset of an 8-dimensional Euclidean space. However, as Theorem 4.1 below shows, the solution can be obtained by solving a nonlinear minimization problem over an open convex subset of $\mathbb{R}^2$.

Theorem 4.1: *Assuming that the points w_j, $j = 1, \ldots, N$ are not colinear, the least squares estimate $\hat{M}$ has the form $(A(\hat{c}), b(\hat{c}), \hat{c})$ and thus lies on the 2-dimensional submanifold $\mathcal{M} \triangleq \{(A, b, c)\colon A = A(c), b = b(c), c \in C_o\}$ of the eight dimensional space $\mathbb{R}^{2\times 2} \times \mathbb{R}^2 \times C_o$.*

Proof: Since $\hat{M}$ minimizes (4.5), it follows that we must have $D_A Q(\hat{M}) = 0$, $D_b Q(\hat{M}) = 0$, and $D_c Q(\hat{M}) = 0$. This yields the normal equations:

$$\hat{A}\sum_j \frac{w_j w_j^T}{(\hat{c}^T w_j + 1)^2} + \hat{b}\sum_j \frac{w_j^T}{(\hat{c}^T w_j + 1)^2} - \sum_j \frac{w_j' w_j^T}{\hat{c}^T w_j + 1} = 0 \tag{4.6}$$

$$\hat{A}\sum_j \frac{w_j}{(\hat{c}^T w_j + 1)^2} + \hat{b}\sum_j \frac{1}{(\hat{c}^T w_j + 1)^2} - \sum_j \frac{w_j'}{\hat{c}^T w_j + 1} = 0 \tag{4.7}$$

$$\sum_j \left(w_j' - \left(\frac{\hat{A}w_j + \hat{b}}{\hat{c}^T w_j + 1}\right)\right)^T \left(\frac{\hat{A}w_j + \hat{b}}{\hat{c}^T w_j + 1}\right)\frac{w_j}{\hat{c}^T w_j + 1} = 0 \tag{4.8}$$

We can rewrite (4.6) and (4.7) as a linear system:

$$[\ \hat{A} \quad \hat{b}\ ]W(\hat{c}) = V(\hat{c}) \tag{4.9}$$

where $W(c) \in \mathbb{R}^{3\times 3}$, $V(c) \in \mathbb{R}^{2\times 3}$ are functions of $c \in \mathbb{R}^2$ and the data points, given by:

$$W(c) = \begin{bmatrix} \sum_{j=1}^N \frac{w_j w_j^T}{q_j^2(c)} & \sum_{j=1}^N \frac{w_j}{q_j^2(c)} \\ \\ \sum_{j=1}^N \frac{w_j^T}{q_j^2(c)} & \sum_{j=1}^N \frac{1}{q_j^2(c)} \end{bmatrix} \tag{4.10}$$

(4.11)

$$V(c) = \begin{bmatrix} \sum_{j=1}^N \frac{w_j' w_j^T}{q_j(c)} & \sum_{j=1}^N \frac{w_j'}{q_j(c)} \end{bmatrix} \tag{4.12}$$

Here $q_j(c) = c^T w_j + 1$. Therefore, defining

$$[\ A(c) \quad b(c)\ ] = V(c)W^{-1}(c) \tag{4.13}$$

we have $(\hat{A}, \hat{b}, \hat{c}) = (A(\hat{c}), b(\hat{c}), \hat{c})$ and the theorem follows. ∎

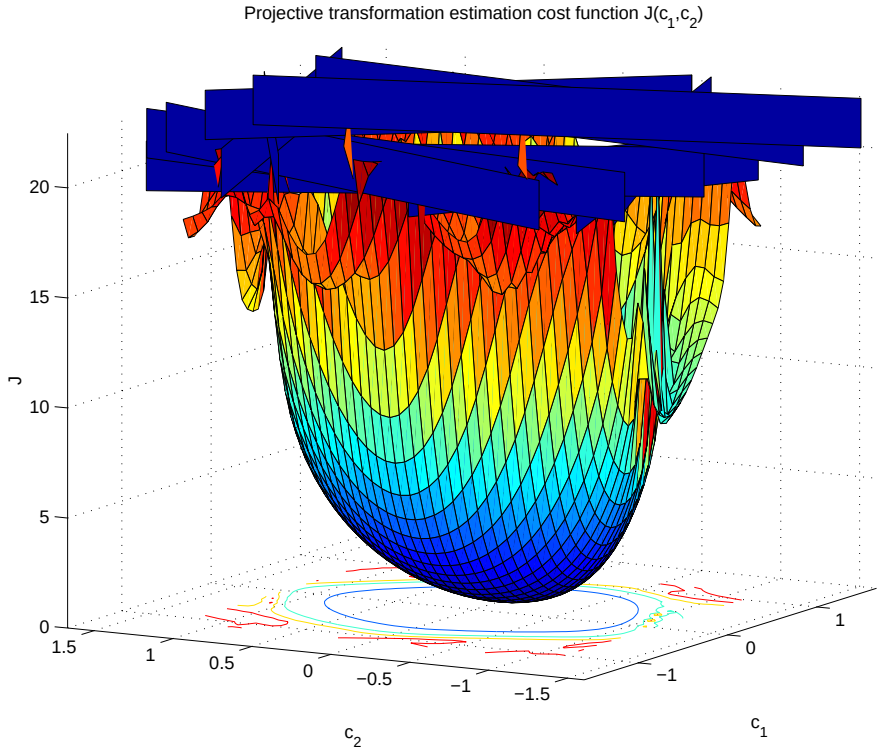
We make a standing assumption that the points $\{w_j : j = 1, \ldots, N\}$ are not colinear in $\mathbb{R}^2$. This ensures that $W(c)$ is positive definite and hence that $A(c)$ and $b(c)$ are defined for all $c \in C_o$.

In view of Theorem 4.1, we can define a two-dimensional cost functional $J : C_o \to \mathbb{R}$ by

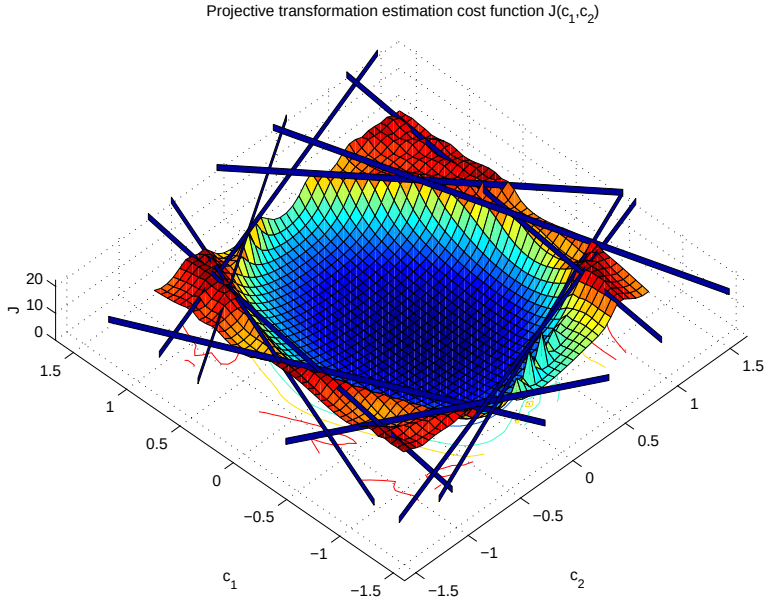
$$J(c) = \frac{1}{2}\sum_{j=1}^N \left(w_j' - \frac{A(c)w_j + b(c)}{c^T w_j + 1}\right)^T \left(w_j' - \frac{A(c)w_j + b(c)}{c^T w_j + 1}\right) \tag{4.14}$$

$J(c)$ is simply the least squares cost function restricted to the manifold $\mathcal{M}$. By construction, for any $M_o = (A(c_o), b(c_o), c_o) \in \mathcal{M}$, $Q(M_o) = J(c_o)$. Hence the global minimizing solution of $J(c)$ within C_o is $\hat{c}$. This reduces the determination of the least squares estimate $\hat{M}$ to the minimization of J over C_o.

From the proof of the theorem, we can see that the 8-dimensional minimization of $Q(M)$ decouples into a nonlinear 2-dimensional minimization of c and a solution of a linear system for the "affine" parameters (A, b). This agrees nicely with the affine solution (3.2) of Section 3.1.1, which is in fact (4.13) with $c = 0$. The problem considered in this chapter can be viewed as a specific case of a general mixed least-squares problem that separates into linear and nonlinear variables. Golub



(a)



(b)

Figure 4.6: Two views of the cost function $J(c_1, c_2)$ for data points from actual images. The dark lines are the singular lines that actively bound C_o.

and Pereyra [12] studied such problems and discussed their minimization. We go into considerably more detail here, exploring the structure of our specific problem.

Casting the problem in a two-dimensional setting allows us to visualize the cost function and the steps that a minimization algorithm takes. We shall show in Section 4.5 that in addition to being of reduced dimensionality, the cost function $J(c)$ can be numerically minimized more efficiently than the cost function $Q(M)$.

Figure 4.6 illustrates two views of the cost function J graphed over the region C_o for data points from a pair of natural images. For this example, the cost function J has a single minimum within C_o, located at the bottom of a deep bowl.

4.3 Data Normalization

To avoid numerical instabilities introduced by data measurements that vary by orders of magnitude, it is generally wise to normalize the data before processing it. Hence, we need to understand how the solution to the least-squares problem using the normalized data is related to the solution of the problem in the original coordinates. To this end, we present the following lemma, which is easily proven.

Lemma 4.1: *Consider the data sets given by*

$$\begin{aligned} z_j &= Qw_j + t \\ z'_j &= Rw'_j + t' \end{aligned}$$

for $t, t' \in \mathbb{R}^2$, $Q, R \in GL(2)$, *and* $j = 1, \ldots, N$. *If* $\hat{M} = (\hat{A}, \hat{b}, \hat{c})$ *is the minimizer of*

$$Q(M) = \frac{1}{2} \sum_{j=1}^{N} \left(w'_j - \frac{Aw_j + b}{c^T w_j + 1} \right)^T \left(w'_j - \frac{Aw_j + b}{c^T w_j + 1} \right)$$

then the minimizer $\tilde{M} = (\tilde{A}, \tilde{b}, \tilde{c})$ *of*

$$\tilde{Q}(M) = \frac{1}{2} \sum_{j=1}^{N} \left(z'_j - \frac{Az_j + b}{c^T z_j + 1} \right)^T \left(z'_j - \frac{Az_j + b}{c^T z_j + 1} \right)$$

is given by

$$(\tilde{A}, \tilde{b}, \tilde{c}) = \left(\frac{R\hat{A}Q^{-1} + t'\hat{c}^T Q^{-1}}{1 - \hat{c}^T Q^{-1} t}, \frac{R\hat{b} - R\hat{A}Q^{-1}t}{1 - \hat{c}^T Q^{-1} t}, \frac{Q^{-T}c}{1 - \hat{c}^T Q^{-1} t} \right)$$

In other words,

$$(\hat{A}, \hat{b}, \hat{c}) = \left(\frac{R^{-1}(\tilde{A} - t'\tilde{c}^T)Q}{1 + \tilde{c}^T t}, \frac{R^{-1}(\tilde{b} + \tilde{A}t)}{1 + \tilde{c}^T t} - R^{-1}t', \frac{Q^T \tilde{c}}{1 + \tilde{c}^T t} \right) \tag{4.15}$$

In practice, we normalize the data so that the measurements are zero mean with range approximately $[-1, 1]$. This corresponds to a choice of

$$\begin{aligned} t &= -\tfrac{\mu}{\alpha} \quad t' = -\tfrac{\mu'}{\alpha'} \\ Q &= \tfrac{1}{\alpha} I \quad R = \tfrac{1}{\alpha'} I \end{aligned}$$

where

$$\begin{aligned} \mu &= \tfrac{1}{N} \textstyle\sum_{j=1}^{N} w_j & \mu' &= \tfrac{1}{N} \textstyle\sum_{j=1}^{N} w'_j \\ \alpha &= \tfrac{1}{2} \left(\max_j |x_j - \mu_x| + \max_j |y_j - \mu_y| \right) & \alpha' &= \tfrac{1}{2} \left(\max_j |x'_j - \mu'_x| + \max_j |y'_j - \mu'_y| \right) \end{aligned}$$

For this choice of (t, t', Q, R), we can rewrite (4.15) as

$$\begin{aligned} \hat{A} &= \frac{\frac{1}{\alpha}\left(\alpha' \tilde{A} + \mu' \tilde{c}\right)}{1 - \frac{\mu}{\alpha}\tilde{c}} \\ \hat{b} &= \frac{\alpha'(\tilde{b} - \tilde{A}\frac{\mu}{\alpha})}{1 - \frac{\mu}{\alpha}\tilde{c}} + \mu' \\ \hat{c} &= \frac{\tilde{c}}{\alpha - \mu\tilde{c}} \end{aligned}$$

4.4 The Behavior of J on Singular Lines

For c^* on one or more singular lines, the matrices $W(c^*)$ and $V(c^*)$ that define $(A(c^*), b(c^*))$ in (4.13) are not defined. However, below we provide two results concerning the finiteness and continuity of the functions $A(c)$ and $b(c)$ as c approaches a singular line. In the first theorem, we consider the behavior as we approach a point that lies on exactly one singular line.

Theorem 4.2: *Fix c^* such that $c^{*T} w_1 + 1 = 0$, and $c^{*T} w_j + 1 \neq 0$ for $j \neq 1$. Define $c(\alpha) = c^* + \alpha h$, where h is an approach vector in $\mathbb{R}^2$. Then $q_1(c(\alpha)) = \alpha h^T w_1 \neq 0$ when $\alpha \neq 0$ and $h^T w_1 \neq 0$*

(i.e. the approach direction is not along the singular line). We henceforth assume that $h^T w_1$ *is normalized to 1, so that*[2] $q_1(c(\alpha)) = \alpha$. *By solving (4.13) with* $c(\alpha)$, *we naturally define* $A(\alpha)$ *and* $b(\alpha)$.

1. *The limiting values*

$$\begin{bmatrix} A_o & b_o \end{bmatrix} = \lim_{\alpha \to 0} \begin{bmatrix} A(\alpha) & b(\alpha) \end{bmatrix}$$

   are well-defined and finite.

2. (A_o, b_o) *is the solution to the well-defined constrained least-squares problem*

$$\begin{aligned} \min_{A,b} \quad & \frac{1}{2} \sum_{j=2}^{N} \left(w_j' - \frac{Aw_j + b}{c^{*T} w_j + 1} \right)^T \left(w_j' - \frac{Aw_j + b}{c^{*T} w_j + 1} \right) \\ s.t. \quad & Aw_1 + b = 0 \end{aligned}$$

3. *The limiting value of* $\frac{A(c)w_1 + b(c)}{c^T w_1 + 1}$ *as* c *approaches* c^* *is* w_1'.

Proof: The proof can be found in Appendix A. ∎

This result shows that, unlike the cost function $Q(M)$, the cost function $J(c)$ is finite and continuous along the singular lines. However, along singular lines the resulting least-squares projective transformation estimates are not members of $GL(3)$. The second and third parts of the theorem give some additional intuition as to how A_o and b_o are converging. Not only are they selected to keep the cost function finite, but they act to zero out the offending data point's contribution to the cost function.

It is also important to consider the limiting behavior of $(A(c), b(c))$ as c approaches an intersection of two singular lines. To this end, we state the following theorem without proof; the omitted proof is straightforward but tedious, and follows the same pattern as the proof of Theorem 4.2.

Theorem 4.3: *Suppose* $c^{*T} w_1 + 1 = 0$ *and* $c^{*T} w_2 + 1 = 0$, *with* $c^{*T} w_j + 1 \neq 0$ *for* $j > 2$. *Define* $c(\alpha) = c^{*T} + \alpha h$, *where* h *is an approach vector in* $\mathbb{R}^2$; *this defines* $A(\alpha)$ *and* $b(\alpha)$ *through (4.13). Abbreviate* $p = [w_1^T \ 1]^T$ *and* $q = [w_2^T \ 1]^T$. *We assume these points are distinct.*

[2] $h^T w_1 = 1$ *is just a line parallel to the singular line. As we decrease* α, *we approach the singular point* c^* *along lines parallel to* $c^{*T} w_1 + 1 = 0$.

1. *The limiting values as c approaches the intersection of singular lines is well-defined and finite:*

$$\begin{aligned}
\left[\begin{array}{cc} A_o & b_o \end{array}\right] &= \lim_{\alpha\to 0}\left[\begin{array}{cc} A(\alpha) & b(\alpha) \end{array}\right] \\
&= V_3 W_3^{-1}\left[I - \frac{[(q^T W_3^{-1} q)p - (q^T W_3^{-1} p)q]p^T W_3^{-1}}{(p^T W_3^{-1} p)(q^T W_3^{-1} q) - (q^T W_3^{-1} p)^2}\right. \\
&\quad \left. + \frac{[(p^T W_3^{-1} p)q - (p^T W_3^{-1} q)p]q^T W_3^{-1}}{(p^T W_3^{-1} p)(q^T W_3^{-1} q) - (q^T W_3^{-1} p)^2}\right]
\end{aligned}$$

where

$$\begin{aligned}
W_3 &= \left[\begin{array}{cc} \sum_{j=3}^N \frac{w_j w_j^T}{q_j^2(c^*)} & \sum_{j=3}^N \frac{w_j}{q_j^2(c^*)} \\ \sum_{j=3}^N \frac{w_j^T}{q_j^2(c^*)} & \sum_{j=3}^N \frac{1}{q_j^2(c^*)} \end{array}\right] \\
V_3 &= \left[\begin{array}{cc} \sum_{j=3}^N \frac{w_j' w_j^T}{q_j(c^*)} & \sum_{j=3}^N \frac{w_j'}{q_j(c^*)} \end{array}\right]
\end{aligned}$$

2. *The expression above is equivalent to the solution of the constrained minimization problem over $N-2$ data points:*

$$\begin{aligned}
\min_{A,b} \quad & \frac{1}{2}\sum_{j=3}^N \left(w_j' - \frac{Aw_j + b}{c^{*T} w_j + 1}\right)^T \left(w_j' - \frac{Aw_j + b}{c^{*T} w_j + 1}\right) \\
s.t. \quad & Aw_1 + b = 0 \qquad (4.16) \\
& Aw_2 + b = 0 \qquad (4.17)
\end{aligned}$$

The corresponding Lagrange multipliers for (4.16) and (4.17) respectively are:

$$\begin{aligned}
\lambda &= V_3 W_3^{-1} \frac{(q^T W_3^{-1} q)p - (q^T W_3^{-1} p)q}{(p^T W_3^{-1} p)(q^T W_3^{-1} q) - (q^T W_3^{-1} p)^2} \\
\mu &= V_3 W_3^{-1} \frac{(p^T W_3^{-1} p)q - (p^T W_3^{-1} q)p}{(p^T W_3^{-1} p)(q^T W_3^{-1} q) - (q^T W_3^{-1} p)^2}
\end{aligned}$$

3. *The limiting values satisfy*

$$\begin{aligned}
\lim_{\alpha\to 0} \frac{A(\alpha)w_1 + b(\alpha)}{(c^* + \alpha h)^T w_1 + 1} &= w_1' \\
\lim_{\alpha\to 0} \frac{A(\alpha)w_2 + b(\alpha)}{(c^* + \alpha h)^T w_2 + 1} &= w_2'
\end{aligned}$$

The intersection of three singular lines requires that three data points be colinear, which is generally not the case. However, in such an event one can prove a corresponding result on the finiteness and continuity of $A(c)$ and $b(c)$, and so on.

4.5 Line-Search Descent

Typical algorithms for the minimization of a nonlinear function such as (4.5) operate in an iterative fashion as follows. Given a current approximation M_k of $\hat{M}$ select a direction d_k and search along the line from M_k in the direction d_k for the minimum of the objective function. The next approximation M_{k+1} is the value of M at this minimum. Typically the direction d_k is related to the gradient of the objective function evaluated at M_k.

Specifically, we consider a line-descent-based approach for minimizing $Q(M)$. Let $M_k = (A_k, b_k, c_k)$, $k \geq 0$, be the approximation of $\hat{M}$ after step k and let $d_k = (F_k, g_k, h_k)$ denote the search direction used at step k. Then

$$(A_{k+1}, b_{k+1}, c_{k+1}) = (A_k, b_k, c_k) + \alpha_k(F_k, g_k, h_k)$$

where the step size $\alpha_k \geq 0$ is selected to ensure that $Q(M_{k+1}) \leq Q(M_k)$.

For all such schemes we can make several observations. Let $M_o = (A_o, b_o, c_o)$ with $A_o \in \mathbb{R}^{2\times 2}$, and $b_o, c_o \in \mathbb{R}^2$. Define the projection of M_o onto $\mathcal{M}$ to be $P(M_o) \triangleq (A(c_o), b(c_o), c_o)$.

Theorem 4.4: *Let $d = (F, g, h)$ with $F \in \mathbb{R}^{2\times 2}$, and $g, h \in \mathbb{R}^2$. Then*

1. *For any M_o, $J(c_o) = Q(P(M_o)) \leq Q(M_o)$.*
2. *For M_o on $\mathcal{M}$, define*

$$\begin{aligned} M(\alpha) &= M_o + \alpha d \\ c(\beta) &= c_o + \beta h \\ \alpha^* &= \operatorname{argmin}_{\alpha\geq 0} Q(M(\alpha)) \\ \beta^* &= \operatorname{argmin}_{\beta\geq 0} J(c(\beta)) \\ M_{\beta^*} &= (A(c(\beta^*)), b(c(\beta^*)), c(\beta^*)) \end{aligned}$$

   Then $Q(M_{\beta^*}) = J(c(\beta^*))$, and $Q(M_{\beta^*}) \leq Q(P(M(\alpha^*))) \leq Q(M(\alpha^*)) \leq Q(M_o)$.

3. *For M_o on $\mathcal{M}$, if $d = (F, g, h)$ is a descent direction for Q at M_o, then h is a descent direction for J at c_o.*

Proof:

1. Consider minimizing $Q(M)$ with M constrained so that $c = c_o$. The normal equations for this problem are linear and have the unique solution $A(c_o)$ and $b(c_o)$. Hence on the constraint set $c = c_o$, $Q(M)$ has a unique global minimum at the point $(A(c_o), b(c_o), c_o) = P(M_o)$. Since M_o lies in this set, $Q(P(M_o)) \leq Q(M_o)$.

2. For $\beta \geq 0$, $M_\beta = (A(c(\beta)), b(c(\beta)), c(\beta))$ is a curve on $\mathcal{M}$ passing through M_o ($\beta = 0$) and $P(M(\alpha^*))$ ($\beta = \alpha^*$). Along this curve $Q(M_\beta) = J(c(\beta))$. Hence the minimum of Q along the curve occurs at $\beta = \beta^*$. Thus $J(c(\beta^*)) = Q(M_{\beta^*}) \leq Q(P(M(\alpha^*)))$. The other inequalities follow from part (1) and the definition of α^*.

3. Since (F, g, h) is a descent direction for Q at M_o, there exists $\alpha_o > 0$ such that $Q(M_o + \alpha d) \leq Q(M_o)$ for all $\alpha \in [0, \alpha_o]$. For $\alpha \geq 0$ let $M_\alpha = (A(c_o + \alpha h), b(c_o + \alpha h), c_o + \alpha h)$. Then for all $\alpha \in [0, \alpha_o]$, $J(c_o + \alpha h) = Q(M_\alpha) \leq Q(M_o + \alpha d) \leq Q(M_o) = J(c_o)$. The first inequality follows from part (1); the second follows from the fact that d is a descent direction for Q at M_o. ∎

Theorem 4.4 indicates that each step of an iterative minimization of $Q(M)$ can be improved by exploiting the formulas $A(c)$ and $b(c)$ to project the next approximation onto the manifold $\mathcal{M}$. Moreover, part (2) indicates that minimizing $J(c)$ in the direction h_k from c_k yields a greater decrease in the least squares objective than either minimizing $Q(M)$ in the direction d_k from M_k and then projecting, or simply minimizing $Q(M)$ in the direction d_k from M_k. Other issues aside, this suggests that obtaining the least squares estimate by iteratively minimizing $J(c)$ is more efficient than a similar scheme applied to $Q(M)$. The third part of the theorem shows that at any point on the manifold $\mathcal{M}$, every descent direction for Q yields a corresponding descent direction for J. If we combine this with part (2) we see that minimization of J along this direction will yield a smaller value of the least squares objective function than minimizing Q in the given descent direction. Note that parts (2) and (3) of the theorem do not generally hold for M_o off the manifold $\mathcal{M}$.

Of course, J is a more complex function than Q and hence it is conceivable that the necessary computations in minimizing J are also more complex. However, as far as the gradient is concerned

this is not the case. To see this, let $M(c) = (A(c), b(c), c)$. Then for each $h \in \mathbb{R}^2$,

$$DJ(c)h = D_A Q(M(c)) \cdot D_c A(c)h + D_b Q(M(c)) D_c b(c)h + D_c Q(M(c))h$$

Since $M(c)$ lies on $\mathcal{M}$, $D_A Q(M(c)) = D_b Q(M(c)) = 0$. Then from (4.8),

$$\begin{aligned} \nabla J(c) &= D_c Q(M(c)) \\ &= \sum_{j=1}^{N} \left(w_j' - \frac{A(c)w_j + b(c)}{c^T w_j + 1} \right)^T \frac{A(c)w_j + b(c)}{c^T w_j + 1} \frac{w_j}{c^T w_j + 1} \end{aligned} \tag{4.18}$$

The computation of $A(c)$ and $b(c)$ is equivalent to the computation of $\nabla_A Q$ and $\nabla_b Q$, and can be efficiently accomplished by solving the linear system (4.9). The computation of ∇J given $A(c)$ and $b(c)$ is equivalent to the computation of $\nabla_c Q$. Thus the computation of the gradient of J is no more complex than computing the gradient of Q.

We note that at this point, we can prove the following:

Theorem 4.5: *If the set of admissible estimates C_o is compact, then J has a global minimum in C_o and Q has a global minimum in $\mathbb{R}^{2\times 2} \times \mathbb{R}^2 \times C_o$.*

Proof. From Theorems 4.2 and 4.3, we have that J is continuous over the compact set C_o, so it must have a global minimizer $\hat{c}$ in C_o. For any $M = (A, b, c)$ with $c \in C_o$, we have from the first part of Theorem 4.4 that $Q(M) \geq Q(P(M)) = J(c)$, so the global minimizer of Q must be $(A(\hat{c}), b(\hat{c}), \hat{c})$. ∎

4.6 Second-Derivative Methods

It is well known that minimization methods based on the second derivative of the objective function have superior rates of convergence. These methods are based on various modifications of the Newton-Raphson and Gauss-Newton schemes (see Appendix B). Applied to Q, these operate by setting

$$M_{k+1} = M_k - H(M_k)^{-1} \nabla Q(M_k)$$

where $H(M_k)$ is either the Hessian of Q at M_k or a suitable approximation.

If we define $\hat{w}_j = \frac{A(c)w_j + b(c)}{c^T w_j + 1}$, then we can write

$$Q(M) = \frac{1}{2}\sum_{j=1}^{N}(w'_j - \hat{w}_j)^T(w'_j - \hat{w}_j)$$

Then

$$\begin{aligned} DQ(M) &= -\sum_{j=1}^{N}\left(w'_j - \hat{w}_j\right)^T D\hat{w}_j \\ D^2Q(M) &= \sum_{j=1}^{N} D\hat{w}_j^T D\hat{w}_j - \sum_{j=1}^{N}\left(w'_j - \hat{w}_j\right)^T D^2\hat{w}_j \end{aligned}$$

$D^2Q(M)$ is the Hessian of Q at M and the first term is the Gauss-Newton approximation of the Hessian.

It is straightforward to derive expressions for the Hessians of Q and J and their Gauss-Newton approximations. The Hessian for J is quite cumbersome since J depends on c both directly and through the dependence of $A(c)$ and $b(c)$ on c. The result is:

$$H = \sum_{j=1}^{N}\frac{1}{q_j^2(c)}\left[(\hat{w}_j - 2\varepsilon_j)^T\hat{w}_j w_j w_j^T - N_j^T(\hat{w}_j - \varepsilon_j)w_j^T\right] \tag{4.19}$$

where

$$\begin{aligned} \varepsilon_j &= \left(w'_j - \hat{w}_j\right) \\ N_j &= \left[\frac{\partial A}{\partial c_1}w_j + \frac{\partial b}{\partial c_1} \quad \frac{\partial A}{\partial c_2}w_j + \frac{\partial b}{\partial c_2}\right] \end{aligned}$$

The Gauss-Newton approximation to the Hessian is:

$$H_{GN} = \sum_{j=1}^{N}\frac{1}{q_j^2(c)}(N_j - \hat{w}_j w_j^T)^T(N_j - \hat{w}_j w_j^T)$$

The details of these derivations, as well as an explanation of how to compute the partial derivatives of A and b with respect to c, are contained in Appendix C.

The complexity of these expressions raises the issue of obtaining efficiently computable approximations to the Hessian of J. For example, one natural approximation is to assume that A and b are independent of c so that N_j becomes 0. This results in the approximation to the Hessian

$$\hat{H} = \sum_{j=1}^{N}\frac{1}{q_j^2(c)}\left[(\hat{w}_j - 2\varepsilon_j)^T\hat{w}_j w_j w_j^T\right] \tag{4.20}$$

In fact, this matrix is the same as the 2×2 block of partials $\frac{\partial^2 Q}{\partial c^2}$. We will see how algorithms based on this approximation and the Gauss-Newton approximation fare in the presence of different types of noise in Section 4.7.

In general, we will use the following framework for our second-derivative methods to minimize $J(c)$. The only difference is the approximation to the Hessian used in step 3. Figure 4.7 illustrates the process.

Algorithm 4.1: *Newton scheme for minimizing J.*

1. *Initialize $c = 0$.*
2. *Compute the gradient of J exactly using (4.18). That is:*

$$\nabla J(c) = \sum_{j=1}^{N} \frac{1}{q_j(c)} \varepsilon_j^T \hat{w}_j w_j$$

3. *Approximate the Hessian $\frac{\partial^2 J}{\partial c^2}$ by some positive semidefinite matrix $\hat{H}$.*
4. *Use these quantities to update the value of c using an approximate Newton-Raphson step. That is:*

$$c \leftarrow c + \alpha \hat{H}^{-1} \nabla J(c)$$

5. *Use the new value of c to update the values of A and b using the formulas for $A(c)$ and $b(c)$. That is, solve:*

$$\begin{bmatrix} A(c) & b(c) \end{bmatrix} W(c) = V(c)$$

where $q_j(c) = c^T w_j + 1$ and

$$W(c) = \begin{bmatrix} \sum_{j=1}^{N} \frac{w_j w_j^T}{q_j^2(c)} & \sum_{j=1}^{N} \frac{w_j}{q_j^2(c)} \\ \sum_{j=1}^{N} \frac{w_j^T}{q_j^2(c)} & \sum_{j=1}^{N} \frac{1}{q_j^2(c)} \end{bmatrix}$$

$$V(c) = \begin{bmatrix} \sum_{j=1}^{N} \frac{w'_j w_j^T}{q_j(c)} & \sum_{j=1}^{N} \frac{w'_j}{q_j(c)} \end{bmatrix}$$

6. *Test for convergence. Exit or return to step 2.*

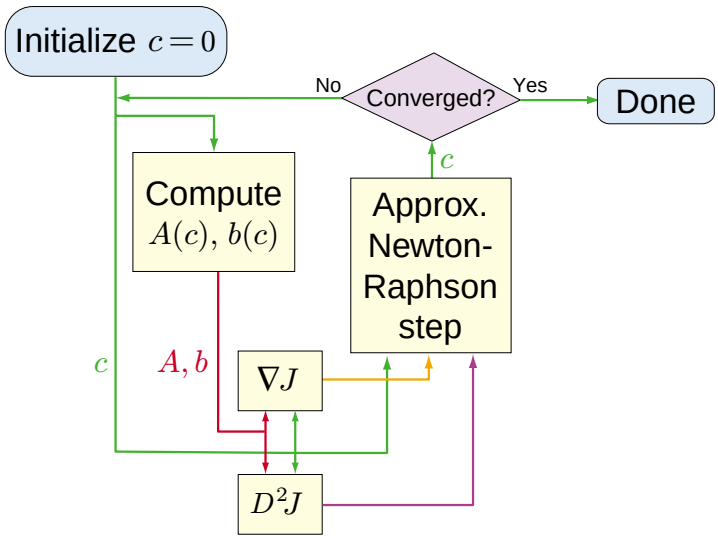


Figure 4.7: Proposed algorithm for minimizing $J(c)$.

The initialization of $c = 0$ in step 1 is justified in practice, since the values of c for projective transformations arising from real image processing problems often have $c = O(10^{-4})$ (see Table 4.1). This provides an additional advantage over the numerical minimization of $Q(M)$, which requires accurate initial estimates of the parameters A and b. Since these parameters relate to the zooming, rotation, and translation between an image pair, additional pre-processing is generally required to obtain even coarse initial estimates. Algorithms to find a value of α in step 4 that brings about a sufficient decrease in the cost function are generally based on a backtracking and cubic interpolation strategy [9]. Experimental results on the performance of this proposed algorithm are reported in Section 4.7.

4.7 Experimental Results

We implemented five minimization algorithms:

1. GNQ: Standard Gauss-Newton applied to Q.

2. GNJ: Standard Gauss-Newton applied to J.

3. $\hat{N}$: Approximate Newton applied to J, using $\hat{H}$ from (4.20).

4. $QdirJ$: Approximate Newton applied to J, using the projections of search directions from Q onto the manifold, as suggested by Theorem 4.4.

5. NJ: Full Newton applied to J, using the actual Hessian (4.19).

The algorithms were compared on six sets of point correspondences, each obtained from pairs of natural images related by projective transformations. Three of the image pairs were created by a rotating camera; point correspondences for these images were obtained automatically using the feature detection and matching algorithm described in Section 3.4. The other three image pairs are different views of planar scenes; in these cases the point correspondences were obtained manually. For all six images there is very little noise in the correspondences. However, in our experiments, we added noise of two different types to each of the measurements to test the algorithms' robustness. This noise was added prior to the normalization described in Section 4.3. The two types of noise were:

1. Gaussian noise of increasing variance. That is, random noise was added to each nominal correspondence $(x, y) \mapsto (x', y')$ to obtain $(\tilde{x}, \tilde{y}) \mapsto (\tilde{x}', \tilde{y}')$, where

$$\begin{aligned} \tilde{x} &= x + n_1 & \tilde{y} &= y + n_2 \\ \tilde{x}' &= x' + n_3 & \tilde{y}' &= y' + n_4 \end{aligned}$$

   and n_i, $i = 1, 2, 3, 4$ are independent zero-mean Gaussian random variables with variance σ^2.

2. As above, except n_i, $i = 1, 2, 3, 4$ is drawn from a zero-mean Gaussian distribution of variance 5 with probability $1 - p$, and from a uniform distribution over $[-50, 50]$ with probability p.

The first type of noise simulates increasingly inaccurate feature correspondences. Inaccuracies in real applications could come from poor sensors, suboptimal correspondence algorithms, or coarsely subsampled data. For example, if the images were subsampled by a factor of 16 in each direction before estimating correspondence, we could expect errors in the range ± 8 pixels in the original coordinates.

Example Number	First Image	Second Image	Number of w_j	$\hat{a}_{11}$ $\hat{a}_{21}$	$\hat{a}_{12}$ $\hat{a}_{22}$	$\hat{b}_1$ $\hat{b}_2$	$\hat{c}_1$ $\hat{c}_2$
1	Firestone1	Firestone2	70	1.1781 0.1316	-0.0640 1.1045	-173.88 0.01	0.0006 -0.0001
2	Firestone2	Firestone1	69	0.8486 -0.0976	0.0636 0.9728	146.44 -17.47	-0.0005 0.0001
3	B320fr0	B320fr1	90	0.8532 -0.0166	-0.0223 0.9639	8.51 -1.68	-0.0004 0.0002
4	Track1	Track2	30	0.9703 -0.0404	-1.5266 0.9630	83.10 -5.85	-0.0004 -0.0007
5	Atrium1	Atrium2	35	1.1146 -0.0790	0.6413 0.6171	-95.36 2.50	0.0005 -0.0008
6	Atrium2	Atrium3	33	0.7564 0.0010	-0.6599 0.8996	160.15 13.44	-0.0004 0.0009

Table 4.1: Information and nominal parameters for the 6 data sets.

The second type of noise simulates a generally good correspondence algorithm with increasing probability of obtaining a non-Gaussian outlier. Such outliers can occur, for example, when a block-matching algorithm "finds" a matching block with a lower mean-squared-error than the correct block induced by camera and object motion.

The information about the test images and the nominal (zero-noise) estimated projective transformation parameters for each example are given in Table 4.1. For all examples, the five different minimization algorithms all converged to the same projective transformation estimate. The 2-dimensional methods were initialized with $c = 0$. The 8-dimensional methods were initialized with $A = I, b = 0, c = 0$. In each case we ensured that the algorithms employed the same computational procedures and tests for convergence in the appropriately-dimensioned space. Namely, the algorithm terminates when either of the following conditions are fulfilled:

1. The relative change in the gradient is small enough:

$$\max_{1 \le i \le d} \left| \frac{\nabla f(x)_i \max\{|x_i|, t_i\}}{|f(x)|} \right| \le 10^{-6}$$

2. The relative change in successive values of the parameters is small enough:

$$\frac{|\Delta x|}{\max\{|x_i|, t_i\}} \le 10^{-6}$$

where $x = (a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2, c_1, c_2)$, $d = 8$ in the eight-dimensional case and $x = (c_1, c_2)$, $d = 2$ in the two-dimensional case, and f is the appropriately-dimensioned least-squares functional Q or J. Additionally, t_i is a "typical" value of parameter i to avoid problems with defining relative change when the parameters are small. In our tests we used $t = (1, 1, 1, 1, 100, 10, .0001, .0001)$. This choice is justified given the underlying parameters for our data set (see Table 4.1).

The number of floating point operations required for the three algorithms to converge with the purely Gaussian noise model is illustrated in Figures 4.8-4.13. Figures 4.8-4.10 pertain to the images taken by rotating cameras, and Figures 4.11-4.13 pertain to the images of planar scenes. The x axis in each figure is the variance σ^2 of the noise added to the correspondences. The number of floating point operations in each line graph is the mean of 100 trials at the same noise variance with different realizations of the random variables.

We can see that using the Q search directions on J is uniformly better than doing standard Gauss-Newton on Q, and that Gauss-Newton on J is uniformly better than both. The full Newton method on J does better than the Gauss-newton method on J at higher noise variances, though worse at lower noise variances. This is consistent with the observations in Dennis [9, p. 226].

Interestingly, the approximate Newton method using the Hessian approximation in (4.20) is only superior to other methods at high variances. This would indicate that while $\hat{H}$ can be computed efficiently, it is a poor approximation to the true Hessian H, i.e. the partial derivatives of A and b with respect to c are significant. This is confirmed by plotting the indicator $\|\hat{H} - H\|/\|H\|$ as a function of the noise variance for the first data set, illustrated in Figure 4.14. For comparison, we also show the indicator $\|H_{GN} - H\|/\|H\|$ for the Gauss-Newton method. We can see that in the presence of no noise, roughly 85% of the Hessian is "unapproximated" by $\hat{H}$, compared to only 0.1% for the Gauss-Newton case. Though the $\hat{N}$ iteration requires fewer floating point operations, 18 iterations were required compared to only 3 for Gauss-Newton. However, the Gauss-Newton approximation contains none of the terms in the full Hessian involving ε_j, the errors in the fitted data. Hence, as the noise variance increases, H_{GN} becomes an increasingly poor approximation. On the other hand, $\hat{H}$ contains one of the ε_j terms from the full Hessian and incrementally improves with increasing ε_j. Of course, the substantial partial derivative terms that make up most of the Hessian are still ignored.

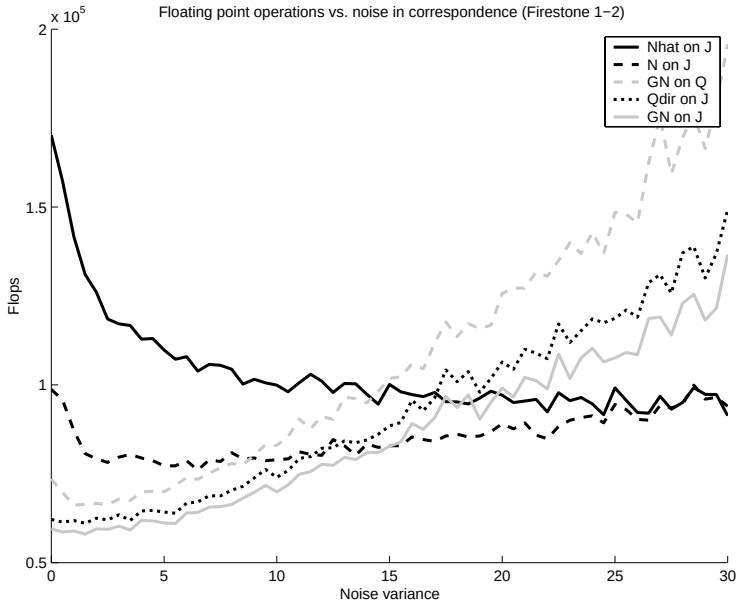


Figure 4.8: Floating point operation counts for the Firestone 1-2 data set, purely Gaussian noise.

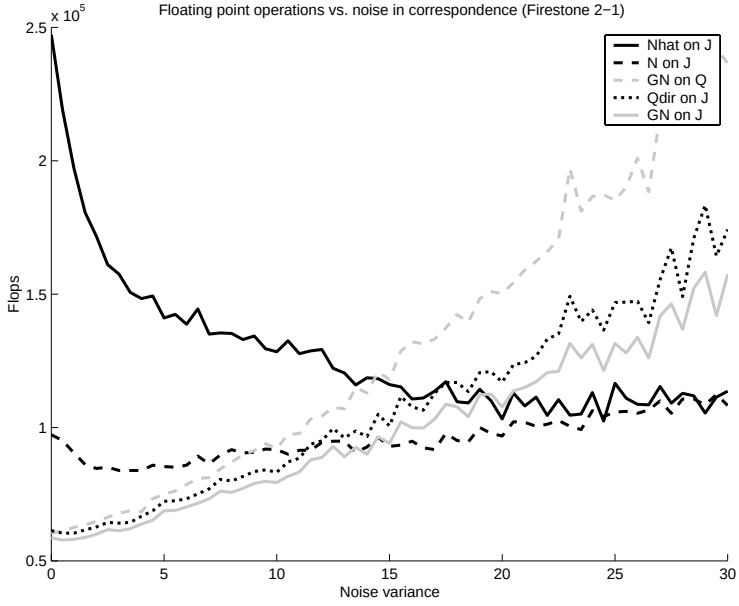


Figure 4.9: Floating point operation counts for the Firestone 2-1 data set, purely Gaussian noise.

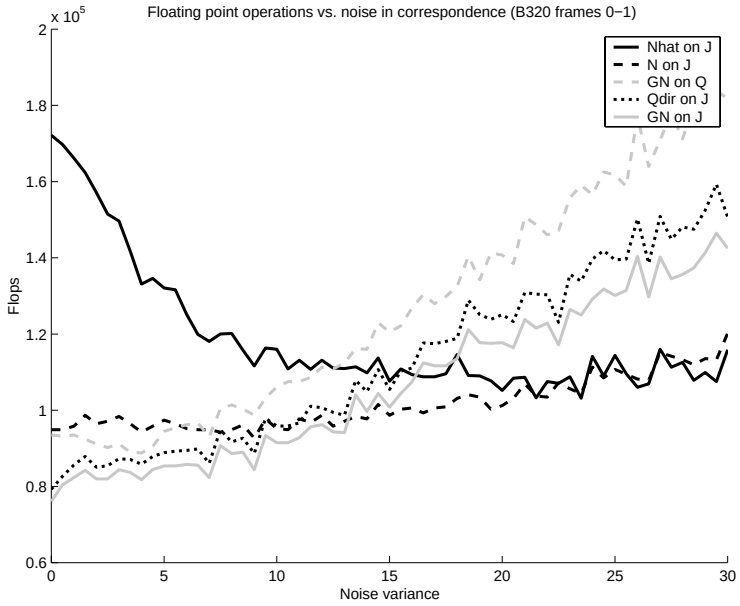


Figure 4.10: Floating point operation counts for the B320 0-1 data set, purely Gaussian noise.

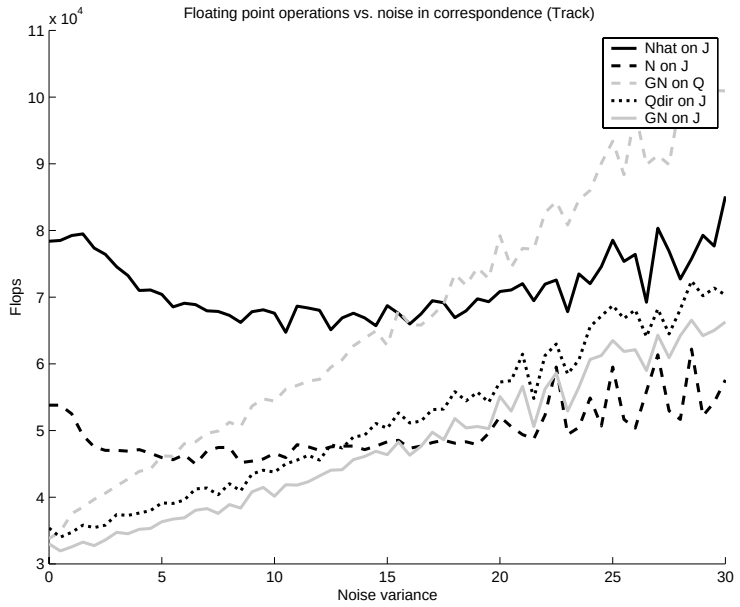


Figure 4.11: Floating point operation counts for the Track data set, purely Gaussian noise.

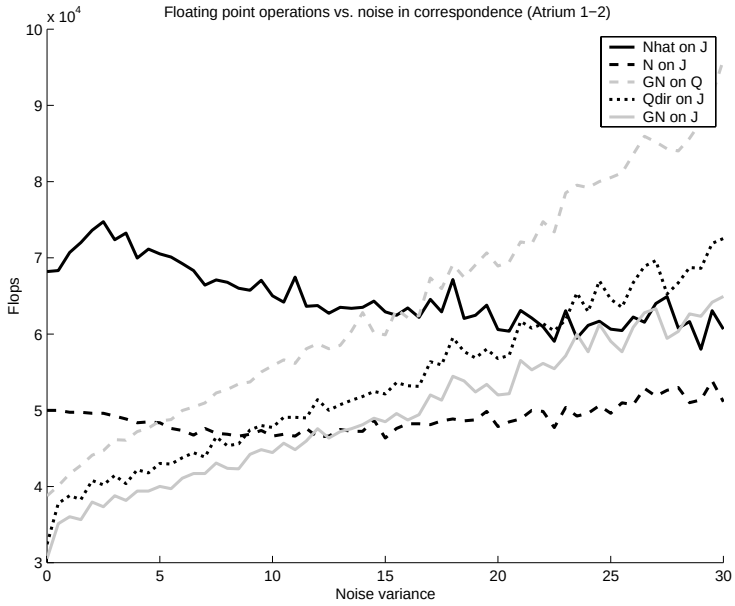


Figure 4.12: Floating point operation counts for the Atrium 1-2 data set, purely Gaussian noise.

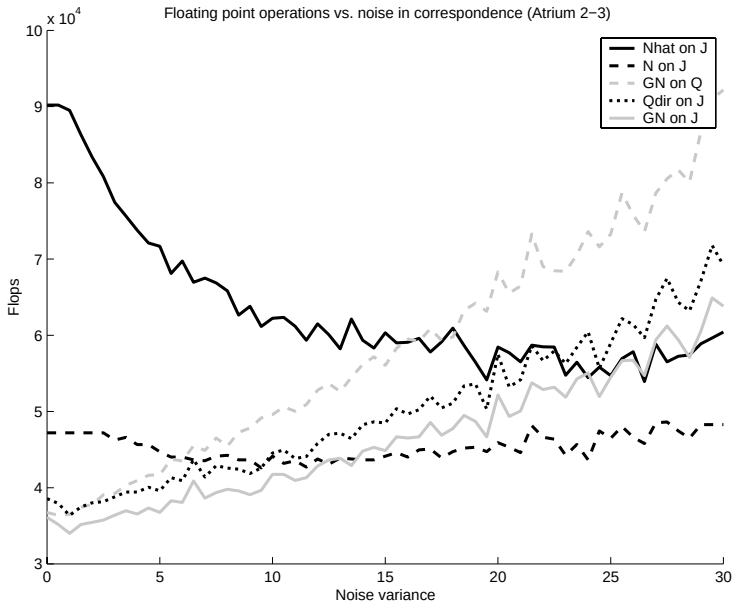


Figure 4.13: Floating point operation counts for the Atrium 2-3 data set, purely Gaussian noise.

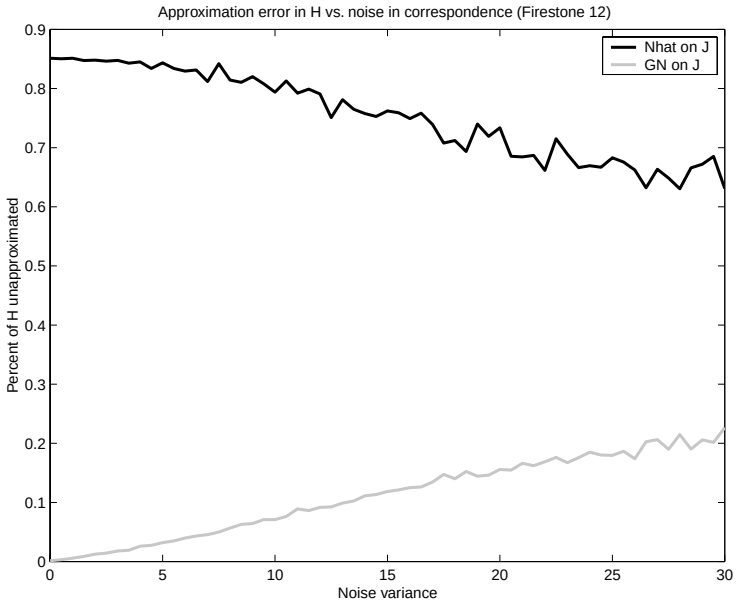


Figure 4.14: $\|\hat{H} - H\|/\|H\|$ **as a function of noise variance for** $\hat{N}$ **and** GNJ **methods.**

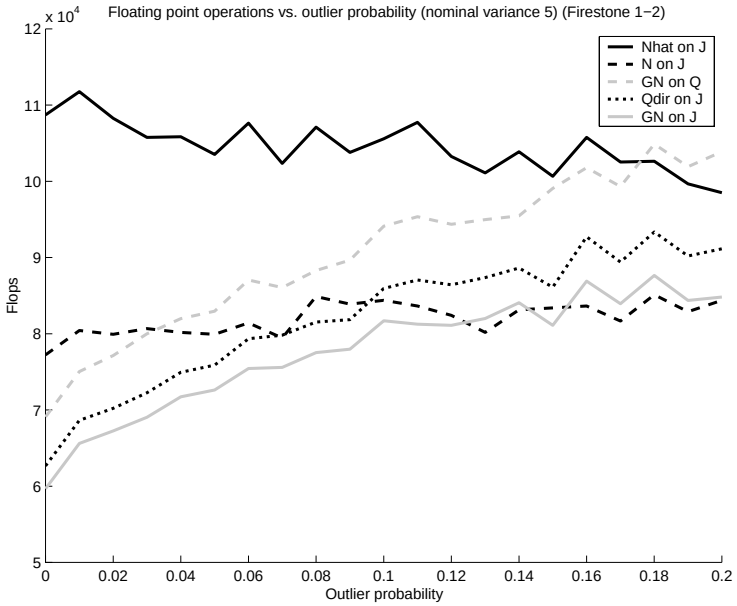


Figure 4.15: Floating point operation counts for the Firestone 1-2 data set, Gaussian noise with outliers.

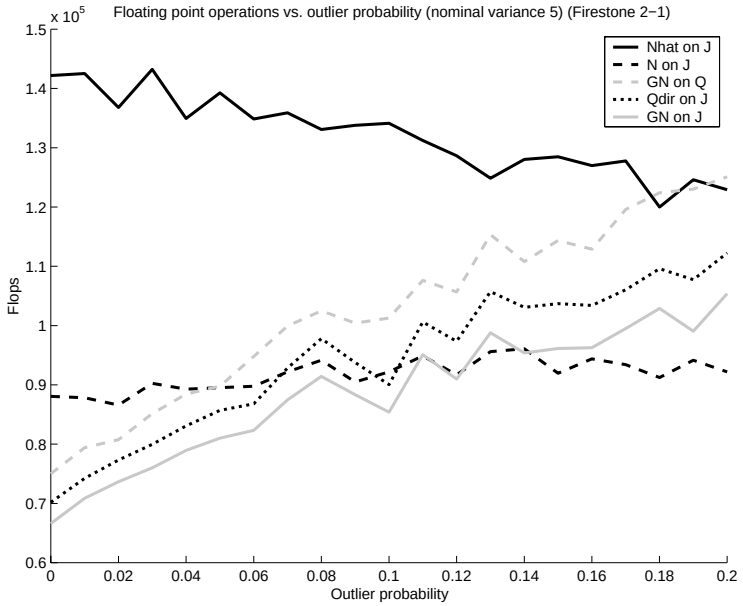


Figure 4.16: Floating point operation counts for the Firestone 2-1 data set, Gaussian noise with outliers.

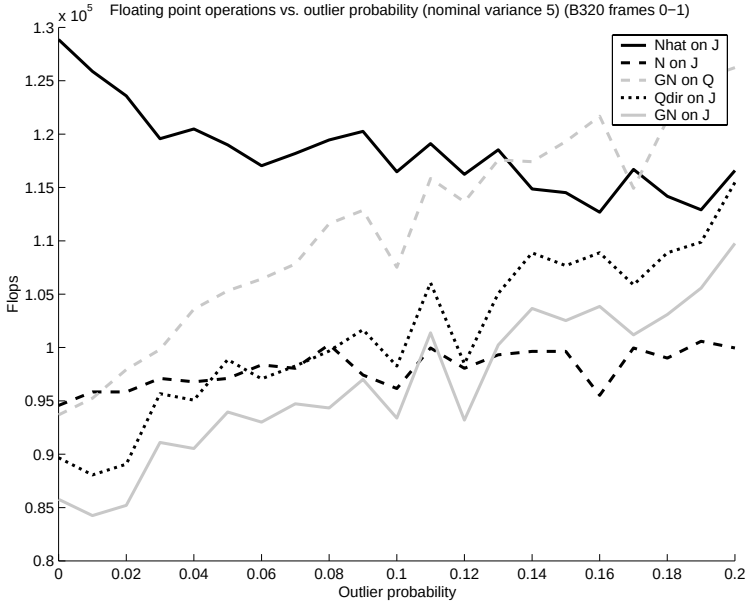


Figure 4.17: Floating point operation counts for the B320 0-1 data set, Gaussian noise with outliers.

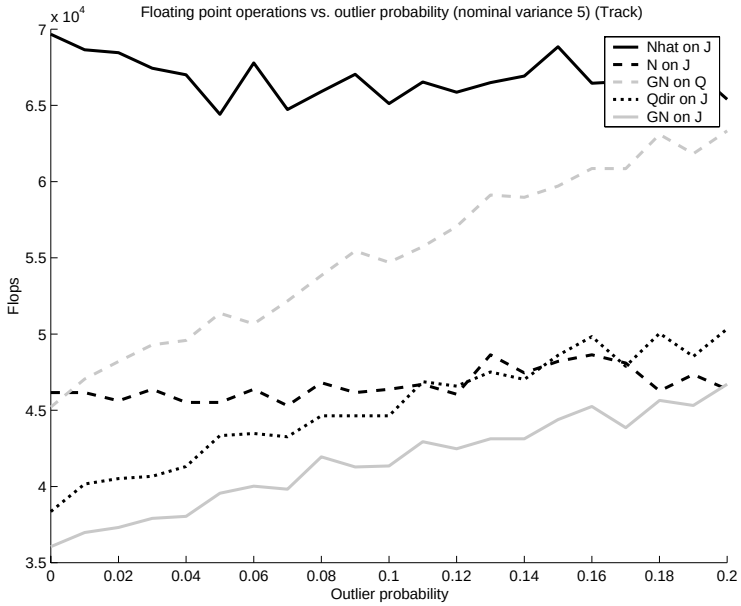


Figure 4.18: Floating point operation counts for the Track data set, Gaussian noise with outliers.

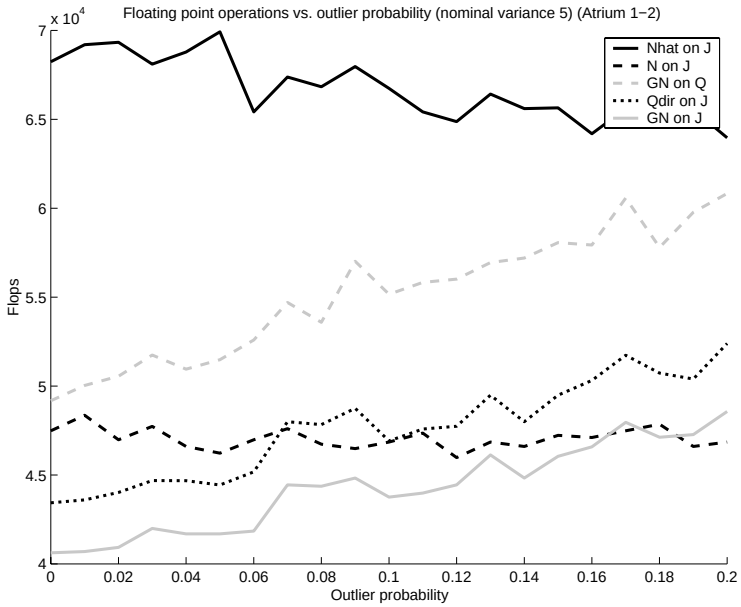


Figure 4.19: Floating point operation counts for the Atrium 1-2 data set, Gaussian noise with outliers.

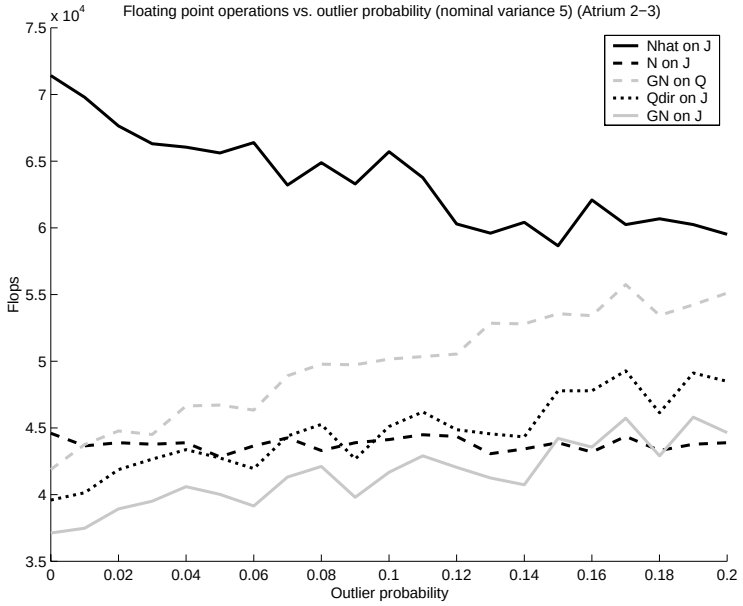


Figure 4.20: Floating point operation counts for the Atrium 2-3 data set, Gaussian noise with outliers.

The number of floating point operations required for the three algorithms to converge with the outlier noise model is illustrated in Figures 4.15-4.20. Again, Figures 4.15-4.17 pertain to the images taken by rotating cameras, and Figures 4.18-4.20 pertain to the images of planar scenes. The x axis in each figure is the probability p that a coordinate is an outlier. The number of floating point operations in each line graph is the mean of 100 trials at the same outlier probability with different realizations of the random variables.

The results here again indicate the superiority of the two-dimensional algorithms. The main difference is the lower rate of decrease of the $\hat{N}$ curves, which indicates that the Gauss-Newton method on J is a better choice overall when the correspondence contains outliers. Of course, a good estimation scheme will iteratively reject outliers until the noise can be well-modeled by a Gaussian distribution, and re-estimate, in which case the $\hat{N}$ method may be more efficient.

4.8 Conclusions

The experimental results indicate that obtaining the least squares estimate of the parameters of a projective transformation using the algorithms proposed in Section 4.6 to minimize $J(c)$ offers a distinct efficiency advantage over using a standard algorithm such as Gauss-Newton to minimize $Q(M)$.

Future research in this area includes a deeper investigation of how the relationship between the positions of the data points, the noise in their measurement, and the underlying projective transformation parameters affect the convergence of the algorithm. For example, our simulations indicate that the $\hat{N}$ algorithm presented is quite robust to high-variance noise. Its computational cost seems to decrease with noise variance while the costs of the other algorithms increase. However, we lack a rigorous analysis of why this is so.

Additionally, we hope to use the two-dimensional cost function $J(c)$ to analyze the existence and behavior of local minima. We have been able to construct data sets that induce a cost function $J(c)$ with multiple local minima over the region C_o, and have experimentally obtained bifurcation diagrams for the minima as the configuration of the data points is continuously varied. However, in our experience with projective transformations arising from real data, we have never observed multiple local minima in the least-squares cost functional. We would like to prove or disprove the hypothesis that in the general case (e.g. a large number of noisy measurements obtained from real images), the cost function is convex over C_o and hence has a unique global minima in this domain.

We only address the estimation of a single projective transformation here, but there are natural extensions to the joint estimation of the projective transformations relating several images, e.g. frames of a video sequence. The composition of multiple pairwise estimates is suboptimal for the joint problem, and can lead to unstable error growth. Additional issues arise when the images are constrained to form a seamless 360° panorama, as in Szeliski [3].

4.9 References

[1] K. Kanatani. *Geometric Computation for Machine Vision.* Clarendon Press, 1993.

[2] S. Mann and R.W. Picard. Video Orbits of the Projective Group: A Simple Approach to Featureless Estimation of Parameters. *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1281-1295, 1997.

[3] R. Szeliski and H. Shum. Creating Full View Panoramic Image Mosaics and Environment Maps. *Computer Graphics (SIGGRAPH '97)*, pp. 251–258, August 1997.

[4] Y.P. Tan, S.R. Kulkarni and P.J. Ramadge. A New Method for Camera Motion Parameter Estimation. *Proc. ICIP 1995*, pp. 406–409, 1995.

[5] H.S. Sawhney, A. Ayer and M. Gorkani. Model Based 2D and 3D Dominant Motion Estimation for Mosaicing and Video Representation. *Proc. ICCV 1995*, pp. 583–590, 1995.

[6] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu. Efficient Representations of Video Sequences and Their Applications. *Signal Processing: Image Communication, special issue on Image and Video Semantics: Processing, Analysis, and Application*, Vol. 8, No. 4, pp. 327–351, May 1996.

[7] R.Y. Tsai and T.S. Huang. Estimating the Three-Dimensional Motion Parameters of a Rigid Planar Patch. *IEEE Trans. ASSP*, vol. 25, no. 6, pp. 1147–1152, December 1981.

[8] R. Radke, P. Ramadge, S. Kulkarni, T. Echigo, S. Iisaku. Recursive Propagation of Correspondences with Applications to the Creation of Virtual Video. In *Proc. ICIP 2000*, September 2000.

[9] J.E. Dennis, Jr. and R.B. Schnabel. *Numerical Methods for Unconstrianed Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.

[10] R. Radke, P. Ramadge, T. Echigo, and S. Iisaku. Efficiently Estimating Projective Transformations. In *Proc. ICIP 2000*, September 2000.

[11] Y.P. Tan. *Digital Video Analysis and Manipulation.* Ph.D Thesis, Department of Electrical Engineering, Princeton University, November 1997.

[12] G.H. Golub and V. Pereyra. The Differentiation of Pseudo-Inverse and Non-Linear Least Squares Problems Whose Variables Separate. *SIAM J. Numer. Anal.*, vol. 10, pp. 413–432, 1973.

Chapter 5

Correspondence

Many image and video processing problems hinge on possessing a dense (subpixel-level) estimate of correspondence between a set of still images. Applications of correspondence are diverse, including motion compensation in video coding, construction of a 3-D model of a scene from images, and the association of points on similar objects for computer graphics effects.

Of course, the more that is known *a priori* about the position and motion of cameras and scene objects, the easier the correspondence problem becomes. However, in this chapter, we assume no prior knowledge about the content of the scene or the cameras that produced the images, since this information is often unavailable, e.g. for archival video.

Moreover, we distinguish between photometric techniques, which match points based entirely on the local variation of intensity between images, and geometric techniques, which attempt to find correspondence consistent with a physical scene. Photometric correspondence is useful in domains such as video coding, but less useful for applications where geometry is crucial, such as the virtual view synthesis algorithms we will discuss in the next chapter.

The classical correspondence problem is a fundamental and difficult problem in computer vision, as evidenced by more than 30 years of research. In Section 3.5, we reviewed several lines of investigation. Many notable approaches derive from optical flow and other techniques in stereo.[1]

[1]The word "stereo" is generally used in the context of cameras whose centers of projection are closely separated with respect to their distance to the scene, so that the effects of occlusions are minor. The situation in which the centers of projection of the cameras are widely separated with respect to their distance to the scene is sometimes referred to as the wide-baseline case.

While in photometric applications, an unstructured optical flow field may be an adequate representation of correspondence between an image pair, there are many practical situations in which a parametrized or structured correspondence is induced by the geometry of the cameras. For example, when a camera undergoes rotation only, the projective transformations discussed in Chapter 4 provide one example of correspondence over entire image planes parametrized by only eight real numbers. Layered-motion techniques also fall into this category.

However, for an arbitrary image pair of the same scene, the correspondence between the images has no simple global parametrization. The only *a priori* constraint on correspondence is the well-known epipolar constraint [1]. In theory, this reduces the correspondence problem to a series of 1-D matching problems. We review several approaches to solving the correspondence problem in the context of conjugate epipolar lines in Section 5.1. While these techniques are unstable in the small-baseline case[2], our interest here and in the following chapters is in the wide-baseline setting.

Virtually every epipolar-line-based correspondence algorithm makes the assumption that scene points are projected onto conjugate epipolar lines in the same order. This is called the monotonicity assumption. Typically, it is made so that dynamic programming or polynomial-time algorithms can be used to efficiently obtain solutions. However, as we illustrate in Section 5.2, the monotonicity assumption is generally invalid in the wide-baseline case.

Our goal in Section 5.3 is to fully describe the class of sets of corresponding points that can arise from a real imaging system. Instead of making the monotonicity assumption, we consider the correspondence induced by arbitrarily complicated scenes, and encapsulate this structure in the correspondence graph, the set of all points that are visible in two conjugate epipolar lines. Using the formalism of correspondence graphs, we can ensure that any estimated correspondence is consistent with a physical imaging system, which is especially important for geometric applications.

The second main contribution of the chapter is Section 5.4, in which we present an algorithm for estimating correspondence graphs from real images. As a result, we can generate dense, physically consistent correspondence between images taken by widely separated cameras, useful for applications where geometric accuracy is crucial. Each step of the estimation algorithm is illustrated in Section 5.5 with an example from natural, outdoor video.

[2]The fundamental matrix is undefined when the camera centers are coincident.

The correspondence graph was introduced in [2], and is used in the virtual video applications we discuss in Chapters 6 and 7.

5.1 Review of Epipolar Correspondence Algorithms

As reviewed in Section 2.5, epipolar lines exist in conjugate pairs (ℓ_0, ℓ_1), such that the match to a point $w \in \ell_0$ must lie on ℓ_1, and vice versa. This means that the correspondence problem is fundamentally a one-dimensional problem, not a two-dimensional one. If an estimate of the epipolar geometry is available, many correspondence algorithms exist to exploit this constraint.

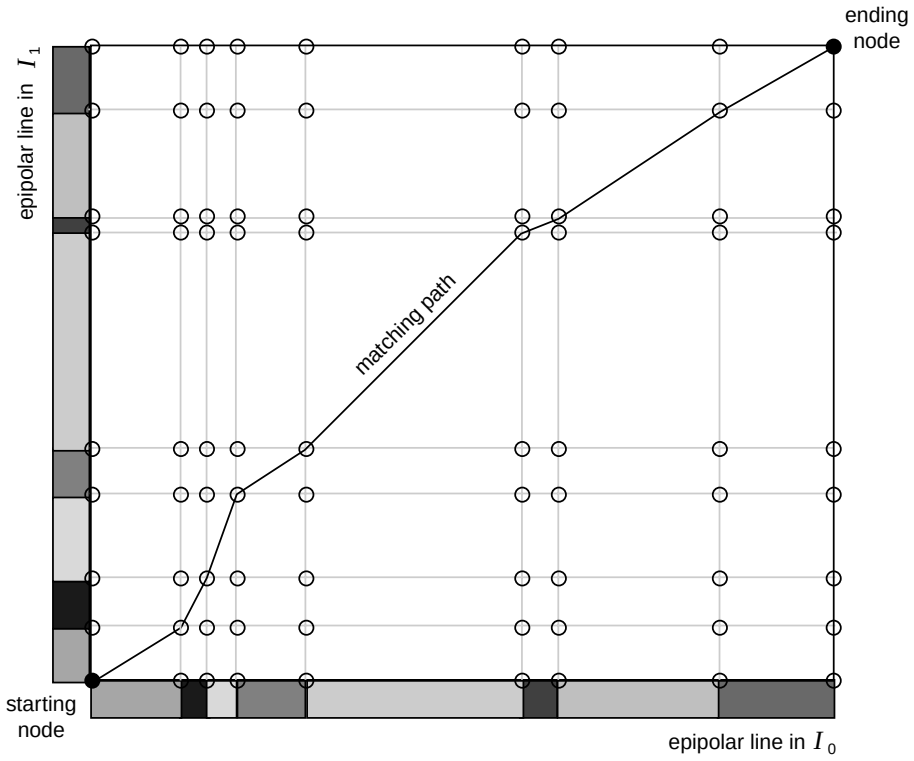


Figure 5.1: Matching graph for conjugate epipolar lines.

Each approach we review below makes the monotonicity assumption that correspondences appear along conjugate epipolar lines in the same order. This allows the use of dynamic programming [3] techniques to efficiently solve the various estimation problems. The result of the estimation for a conjugate epipolar line pair (ℓ_0, ℓ_1) can then be expressed as a monotonic path through $\ell_0 \times \ell_1$,

as illustrated in Figure 5.1. However, we will show in the remainder of the chapter that correspondence from real images can be considerably more complex, and discuss how to estimate 1-D correspondence in full generality.

Some algorithms build in constraints that ensure similar correspondence is estimated across adjacent epipolar lines. This is especially important in light of the observation in [4] that multiple global minima may exist for problems solved at each conjugate epipolar line pair.

5.1.1 Basic Dynamic Programming: Ohta and Kanade

Ohta and Kanade [5] described a dynamic programming approach in which the nodes of the program correspond to edges detected in each epipolar line. The entities that are matched between conjugate epipolar lines are intervals of nearly constant-intensity pixels. Points in a pair of matched intervals are put into correspondence by linearly interpolating between the endpoints.

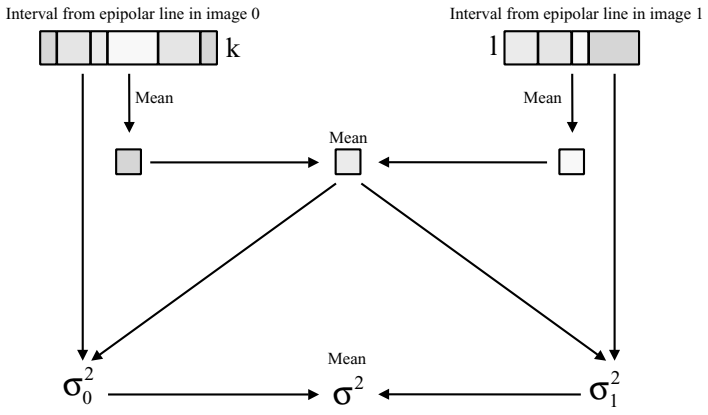


Figure 5.2: Ohta and Kanade interval-matching cost function.

The function used to measure the cost of matching an interval $i_0 \in \ell_0 = \{a_1, \ldots, a_k\}$ and $i_1 \in \ell_1 = \{b_1, \ldots, b_l\}$ is based on the variance σ^2 of the intensities in the two intervals from a sample mean m, calculated as:

$$m = \frac{1}{2}\left(\frac{1}{k}\sum_{i=1}^{k} a_i + \frac{1}{l}\sum_{j=1}^{l} b_j\right)$$

$$\sigma^2 = \frac{1}{2}\left(\frac{1}{k}\sum_{i=1}^{k}(a_i - m)^2 + \frac{1}{l}\sum_{j=1}^{l}(b_j - m)^2\right)$$

The cost function is illustrated schematically in Figure 5.2. Both intervals contribute equally to the mean and variance. The cost of the segment that matches these intervals is then computed as

$$C = \sigma^2\sqrt{k^2 + l^2}$$

The cost function is motivated by the assumption that pixels in matching intervals arise from a homogeneous-intensity surface in the scene and therefore have similar image intensities, so that the variance between correctly matched intervals should be small. A slightly different, somewhat ad-hoc cost was defined for an occluded path.

The authors also described a higher-dimensional matching problem over the entire image pair in which the nodes in the dynamic program are edges that cross many epipolar lines. This formulation explicitly enforces consistency between nearby epipolar lines.

Once the cost function and the nodes of the problem are fixed, it is straightforward to apply dynamic programming to find the least-cost path through the epipolar-line matching graph.

5.1.2 Bayesian Approach: Belheumer

Belheumer [6] discussed a series of explicit prior models for the structure of a scene, and developed a Bayesian approach to solving the correspondence problem for each model. The goal is to obtain the maximum *a posteriori* (MAP) estimate of the disparities (i.e. motion vectors between corresponding points) between the epipolar line pair. This was accomplished by defining a prior distribution on the disparity function as a stochastic process comprised of Brownian motion and Poisson processes. In the most complicated model, the scene is composed of multiple objects, the surfaces of which may be steeply sloping or have creases.

Dynamic programming is applied in the case where conjugate epipolar line pairs are treated independently, and a heuristic variant dubbed "iterated stochastic dynamic programming" is used to enforce consistency constraints. In addition, this approach is notable in its exploration of the relationship between foreground objects and "half-occluded" (i.e. visible in one image but not

both) background regions. We discuss the full structure and implications of this relationship in the next section.

5.1.3 Maximum Likelihood Approach: Cox et al.

While Belheumer placed a prior density on the structure of the scene and computed the Bayesian MAP estimate, Cox et al. [4] derived a maximum likelihood (ML) estimate for the stereo correspondence problem that requires no prior distributions. Their maximum likelihood algorithm assumes that the intensities of corresponding pixels are normally distributed about a true common value, which leads to a matching cost based on the weighted squared error between the intensities of candidate corresponding points. The authors were able to obtain good results without further assumptions about local smoothness of correspondences.

A system of cohesiveness constraints is also introduced to minimize the number of horizontal discontinuities in each epipolar line and the number of vertical discontinuities across epipolar lines. Occlusions are modeled, though the cost of occlusion is the same regardless of local image detail.

5.1.4 Maximum-Flow Graph: Ishikawa and Geiger

Ishikawa and Geiger [7] described an approach to compute the disparity map by solving a global optimization problem that modeled occlusions, discontinuities, and epipolar line interactions. The optimization problem is mapped to a maximum-flow problem on a directed graph, which can be solved in polynomial time. In their model, a disparity discontinuity in one image is constrained to match an occluded region in the other image. Edges and junctions are used as matching primitives. The capacities of edges in the graph are adjusted to enforce the monotonicity constraint, require consistency between epipolar lines and smoothness in disparity, and penalize discontinuities and occlusions. However, their algorithm appears to be very slow.

5.1.5 Curve Matching: Tomasi and Manduchi

Tomasi and Manduchi [8] proposed a novel approach for matching epipolar lines, based on representing each epipolar line as a curve in a higher dimensional space whose coordinate axes are

intensity, derivative of intensity, and so on. Then the correspondence problem can be cast as finding the matching of points on a pair of curves in $\mathbb{R}^N$ that minimizes a radial distance function. Occluded regions are represented in this context by unmatched loops of the curves.

5.2 Non-Monotonicity

Here we illustrate that the order of corresponding points along conjugate epipolar lines is not invariant from image to image, even though this is the basis of the commonly invoked monotonicity assumption. Consider the scene in Figure 5.3, in which a thin post stands before a wall. The left camera sees point A to the left of point B, while the right camera sees point A to the right of point B.

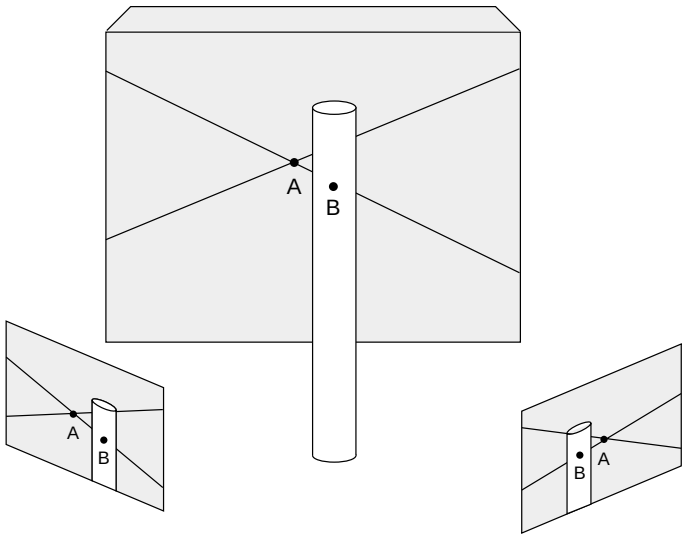


Figure 5.3: The "double nail illusion".

This phenomenon is sometimes called the "double-nail illusion", and is dismissed as relatively uncommon in stereo. However, it occurs frequently in images from wide-baseline video, which comprise most of the examples in this thesis.

Figure 5.4 illustrates regions of two real images of the same scene, rectified so that epipolar lines are horizontal. The numbered objects appear in different orders along conjugate epipolar lines due to the large perspective difference between the images. Each inconsistency in ordering generates a

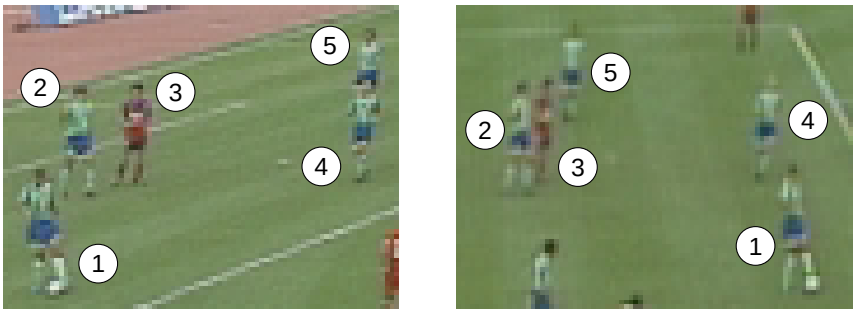


Figure 5.4: Violations of monotonicity.

local violation of the monotonicity assumption in the affected conjugate epipolar lines. A monotonic path through a matching graph such as the one illustrated in Figure 5.1 cannot represent the correct matching.

Unfortunately, relaxing the monotonicity assumption to allow arbitrary matching of points between conjugate epipolar lines results in a problem of high combinatorial complexity, not suitable for dynamic programming [10]. However, the set of correspondences that are physically realizable is not entirely unconstrained, and has a specific structure that we derive in the next section.

5.3 The Correspondence Graph

5.3.1 Constraints on Correspondence

In the following, we fix a pair of cameras $(\mathcal{C}_0, \mathcal{C}_1)$ whose centers of projection are O_0 and O_1, respectively. These cameras have associated image planes $\mathcal{P}_0$ and $\mathcal{P}_1$, that lie between the cameras' respective centers of projection and the scene $\mathcal{S}$, a collection of points in $\mathbb{R}^3$. Select a plane Φ containing the baseline, and view the intersection of Φ with the camera centers, the image planes, and the scene points as an imaging system with a 2-D scene $\mathbf{S} = \mathcal{S} \cap \Phi$ and 1-D image planes (the pair of conjugate epipolar lines (ℓ_0, ℓ_1)). This is illustrated in Figure 5.5.

We fix a coordinate system (x, y) on Φ by letting $O_0 = (0, 0)$ and placing O_1 at $(1, 0)$.[3] The

[3]Scene points are assumed to have positive y coordinates.

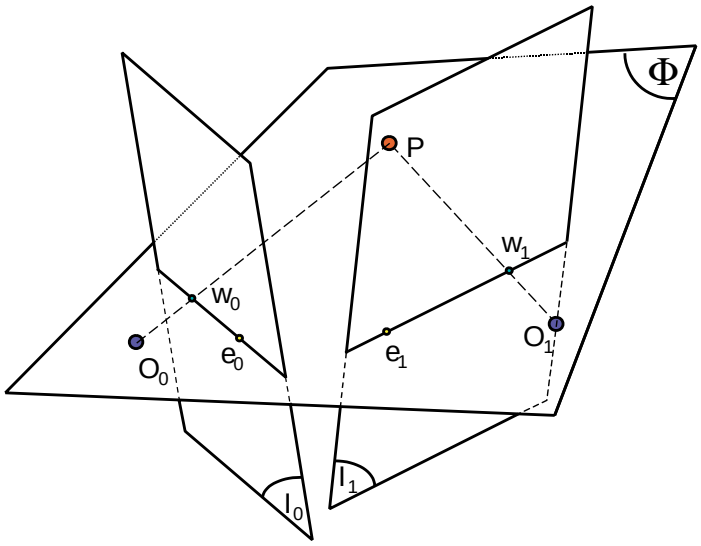


Figure 5.5: Epipolar geometry.

epipolar lines ℓ_0 and ℓ_1 inherit natural one-dimensional coordinate systems (denoted i and j respectively), oriented so that increasing i and j correspond to increasing x. In this setting, a correspondence is the realization of a point (x, y) in the scene as a pair $(i, j) \in \ell_0 \times \ell_1$. We will denote as $\mathbf{S}'$ the representation of the scene $\mathbf{S}$ in (i, j)-space.

Explicitly, the bijective transformation from (x, y)-space to (i, j)-space is given by:

$$i(x,y) = f_0 \frac{x \sin\gamma_0 - y\cos\gamma_0}{x\cos\gamma_0 + y\sin\gamma_0} \tag{5.1}$$

$$j(x,y) = f_1 \frac{(x-1)\sin\gamma_1 - y\cos\gamma_1}{(x-1)\cos\gamma_1 + y\sin\gamma_1} \tag{5.2}$$

where γ_i is the angle the optical axis of $\mathcal{C}_i$ makes with the positive x-axis. We implicitly define two new coordinate systems, (r_0, θ_0), (r_1, θ_1) in terms of the (x, y) coordinate system by:

$$(x,y) = (r_0\cos\theta_0,\ r_0\sin\theta_0)$$

$$(x,y) = (r_1\cos\theta_1 + 1,\ r_1\sin\theta_1)$$

These are just the polar coordinates of (x, y) with respect to O_0 and O_1, respectively. It is clear that the mappings between the four sets of coordinates are bijective, and hence the coordinate transforms $(i, j) = J_0(r_0, \theta_0)$ and $(i, j) = J_1(r_1, \theta_1)$ are well-defined. An important property of these

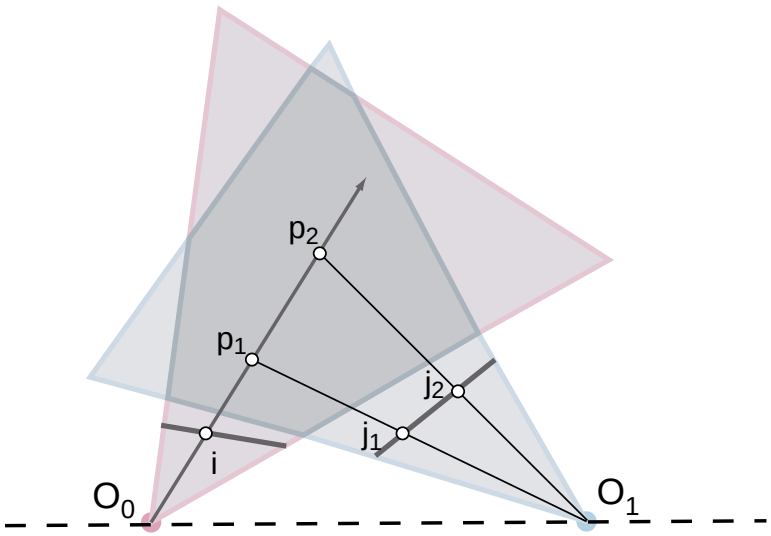


Figure 5.6: Mapping from (x, y)-space to (i, j)-space.

mappings is:

Proposition 5.1: *For any fixed $\theta_0, \theta_1 \in (0, \pi)$,*

$$\frac{\partial i}{\partial r_0} = 0 \qquad \frac{\partial j}{\partial r_0} > 0$$
$$\frac{\partial i}{\partial r_1} < 0 \qquad \frac{\partial j}{\partial r_1} = 0$$

Proof. The partials of (i, j) with respect to (x, y) are

$$\begin{bmatrix} \frac{\partial i}{\partial x} \\ \frac{\partial i}{\partial y} \end{bmatrix} = \frac{f_0}{(x\cos\gamma_0 + y\sin\gamma_0)^2} \begin{bmatrix} y \\ -x \end{bmatrix}$$
$$\begin{bmatrix} \frac{\partial j}{\partial x} \\ \frac{\partial j}{\partial y} \end{bmatrix} = \frac{f_1}{((x-1)\cos\gamma_0 + y\sin\gamma_0)^2} \begin{bmatrix} y \\ -(x-1) \end{bmatrix}$$

Then by the chain rule, we obtain

$$\begin{bmatrix} \frac{\partial i}{\partial r_0} \\ \frac{\partial i}{\partial r_1} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{-f_0\sin\theta_1}{(x\cos\gamma_0 + y\sin\gamma_0)^2} \end{bmatrix}$$
$$\begin{bmatrix} \frac{\partial j}{\partial r_0} \\ \frac{\partial j}{\partial r_1} \end{bmatrix} = \begin{bmatrix} \frac{f_1\sin\theta_0}{((x-1)\cos\gamma_1 + y\sin\gamma_1)^2} \\ 0 \end{bmatrix}$$

Since $\theta_0, \theta_1 \in (0, \pi)$, $\sin\theta_0$ and $\sin\theta_1$ are positive, and the property is proven. ∎

The proof can also be seen from the diagram in Figure 5.6. The intuition is that any ray from O_0 maps to a line segment with fixed i in (i, j)-space, and provided that the ray is on the "right side" of the baseline, the segment is traversed in the direction of increasing j as we move away from O_0. Similarly, any ray from O_1 maps to a line segment with fixed j in (i, j)-space, which is traversed in the direction of decreasing i.

We can also derive a bound on the correspondences of points on rays from either camera.

Proposition 5.2: *For any fixed* $\theta_0, \theta_1 \in (0, \pi)$,

$$\lim_{r_0 \to \infty} (i(r_0, \theta_0), j(r_0, \theta_0)) = (f_0 \tan(\gamma_0 - \theta_0), f_1 \tan(\gamma_1 - \theta_0)) \quad (5.3)$$

$$\lim_{r_1 \to \infty} (i(r_1, \theta_1), j(r_1, \theta_1)) = (f_0 \tan(\gamma_0 - \theta_1), f_1 \tan(\gamma_1 - \theta_1)) \quad (5.4)$$

This follows from simple algebra. Equation (5.3) acts as an upper bound on j for fixed θ_0, and equation (5.4) acts as a lower bound on i for fixed θ_1. These also give the constant values of i for fixed θ_0 and j for fixed θ_1, respectively, that are implied by Proposition 5.1.

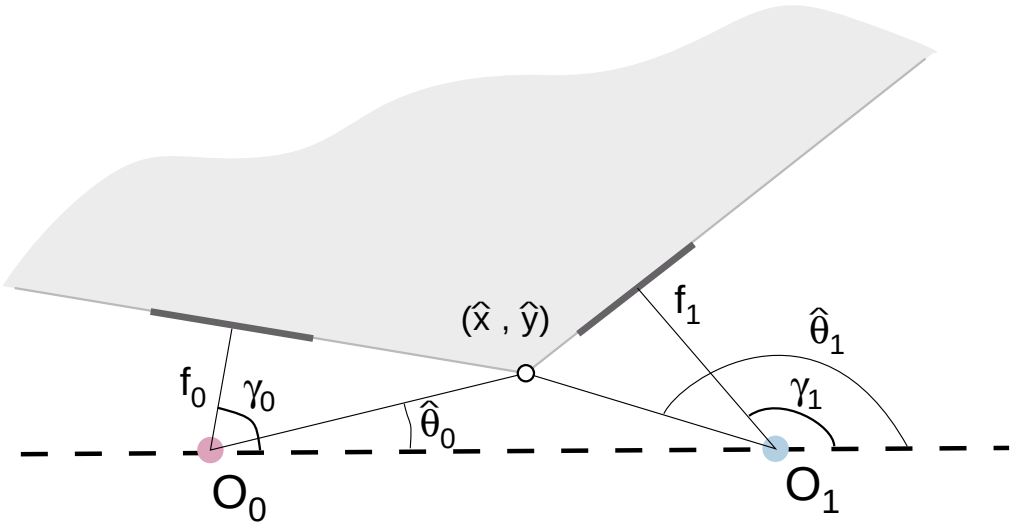


Figure 5.7: The set of points in front of both cameras.

To obtain a lower bound on j for fixed θ_0 and on i for fixed θ_1, we consider Figure 5.7. The bound is derived by requiring that a correspondence lies in front of both image planes. An important

point is $(\hat{x}, \hat{y})$, the intersection of the two image planes. This point induces two critical angles $\hat{\theta}_0$ and $\hat{\theta}_1$. We require that:

$$\theta_0 > \hat{\theta}_0 \qquad \theta_1 < \hat{\theta}_1 \tag{5.5}$$

Assuming a point satisfies these constraints, its r_0 and r_1 coordinates satisfy

$$r_0 > f_0 \sec(\gamma_0 - \theta_0) \qquad r_1 > f_1 \sec(\gamma_1 - \theta_1) \tag{5.6}$$

in order to be visible. Taken together, (5.5) and (5.6) induce a lower bound on j as a function of θ_0 and an upper bound on i as a function of θ_1, though it is not particularly instructive to present the formulas.

5.3.2 The Correspondence Graph

Now we present the main result of this chapter, the correspondence graph. This is a representation of all the points that are visible in both members of a conjugate epipolar line pair. In contrast to assumptions of other algorithms (e.g. Ohta and Kanade, Belheumer), the result is a matching path that need be neither monotonic nor continuous, and occlusions are explicitly removed from the graph instead of being approximated by vertical or horizontal line segments.

Belheumer [6] mentioned a "morphologically filtered version" of the disparity function between an epipolar line pair that is related to the correspondence graph. The filtering operation creates a continuous, monotonic path through the epipolar matching graph that includes regions that are "half-occluded", i.e. visible in one image only. However, this formalism only captures simple scenes that are constrained by monotonicity.

Now we formally define the correspondence graph:

Definition. The correspondence graph $C \subset \ell_0 \times \ell_1$ of a scene $\mathbf{S}$ with respect to the camera pair $(\mathcal{C}_0, \mathcal{C}_1)$ is the set of all points in $\mathbf{S}$ that are visible (i.e. unoccluded) in both ℓ_0 and ℓ_1, transformed into (i, j)-space.

The correspondence graph $C \subset \mathbf{S}'$. Generally $C \neq \mathbf{S}'$, since the correspondence graph takes occlusions into account and the transformed scene $\mathbf{S}'$ does not. However, we will prove that the correspondence graph can be easily obtained from the set $\mathbf{S}'$. We shall see that the construction is related to a certain morphological operation on points in (i, j)-space, described below.

Definition. A set A of points in (i, j)-space is a Southeast set if the subsets $\{(a, b) \in A \mid a = i\}$ and $\{(a, b) \in A \mid b = j\}$ have at most one element for all i, j.

Definition. The Southeasting operation $Se(\cdot)$ produces a Southeast set A' from a set A as follows:

$$A' = Se(A) = \{(i, j) \in A \mid \{(a, j) \in A \mid a > i\} \text{ and } \{(i, b) \in A \mid b < j\} \text{ are empty}\}$$

The set $Se(A)$ can be obtained from A by considering every point $(i, j) \in A$ and removing any points that lie directly above or to the left of it (Figure 5.8).

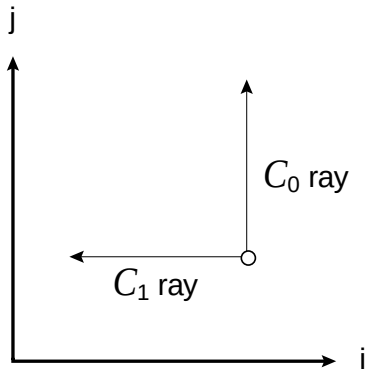


Figure 5.8: The Southeasting operation. Points are removed in the directions of the rays.

Proposition 5.3: *The correspondence graph C for a scene $\mathbf{S}$ with respect to $(\mathcal{C}_0, \mathcal{C}_1)$ can be generated by Southeasting the transformed scene $\mathbf{S}'$.*

Proof. We know that the correspondence graph C is a subset of the transformed scene $\mathbf{S}'$. It remains to determine which points in $\mathbf{S}'$ actually appear in both images. Fix i and consider the set of points $\mathbf{S}'_\mathbf{i} = \{(a, b) \in \mathbf{S}' \mid a = i\}$. From Proposition 5.1, these points lie on the same ray from $\mathcal{C}_0$ in (x, y)-space. The point p' with the smallest j coordinate is closest to $\mathcal{C}_0$ and is hence the only point along the ray that is imaged by $\mathcal{C}_0$. Therefore, the points in $\mathbf{S}'_\mathbf{i}$ with larger j coordinates than p' are not retained in the correspondence graph. Similarly, for fixed j, consider the set $\mathbf{S}'_\mathbf{j} = \{(a, b) \in \mathbf{S}' \mid b = j\}$. These points lie on the same ray from $\mathcal{C}_1$ in (x, y)-space, and the only point that is retained in the correspondence graph is that point q' with the largest i coordinate.

The operation described above is simply the Southeasting of the set $\mathbf{S}'$. By construction, the remaining elements in the Southeast set are precisely those points that appear in both cameras and hence this Southeast set is by definition the correspondence graph of $\mathbf{S}'$. ∎

Additionally, a partial converse to the above proposition is also true. That is:

Proposition 5.4: *Fix a pair of cameras $(\mathcal{C}_0, \mathcal{C}_1)$, normalized to the standard configuration in Section 5.3.1. Then any Southeast set of points in (i, j)-space is the correspondence graph of some physical scene, provided that conditions (5.3)–(5.6) are satisfied.*

Proof. From the proof of Proposition 5.3, it is clear that a correspondence graph must be a Southeast set. Furthermore, (5.3)–(5.6) give bounds on the possible locations of points in (i, j)-space that correspond to a physical scene. For any Southeast point (i, j) satisfying (5.3)–(5.6), a corresponding scene point in (x, y)-space can be obtained by intersection of the appropriate rays from $\mathcal{C}_0$ and $\mathcal{C}_1$, i.e. by applying the inverse transformation of (5.1)-(5.2). Hence if every point in a Southeast set C satisfies the bounds (5.3)–(5.6), a consistent scene $\mathbf{S}$ can be constructed for which C is the correspondence graph of $\mathbf{S}$ with respect to $(\mathcal{C}_0, \mathcal{C}_1)$. ∎

5.3.3 Examples of Correspondence Graphs

A scene $\mathbf{S}$ with a simple obstruction relative to two cameras is illustrated in Figure 5.9a. Figure 5.9b shows the scene transformed into (i, j)-space. The Southeasting process is applied in Figure 5.9c to obtain the correspondence graph in Figure 5.9d. This is the type of correspondence characteristic in stereo, where occlusions introduce discontinuities into the correspondence, but the correspondence remains monotonic.

It is instructive to compare the scene and graph in Figure 5.9 with those of Figure 5.10, in which the occluding segment (labeled 6) has been decreased in length and moved closer to the cameras. In this configuration, a part of the rear segment (labeled 3) behind the occluding segment is visible to both camera 0 and camera 1. The labeled line segments are projected to image plane $\mathcal{P}_0$ in the order 1-2-3-6-5, and to the image plane $\mathcal{P}_1$ in the order 1-6-3-4-5. Segments 3 and 6 appear in different orders in the projections; this reversal produces the phenomenon seen in the correspondence graph,

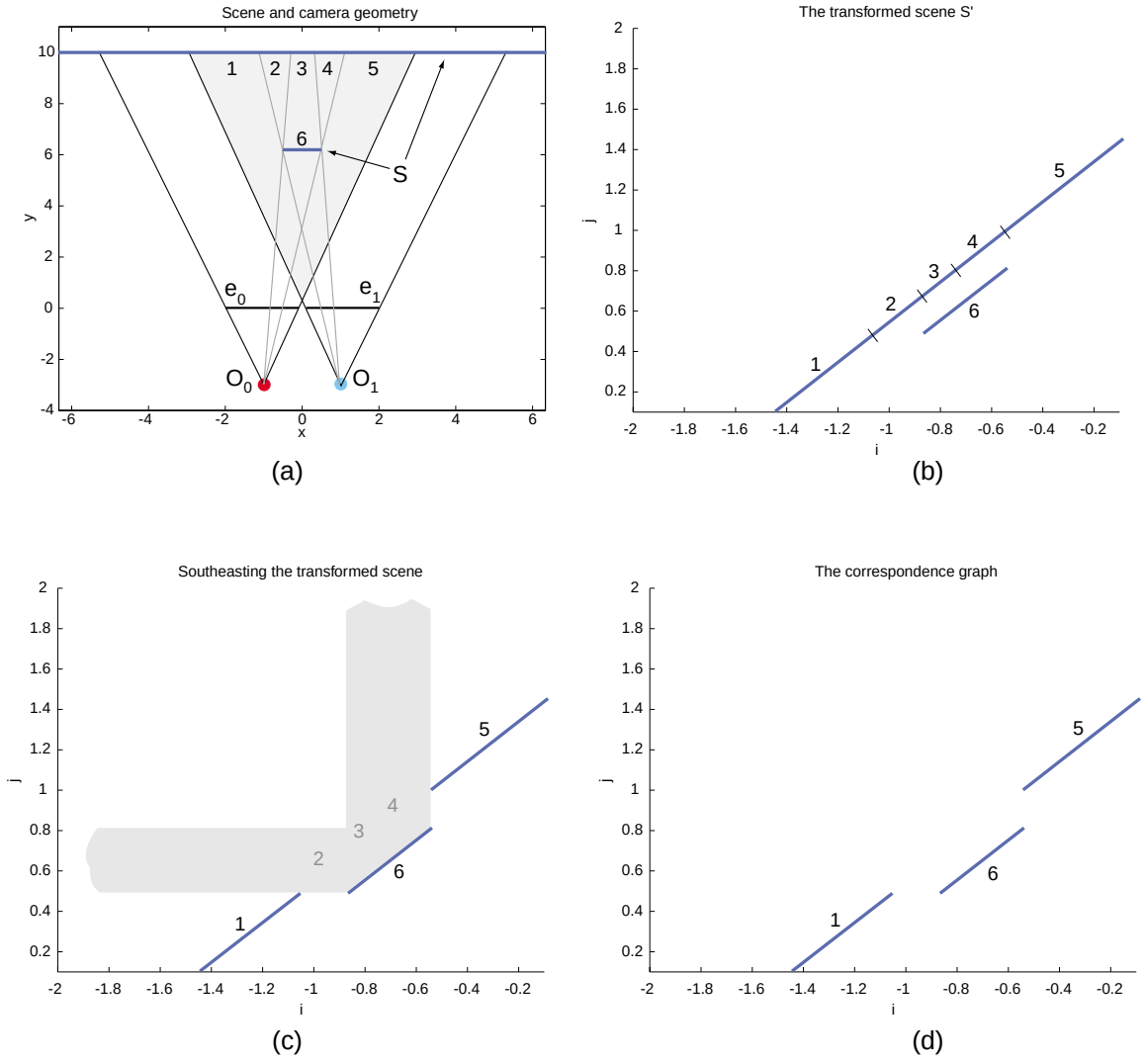


Figure 5.9: Example 1 (simple occlusion). (a) Scene S in (x, y)-space. (b) Transformed scene S' in (i, j)-space. (c) Southeasting the transformed scene. (d) Correspondence graph.

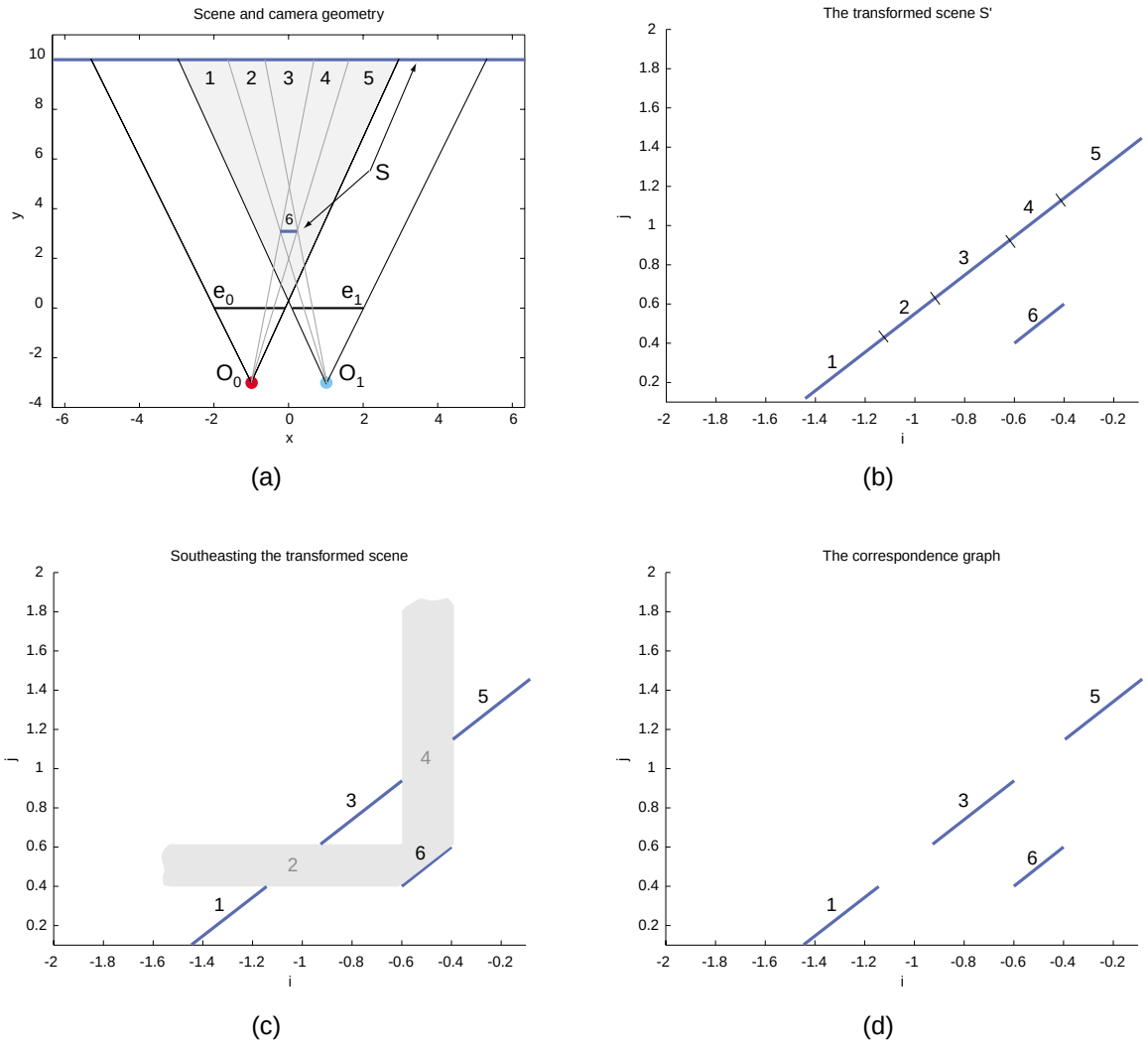


Figure 5.10: Example 2 (double-nail illusion). (a) Scene S in (x, y)-space. (b) Transformed scene S' in (i, j)-space. (c) Southeasting the transformed scene. (d) Correspondence graph.

Figure 5.10d. We can see this is the correspondence graph associated with the double-nail illusion in Section 5.2.

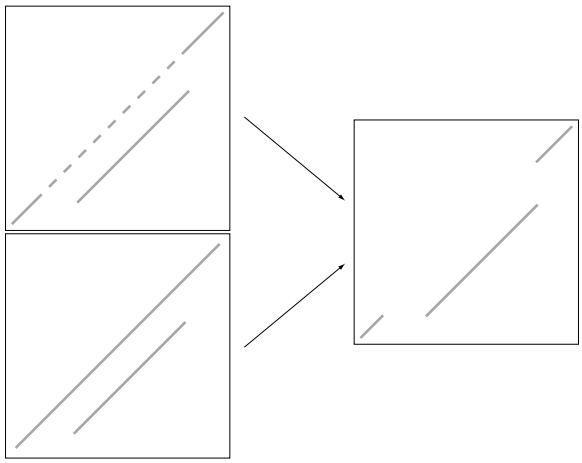


Figure 5.11: An N-piece and 2-piece input can both result in a 3-piece Southeast output.

There is no general rule that relates the number of connected components in A to the number in $Se(A)$ (see Figure 5.11). However, it is easy to see that the number of connected components in $Se(A)$ is at most three times the number of connected components in A, since Southeasting a single connected component can split another connected component into at most three pieces.

5.4 Estimating Correspondence Graphs

Though we fully characterized the structure of a correspondence graph in the previous section, it is not obvious how this result can be applied to real images. Our purpose in this section is to present an algorithm for estimating the correspondence graph for a conjugate pair of epipolar lines in a real image pair.

Algorithm 5.1: *Constructing correspondence graphs.*
Given the following:

- *A pair of images* $(\mathcal{I}_0, \mathcal{I}_1)$
- *Segmentation and matching of the foreground objects in each image*

- *A pair of conjugate epipolar lines* (ℓ_0, ℓ_1)

  Construct the correspondence graph for (ℓ_0, ℓ_1) *with the following four steps:*

1. *Estimate an initial correspondence within the segmented regions.*
2. *Estimate a global correspondence for the background regions.*
3. *Generate a basic correspondence graph by Southeasting the set of background and foreground correspondences.*
4. *Refine the pieces of each correspondence graph using a monotonic epipolar-line-based matching algorithm.*

We reviewed algorithms for estimating the fundamental matrix, and hence, the pairs of conjugate epipolar lines, in Section 3.2. We discuss in more detail the four steps of the algorithm below.

5.4.1 Step 1: Segmentation and Correspondence of Foreground Objects

The automatic detection, segmentation, and matching of objects in an image sequence is a very difficult problem, and beyond the scope of this thesis. In this chapter and the following ones, the segmentation and matching of the objects in the examples were obtained (somewhat painstakingly) by hand. Reviews of segmentation techniques are given in [11, 12].

Generally, determining objects that move independently of the background is greatly facilitated by multiple images, e.g. temporally adjacent frames of video. A general approach is to estimate the dominant motion of the background pixels induced by camera motion first, and analyze regions of pixels whose motion disagrees with this dominant motion [13, 14]. This can be viewed as a specific case of minimum-description-length layered motion estimation [15, 16] where almost all of the pixels are in one layer and the other layers have small spatial extent compared to the first one. Sharp segmentation and classification into the "correct" number of semantic objects can be difficult with these approaches. It is generally also very difficult to deal with occlusions.

Another approach to the segmentation problem comes from active contours. The general idea is to define a cost function related to the edges in an image, and to iteratively deform a closed curve or a series of closed curves defined on the image plane to achieve a minimal cost. If the cost is

defined appropriately, the minimizing curves conform to the contours of foreground objects in the image. This process generally operates on a single image, using intensity gradient information. The canonical references on such "snakes" are [17, 18]. A more recent active-contour-based work that seems very promising is [19]. The conditional-density-estimation algorithm discussed in [20, 21] also seems to be quite good at tracking moving objects in cluttered environments.

Once we possess the segmented and matched objects, we consider the intervals that are formed by intersecting the objects with the epipolar line pair (ℓ_0, ℓ_1). If the segmentation and epipolar geometry estimates are accurate, each interval corresponding to an object in ℓ_0 should have a match in ℓ_1. Correspondence between each interval pair is then initialized by assuming that points match up by linear interpolation between the boundaries.

A minor issue is how to deal with matching two objects with different numbers of connected components when cut by an epipolar line. We use the simple heuristic illustrated in Figure 5.12, by splitting up objects so that they have the same number of connected components along each epipolar line.

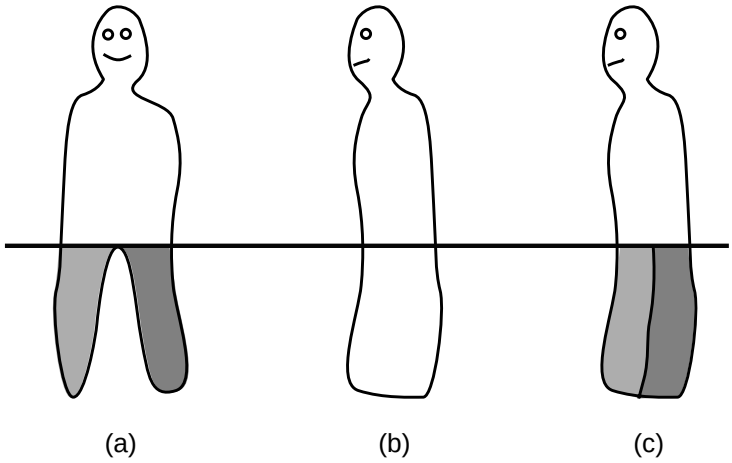


Figure 5.12: Matching a "two-legged" object (a) with a "one-legged" object (b). Below the line where the number of connected components differs, (b) is evenly split into two pieces to produce (c), which can now be matched with (a).

5.4.2 Step 2: Estimating Initial Global Correspondence for Background Pixels

Despite the battery of approaches to the correspondence problem described in Sections 3.5 and 5.1, obtaining a good estimate of the correct correspondence between a real image plane pair, even in the absence of occluding objects, is often problematic. Hence, any prior knowledge about the correct correspondence should be exploited.

For example, in the absence of prior information, an algorithm that matches points along conjugate epipolar lines may make the default assumption that the left and right edge points of the conjugate epipolar line pair correspond. For images in which the field of view of the cameras are very different (e.g. Figure 5.18), this is a poorly founded assumption. The correspondence algorithm can be applied with better results if the endpoints of the largest region that is projected onto both conjugate epipolar lines are estimated. When these starting and ending points of the matching path are well-estimated, the interiors of the delimited intervals can be more accurately matched.

In a certain class of real images (e.g. frames of sports video), much of the field of view of each camera is comprised by a planar surface. From Section 4.1, the relationship between the coordinates of correspondences that lie on this plane can be globally modeled by a projective transformation. Using a small set of correspondences that lie on the plane as input, the parameters of the projective transformation can be estimated using the techniques of Chapter 4. The estimated projective transformation can be applied to the entire image plane $\mathcal{I}_0$ to register the planar surfaces in the coordinate system of $\mathcal{I}_1$.

The outline of the warped image superimposed on $\mathcal{P}_1$ provides an estimate for the correspondences of points on the edges of $\mathcal{I}_0$ in $\mathcal{I}_1$. This is schematically illustrated in Figure 5.13. The filled quadrilaterals are the original image plane pair $(\mathcal{P}_0, \mathcal{P}_1)$ and their rectified counterparts $(\bar{\mathcal{P}}_0, \bar{\mathcal{P}}_1)$. The dotted-line image planes in the left column are the images of $\mathcal{P}_1$ in $\mathcal{P}_0$ and $\bar{\mathcal{P}}_0$ under the assumption that a projective transformation P maps the entire plane of $\mathcal{P}_0$ to $\mathcal{P}_1$. Similarly, the dotted-line image planes in the right column are the images of $\mathcal{P}_0$ in $\mathcal{P}_1$ and $\bar{\mathcal{P}}_1$ under this assumption. By composition of projective transformations, the map from $\bar{\mathcal{P}}_0$ to $\bar{\mathcal{P}}_1$ is given by $Q = HPG^{-1}$. Since the image planes are rectified, we know that if $(\bar{x}, \bar{y}) \in \bar{\mathcal{P}}_0$ maps to $(\bar{x}', \bar{y}') \in \bar{\mathcal{P}}_1$, then $\bar{y}' = \bar{y}$. That

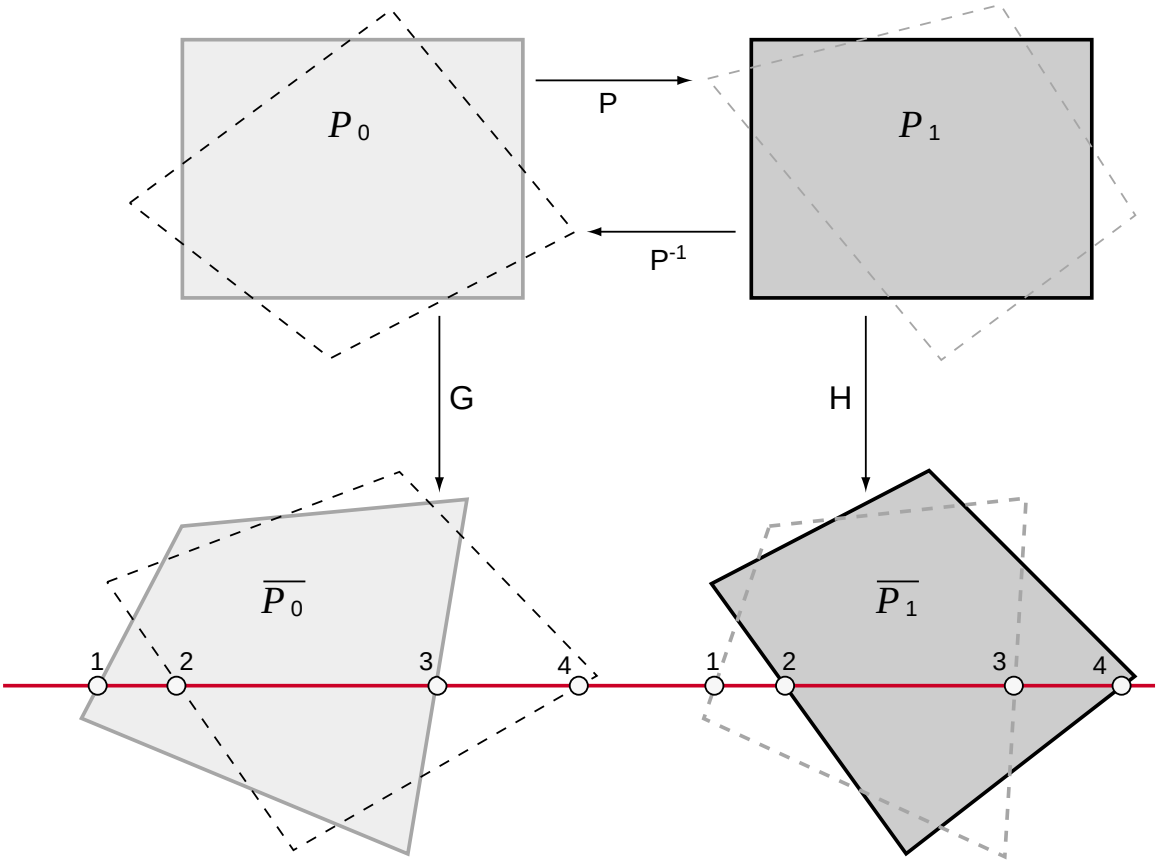


Figure 5.13: Using the planar surface to estimate initial correspondence.

is, if $Q = (A, b, c)$, then for all $(\bar{x}, \bar{y})$,

$$\bar{y} = \frac{a_{21}\bar{x} + a_{22}\bar{y} + b_2}{c_1\bar{x} + c_2\bar{y} + 1}$$

Among other things, this implies that $c_1 = c_2 = 0$, which means that

$$\bar{x}' = a_{11}\bar{x} + a_{12}\bar{y} + b_1$$

For a fixed $\bar{y}$ (e.g. the horizontal line in Figure 5.13), this means that $\bar{x}'$ is a linear function of $\bar{x}$. In this example, this means that we can initially estimate that the points along line $\bar{y}$ that appear in both images lie between the points labeled 2 and 3, and that points between these two correspond by linear interpolation. There are five other cases, depending on where the edges of the image planes lie with respect to each other in each line. We note that non-rectified image planes would not enjoy this linear interpolation property.

In the case where there are multiple planar surfaces in the scene, several projective transformations can be estimated to estimate the initial background correspondence. We discuss this issue further in Section 7.3.

5.4.3 Step 3: Generating the Basic Correspondence Graph

Once we have estimates for the background and foreground correspondences, we can construct a correspondence graph with the correct topology for each pair of conjugate epipolar lines, by Southeasting the collection of foreground pieces with the background piece.

5.4.4 Step 4: Refining each Monotonic Piece

Since by construction, each piece of the correspondence graph is monotonic, a correspondence algorithm that assumes monotonicity can be applied to each piece of the graph independently. If the estimates for the segmentation and background correspondences are sufficiently accurate, the refinement step may be constrained to select a matching path that lies in a nearby neighborhood of the initial matching path. Any of the algorithms from Section 5.1 can be used for the refinement.

5.5 Experimental Results

We now illustrate each of these steps in the estimation of the correspondence graph for a real image pair, and exhibit that the estimation problems involved at each step can be solved accurately. We will use the example image pair illustrated in Figure 5.14. These natural, outdoor images are from widely separated cameras viewing a soccer game.

Figure 5.14: Original image pair $(\mathcal{I}_0, \mathcal{I}_1)$.

The results of our hand-segmentation of the objects in $(\mathcal{I}_0, \mathcal{I}_1)$ is illustrated in Figure 5.15. We have segmented and matched three soccer players, the soccer ball, and the uprights of the soccer goal. Segmentation this precise would be difficult to obtain with an automatic algorithm, especially for the non-convex and transparent goalposts.

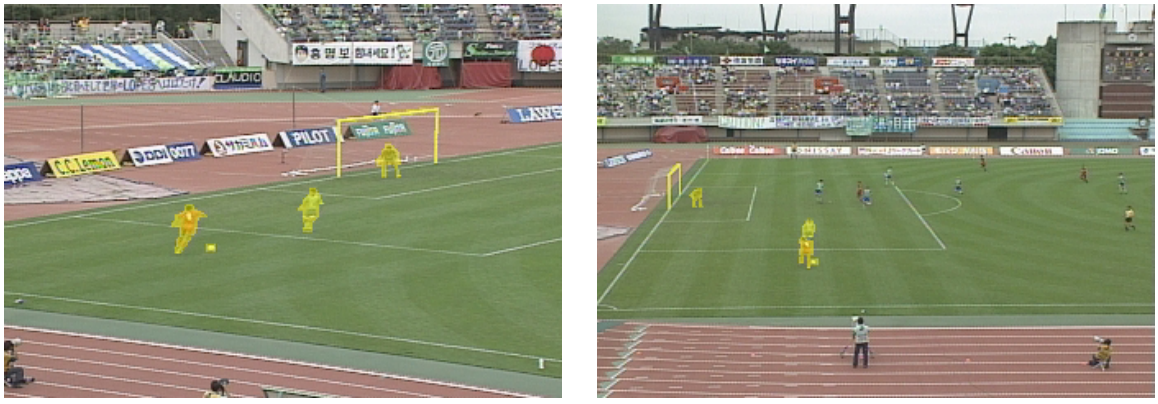


Figure 5.15: Image pair, with segmentation.

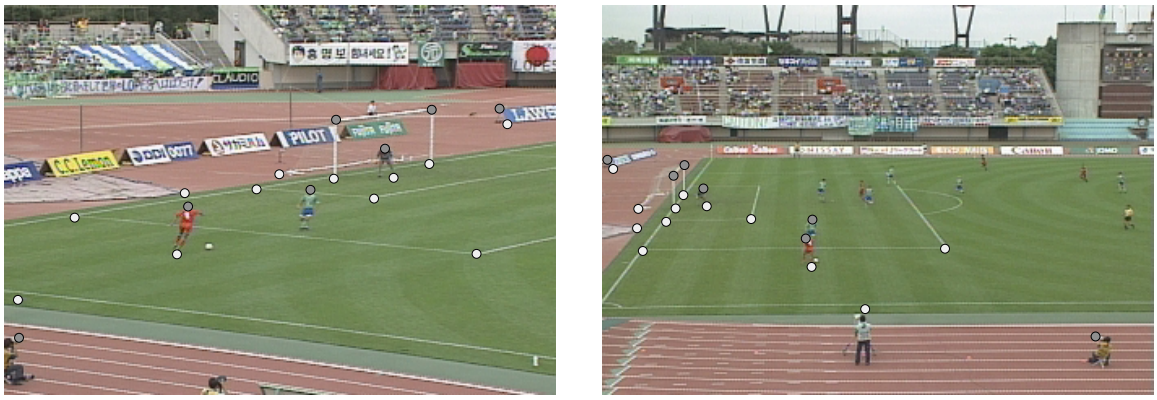


Figure 5.16: Image pair, with point correspondences used for estimation.

The input to the fundamental matrix estimation algorithm for our example is the set of 19 point correspondences illustrated in Figure 5.16. Of these, 12 lie on the plane of the soccer field (the lighter dots), and 7 lie off this plane (the darker dots). All 19 correspondences were used for the estimation of F, which in this case was determined to be:

$$\hat{F} = \begin{bmatrix} -4.3237 \times 10^{-8} & 2.3279 \times 10^{-6} & 3.2995 \times 10^{-5} \\ -1.1623 \times 10^{-5} & -9.8263 \times 10^{-6} & -0.01411 \\ 4.3254 \times 10^{-4} & 0.01117 & 0.9156 \end{bmatrix}$$

Sample epipolar lines corresponding to this estimate are displayed in Figure 5.17. It can be seen that the estimation is good, i.e. a point on an epipolar line in the left image has its match on the corresponding epipolar line in the right image. We can quantify the accuracy by considering the mean signed distance from each of the 19 points to its estimated epipolar line, which in this case is 0.3504, less than half a pixel width.

As discussed in Section 3.3, to implement algorithms on a computer, it is often convenient to rectify the images so that conjugate epipolar lines and aligned and horizontal. We take this approach here; Figure 5.18 illustrates the rectified soccer image pair. The estimated rectifying projective transformations are:

$$G = \begin{bmatrix} 0.9995 & 0.0305 & -0.0008 \\ -0.0305 & 0.9995 & 0 \\ 0.0008 & 2.5797 \times 10^{-5} & 1 \end{bmatrix}$$

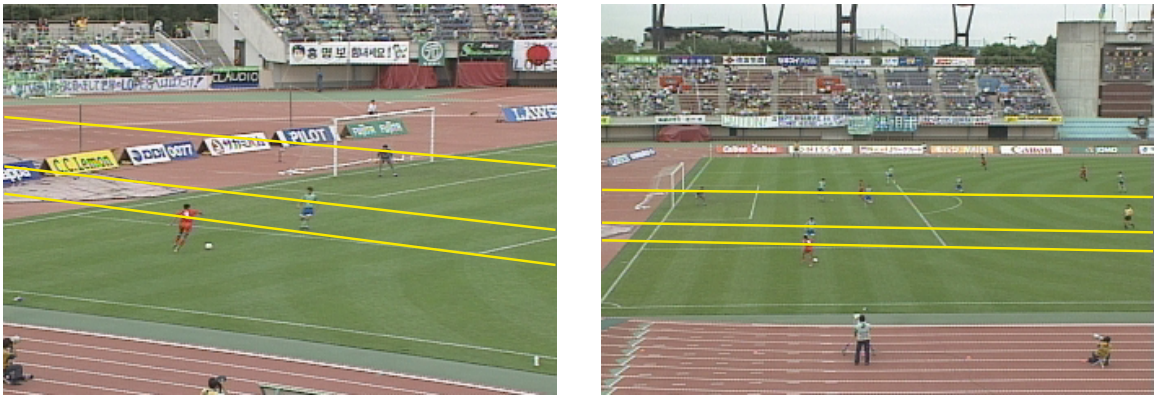


Figure 5.17: Image pair, with sample epipolar lines.

$$H = \begin{bmatrix} 0.9999 & -0.0119 & -0.0002 \\ -0.0030 & 1.2652 & -82.1072 \\ 0.0002 & -0.0008 & 1 \end{bmatrix}$$

The epipolar lines shown in Figure 5.17 are redisplayed, and are shown to be horizontal, which means that the estimation is good. That is, the correspondences on one row in the left image can be found on the same row in the right image. We can assess the quality of the rectifying pair by computing $H^{-T}FG^{-1}$ and comparing it to F^*. In this case,

$$\|H^{-T}FG^{-1} - F^*\|_2 = 1.2829 \times 10^{-14}$$

which is on the order of machine precision. Another measure of rectifier quality is to look at the mean difference of the y coordinates of the 19 rectified data points, which in this case is 0.4181, less than half a pixel width.

Now we are ready to proceed with the construction of the correspondence graph. The images in this example are well-suited to the planar assumption discussed in Section 5.4.2. The 12 points that lie on the soccer field (indicated by the lighter dots in Figure 5.16) are used for the estimation, and the projective transformation that warps the plane of the soccer field in $\mathcal{P}_0$ to the plane in $\mathcal{P}_1$ is estimated to be:

$$P = \begin{bmatrix} 0.6494 & 3.3081 & -432.5712 \\ 0.0238 & 0.7897 & 59.0927 \\ 0.0007 & 0.0004 & 1 \end{bmatrix} \tag{5.7}$$

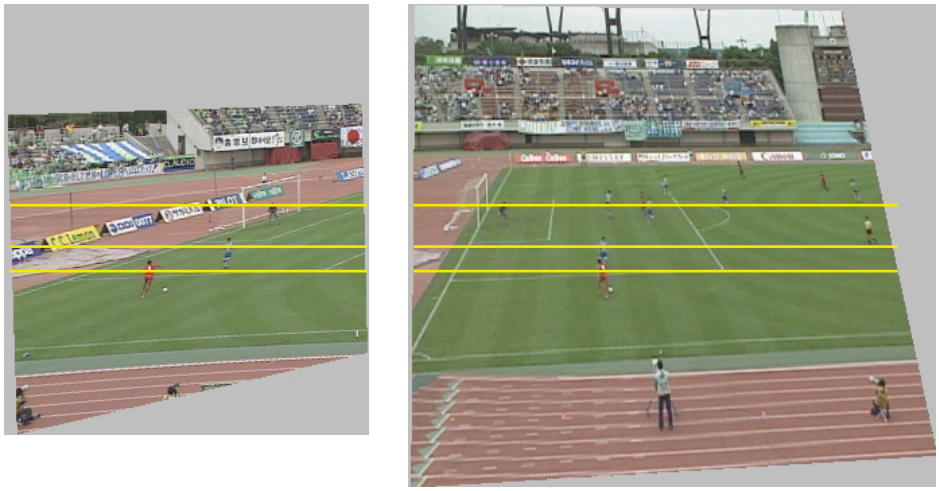


Figure 5.18: Rectified image pair, with sample epipolar lines.

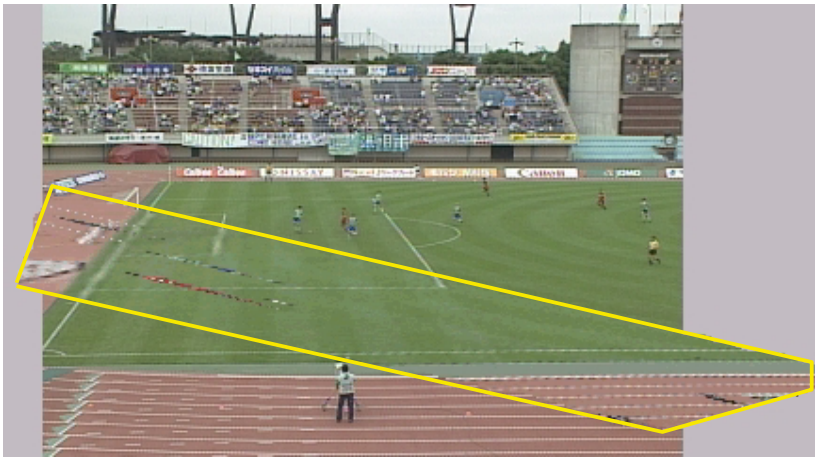


Figure 5.19: Registration of planar surface in soccer images.

Figure 5.19 shows the result of warping $\mathcal{P}_0$ by P to produce a new image $\tilde{\mathcal{P}}_0$, and overlaying this new image onto $\mathcal{P}_1$. Only the region of $\tilde{\mathcal{P}}_0$ containing the soccer field is displayed; of course, points that lie off of the planar surface (e.g. the soccer players) are distorted and registered incorrectly. However, by comparing the continuity of the images across the thick lines delimiting $\tilde{\mathcal{P}}_0$, we can see that the registration of the planar surface is accurate. The mean error in the projective transformation fit to the 12 data points is 0.9731, less than one pixel width.

At this point the foreground objects can be Southeasted against the background to create the

basic topology of the correspondence graph for each conjugate epipolar line pair.

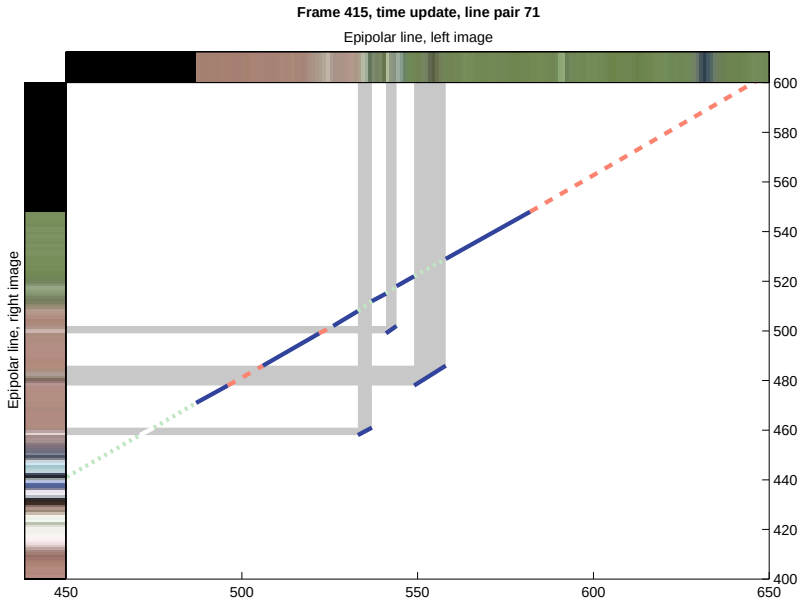


Figure 5.20: Correspondence graph (basic topology), line 71, enhanced to show the Southeasting operation. The thin lines are the goalposts; the thicker line is the goalie.

Figures 5.20-5.22 show the result for the three pairs of epipolar lines in Figure 5.17. The Southeasting operation is shown graphically by the "shadows" in the figures; the correspondence graph itself is comprised by the solid line segments that are unshadowed. The dashed lines indicate regions visible in $\mathcal{I}_0$ but not in $\mathcal{I}_1$ because they are occluded or lie outside the field of view. The dotted lines indicate similar regions visible in $\mathcal{I}_1$ but not in $\mathcal{I}_0$. Figure 5.20 indicates the kind of complexity that can occur in real correspondence. This is the epipolar line pair that cuts across the two goalposts and the goalie. In one image, the goalie stands between the goalposts; in the other he stands completely to one side. We can see that correspondence along this epipolar line pair is decidedly non-monotonic. The Southeast correspondence graph gives a physically consistent estimate of which regions in the line pair can correspond.

Figures 5.23-5.25 show the results of refining the monotonic pieces of each correspondence graphs of Figures 5.20-5.22. In our experiments we use the Ohta and Kanade algorithm on each monotonic piece of the correspondence graph. We can assume there are no occlusions, by construction of the correspondence graph. Here, we searched for the lowest-cost matching path that lies

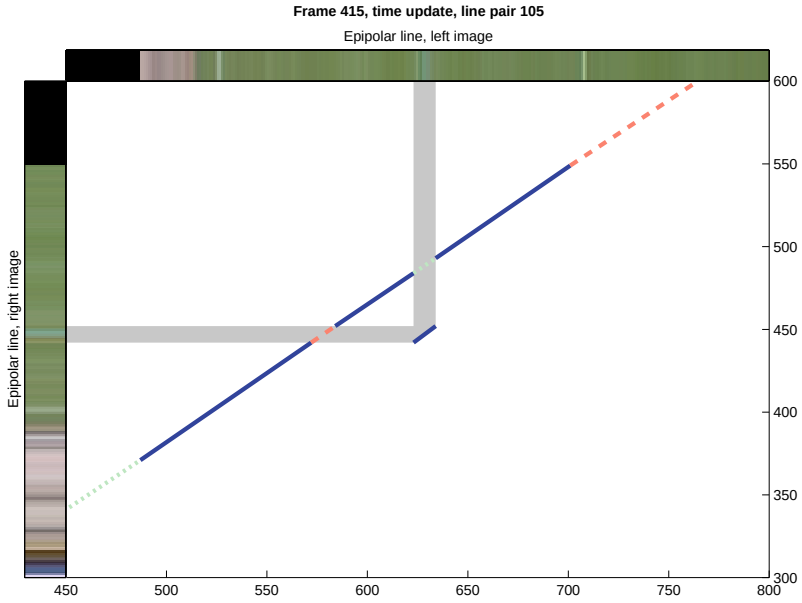


Figure 5.21: Correspondence graph (basic topology), line 105.

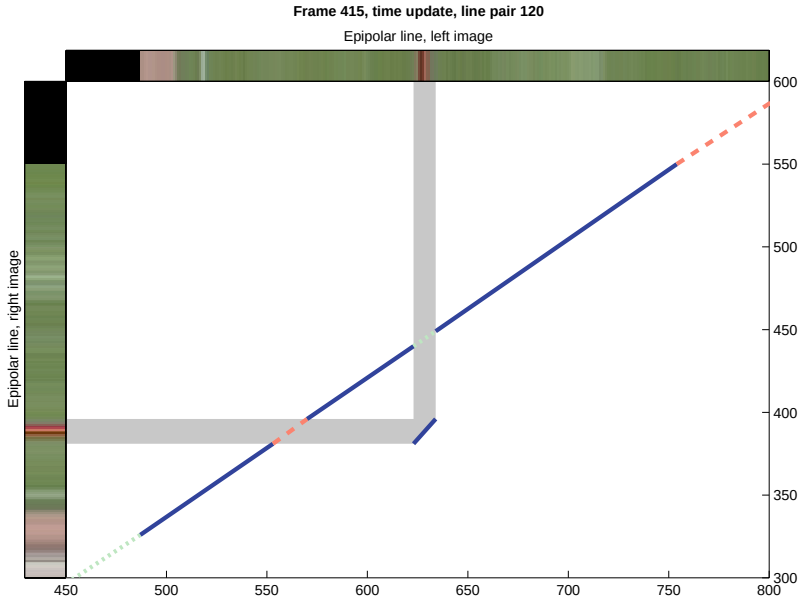


Figure 5.22: Correspondence graph (basic topology), line 120.

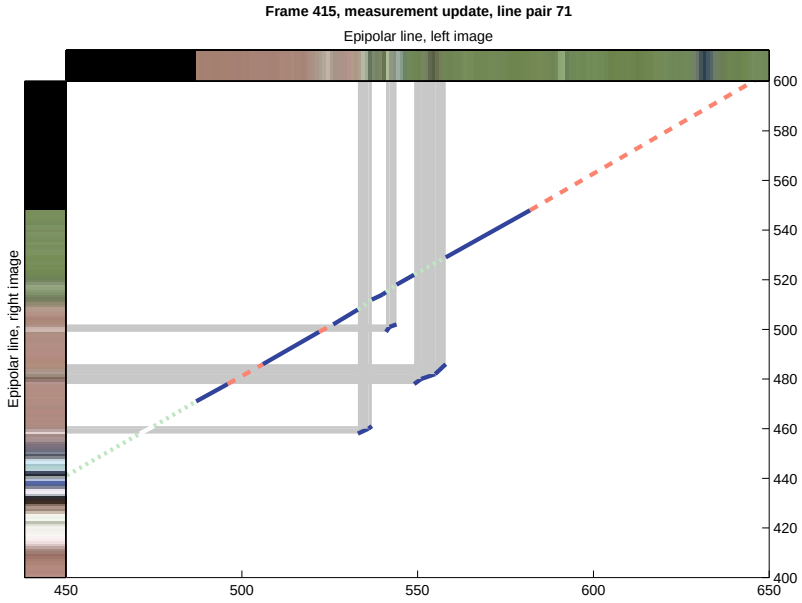


Figure 5.23: Correspondence graph (refined), line 71.

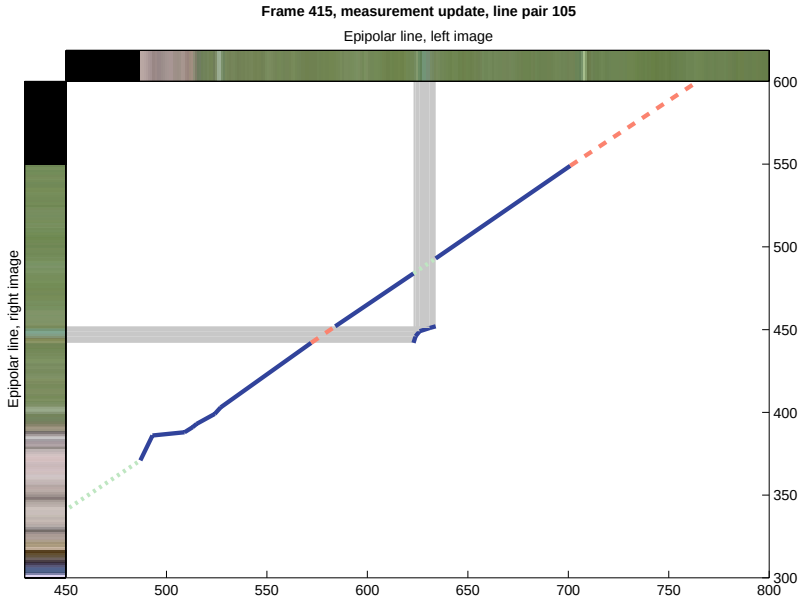


Figure 5.24: Correspondence graph (refined), line 105.

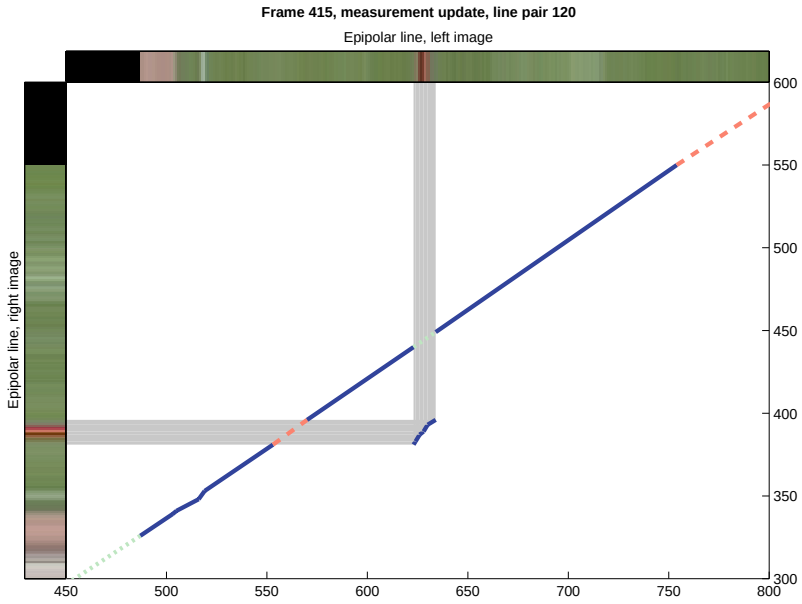


Figure 5.25: Correspondence graph (refined), line 120.

within 8 pixels of the basic correspondence graph.

We can see that correspondence on both the background and foreground objects is refined by the procedure, but that by construction, the topology of the refined correspondence graphs is the same as that of the basic correspondence graphs. The quality of the refined correspondence can be appreciated visually in Figures 6.4-6.6 from Chapter 6.

5.6 Conclusions

By ensuring that an estimated correspondence produces a valid (i.e. Southeast) correspondence graph, we are explicitly prevented from attempting to match pixels from regions that do not appear in both images, a pitfall of many correspondence algorithms.

However, the pieces of the graph removed by Southeasting need not be discarded. The lighter segments in Figures 5.23-5.25 can also be used to estimate the correct locations of regions not seen in both images. For example, given an object in $\mathcal{I}_1$, we can estimate which piece of $\mathcal{I}_0$ it occludes, simply by linearly interpolating between edges of correspondence graph pieces. We shall see an application to filling in "holes" in correspondence in Section 6.2.

The issue of obtaining good rectified images has not yet been satisfactorily solved. Seitz's method, which we currently use, can sometimes produce rectified images that are extremely warped, as in Figure 5.26. It is desirable to solve the problem of finding a pair of rectifying projective transformations that minimizes the distortion of the warped images for a given fundamental matrix estimate. Of course, rectification is just a computational convenience, and a good implementation should avoid unnecessary resampling of images and operate as much as possible on unsampled image data.

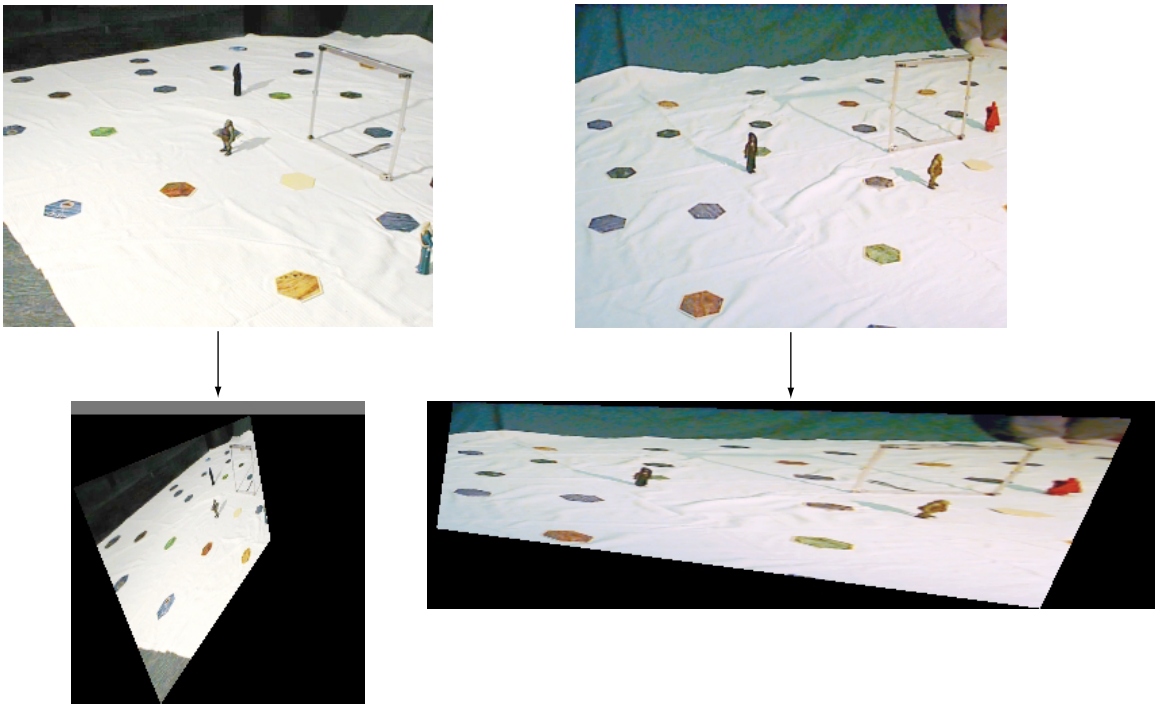


Figure 5.26: An unsatisfactory pair of rectified images.

Exploring the relationship of correspondences in more than two images is a natural extension of this research, and much work in this regard has already been done by Faugeras [22, 23] and Shashua [24, 25]. We have shown that even in the case when the camera centers are colinear, for three conjugate epipolar lines parametrized by (i, j, k), there is no operation analogous to Southeasting in (i, j, k)-space by which points in a set $\mathbf{S}'$ are removed along paths that are independent of the starting point. There are some simplifications when the focal lengths are all the same, but in general,

it seems that a different approach may be required for higher dimensions.

5.7 References

[1] O.D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, 1993.

[2] R. Radke, V. Zagorodnov, S. Kulkarni and P. Ramadge. Estimating Correspondence in Digital Video. In *Proc. ITCC 2001*, Las Vegas, Nevada, April 2001.

[3] M. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[4] I.J. Cox, S.L. Hingorani, and S.B. Rao. A Maximum Likelihood Stereo Algorithm. *Computer Vision and Image Understanding*, Vol. 63, No. 3, pp. 542–567, May 1996.

[5] Y. Ohta and T. Kanade. Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. *IEEE PAMI*, Vol. 7, No. 2, pp. 139–154, March 1985.

[6] P.N. Belhumeur. A Bayesian Approach to Binocular Stereopsis. *International Journal of Computer Vision*, Vol. 19, No. 3, pp 237–260, 1996.

[7] H. Ishikawa and D. Geiger. Occlusions, Discontinuities, and Epipolar Lines in Stereo. In *Proc. ECCV '98*, Freiburg, Germany, 1998.

[8] C. Tomasi and R. Manduchi. Stereo Matching as a Nearest-Neighbor Problem. *IEEE PAMI*, Vol. 20, No. 3, pp. 333–340, March 1998.

[9] R. Radke, P. Ramadge, S. Kulkarni, T. Echigo and S. Iisaku. Recursive Propagation of Correspondences with Applications to the Creation of Virtual Video. In *Proc. ICIP 2000*, Vancouver, Canada, September 2000.

[10] D. Sankoff and J. Kruskal, eds. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, 1983.

[11] N. R. Pal and S. K. Pal. A Review on Image Segmentation Techniques. *Pattern Recognition*, vol. 26, pp. 1277–1294, 1993.

[12] M.G. Strintzis and S. Malassiotis. Object-Based Coding of Stereoscopic and 3-D Image Sequences. *IEEE Signal Processing Magazine*, vol. 16, no. 3, pp. 14–28, May 1999.

[13] N. Diehl. Object-Oriented Motion Estimation and Segmentation in Image Sequences. *Signal Processing, Image Communication*, vol. 3, pp. 23–56, February 1991.

[14] M. Hoetter and R. Thoma. Image Segmentation Based on Object Oriented Mapping Parameter Estimation. *Signal Processing*, vol. 15, no. 3, pp. 315–334, October 1988.

[15] S. Ayer and H.S. Sawhney. Layered Representation of Motion Video using Robust Maximum-Likelihood Estimation of Mixture Models and MDL Encoding. In *Proc. ICCV '95*, June 1995.

[16] H. Zheng and S.D. Blostein. Motion-Based Object Segmentation and Estimation Using the MDL Principle. *IEEE Trans. on Image Processing*, vol. 4, no. 9, pp. 1223–1235, 1995.

[17] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, vol. 1, pp. 321–331, 1988.

[18] C. Xu and J.L. Prince. Snakes, Shapes, and Gradient Vector Flow. *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 359–369, March 1998.

[19] A. Tsai, A. Yezzi, Jr., and A.S. Willsky. Curve Evolution, Boundary-Value Stochastic Processes, the Mumford-Shah Problem, and Missing Data Applications. In *Proc. ICIP 2000*, Vancouver, Canada, September 2000.

[20] M. Isard and A. Blake. CONDENSATION – Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[21] J. MacCormick and A. Blake. A Probabilistic Exclusion Principle for Tracking Multiple Objects. *International Journal of Computer Vision*, vol. 39, no. 1, pp. 57–71, 2000.

[22] O.D. Faugeras and B. Mourrain. On the Geometry and Algebra of the Point and Line Correspondences Between *N* Images. *Proc. ICCV '95*, pp. 951–956, June 1995.

[23] O.D. Faugeras. What Can be Seen in Three Dimensions With an Uncalibrated Stereo Rig? *Proc. ECCV '92*, pp. 563–578, May 1992.

[24] A. Shashua. Trilinear Tensor: The Fundamental Construct of Multiple-view Geometry and its Applications. In *International Workshop on Algebraic Frames For The Perception Action Cycle (AFPAC)*, Kiel, Germany, 1997.

[25] A. Shashua and M. Werman. On the Trilinear Tensor of Three Perspective Views and its Underlying Geometry. In *Proc. ICCV '95*, June 1995.

Chapter 6

Virtual Video

The past six years have witnessed an explosion of techniques for generating what can be termed "virtual views". Given a set of images of the same scene at the same time taken by different cameras, the virtual view problem is to synthesize an image from the viewpoint of a camera not in the original set. The problem can be generalized to that of synthesizing video from the viewpoint of a moving camera, given a set of real video sequences.

Virtual video has made its way into the public eye thanks to commercials (e.g. The Gap's "Khakis Swing" commercial), movies (e.g. The Matrix), and televised sporting events (e.g. Superbowl XXXV). The typical effect is of a camera navigating through a frozen or slowed-down scene. Such effects are created using a highly specialized camera rig with tens or hundreds of cameras positioned along the desired camera path. Each "virtual image" is either a real image from one of the closely spaced cameras, or is interpolated from an adjacent pair of real images using small-baseline algorithms. These techniques can be categorized as hardware solutions to the virtual video problem. Related techniques from computer graphics include the light field [1] and lumigraph [2], which require hundreds or thousands of images, huge amounts of storage, and copious processing time to synthesize new views of a scene.

Our interest is in synthesizing physically correct virtual images, that is, images that are created with well-founded geometric principles instead of ad-hoc techniques. Furthermore, we would like to synthesize virtual images in situations where strong calibration (knowledge of 3-D location and orientation) of the source cameras is unavailable, and many cameras are not required. In this chapter, we will confine our attention to the case when images from exactly two source cameras are available.

Other researchers (e.g. Laveau and Faugeras [3] and Avidan and Shashua [4]) have discussed using images from more than two cameras to create virtual still images. A recent paper by Ma et. al [5] characterizes the set of physically correct virtual images that can be obtained from a finite number of real images. In the case of video, however, it can be quite difficult to obtain two synchronized video sequences of the same scene, much less three or more.

We begin in Section 6.1 by reviewing view morphing, a general method for synthesizing an intermediate virtual image whose optical center lies on the line through the optical centers of the two source cameras. We will illustrate our examples in the text using the view morphing algorithm exclusively. There are many other approaches to image-based view synthesis in the computer graphics literature (e.g. McMillan and Bishop [6]). However, virtual view synthesis algorithms share the trait that they depend fundamentally on estimating a dense correspondence between the source image planes.

We demonstrate in Section 6.2 how the estimate of a set of correspondence graphs between a wide-baseline image pair can be used to generate compelling virtual images of a scene. Unlike many view synthesis results that incorporate correspondence algorithms using a monotonicity assumption, we are able to display a much richer class of virtual images here.

Next we address the virtual video problem. Aside from the hardware solutions discussed above, the only other type of virtual video we know of prior to this work was created by moving a virtual camera through a static scene, so that objects seem to be frozen in time. In contrast, here we create true virtual video from a pair of source video sequences, in the sense that the virtual video evolves dynamically along with the scene.

One naïve solution to the virtual video problem is to treat it as an independent sequence of virtual view problems over the length of the source videos. However, this approach is prohibitively time-consuming, since estimating dense correspondence between an image pair, especially a widely-spaced one, generally requires human intervention. More importantly, the independent problems do not exploit the temporal regularity of the input video. That is, assuming that the motion of the cameras and scene objects is small, we expect that the correspondence required to synthesize virtual images at adjacent frames is similar.

In Section 6.3 we present the main contribution of the chapter, a framework for the recursive

propagation of correspondences between frames of two video sequences. The propagation consists of a time update step and a measurement update step. The time update depends only on the dynamics of the source cameras, while the measurement update can be tailored to any member of a general class of image correspondence algorithms. Using these results, the correspondence estimate relating each frame pair can be propagated and updated in a fraction of the time required to estimate correspondences anew at every frame. While virtual video is our motivating application, the recursive correspondence propagation framework applies to any two-camera video application in which correspondence is difficult and prohibitively time-consuming to estimate by processing frame pairs independently.

We demonstrate our experimental results on real test video from a natural outdoor scene in Section 6.4. The scene is complex, with many moving objects, yet the synthetic virtual video looks realistic and conveys a convincing 3-D effect. The user need only provide a small set of point matches in the first frame pair, and an algorithm to segment and track moving objects in the scene. A shorter version of this work originally appeared in [7].

6.1 Review of View Morphing

6.1.1 View Interpolation

The first result we present is called view interpolation. We consider the camera configuration of Figure 6.1, in which the two image planes $\mathcal{P}_0$ and $\mathcal{P}_1$ are parallel to each other and to the baseline.

Without loss of generality, we can fix $O_0 = (0, 0, 0)$ and $O_1 = (1, 0, 0)$, and take R_0 and R_1 to be the identity I. The camera matrices Π_0 and Π_1 are then given by:

$$\Pi_0 \;=\; \begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

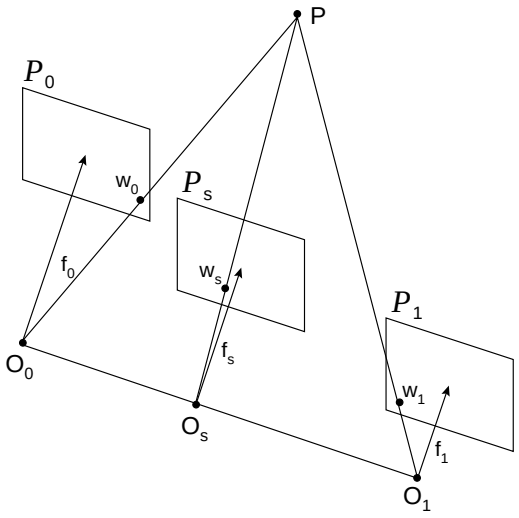


Figure 6.1: View interpolation.

$$\Pi_1 = \begin{bmatrix} f_1 & 0 & 0 & -f_1 \\ 0 & f_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

If we consider a correspondence[1] $(w_0, w_1) \in \mathcal{P}_0 \times \mathcal{P}_1$ induced by a scene point $P = (X, Y, Z)$, then Chen and Williams [8] noted that

$$\begin{aligned} (1-s)\begin{bmatrix} w_0 \\ 1 \end{bmatrix} + s\begin{bmatrix} w_1 \\ 1 \end{bmatrix} &= (1-s)\frac{1}{Z}\Pi_0\begin{bmatrix} P \\ 1 \end{bmatrix} + s\frac{1}{Z}\Pi_1\begin{bmatrix} P \\ 1 \end{bmatrix} \\ &= \frac{1}{Z}\begin{bmatrix} (1-s)f_0 + sf_1 & 0 & 0 & -sf_1 \\ 0 & (1-s)f_0 + sf_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} P \\ 1 \end{bmatrix} \\ &= \frac{1}{Z}\Pi_s\begin{bmatrix} P \\ 1 \end{bmatrix} \end{aligned} \tag{6.1}$$

[1] Here we have made a slight shift in notation, from the (w, w') that we used in the previous chapters to (w_0, w_1). The reason is that we are now viewing $\mathcal{C}_0$ and $\mathcal{C}_1$ as the endpoints of a "line segment" of cameras parametrized by the subscript.

When $s \in [0,1]$, the matrix Π_s corresponds to a camera $\mathcal{C}_s$ with

$$O_s = \left(\frac{sf_1}{(1-s)f_0 + sf_1}, 0, 0\right) \tag{6.2}$$

$$f_s = (1-s)f_0 + sf_1 \tag{6.3}$$

$$R_s = I \tag{6.4}$$

Then if w_s is the projection of P by $\mathcal{C}_s$,

$$\begin{bmatrix} w_s \\ 1 \end{bmatrix} = (1-s)\begin{bmatrix} w_0 \\ 1 \end{bmatrix} + s\begin{bmatrix} w_1 \\ 1 \end{bmatrix}$$

so we have

$$w_s = (1-s)w_0 + sw_1 \tag{6.5}$$

Hence, interpolating the image coordinates of the projections of P is the same as projecting P onto the image plane of an interpolated (in the sense of (6.2)-(6.4)) camera. The fact that the origin of the camera is a nonlinear function of s is slightly disagreeable, but we shall generalize the view interpolation result considerably in the next section.

The basic and important result (6.5) shows that a new projection of the scene can be obtained without knowledge of the three-dimensional locations of cameras or scene points. Provided that given any point $w_0 \in \mathcal{P}_0$, its correspondence $w_1 \in \mathcal{P}_1$ can be estimated, the correspondence $w_s \in \mathcal{P}_s$ can be computed through (6.5). Algorithms for estimating correspondence at a very fine level between an image pair have been extensively studied (see Section 3.5 and Chapter 5), and as a result, compelling and physically "correct" intermediate images of a scene can be synthesized without any three-dimensional modeling. Chen and Williams called this result view interpolation.

Incidentally, the matrix Π_s of (6.1) represents a physical camera provided $f_s > 0$. When $f_0 \geq f_1$, as sketched in Figure 6.1, this means that the view interpolation formulas represent a physical camera whenever $s \leq \frac{f_0}{f_0 - f_1}$. That is, the projection onto any image plane "beyond" $\mathcal{P}_0$ (i.e. $s < 0$) and some image planes "beyond" $\mathcal{P}_1$ (i.e. $s \in \left(1, 1 + \frac{f_1}{f_0 - f_1}\right)$) can be extrapolated. In particular, if $f_0 = f_1$, then the projection onto any $\mathcal{P}_s$ can be computed.

6.1.2 View Morphing

Here we show how the view interpolation result can be extended to a more general class of virtual images.

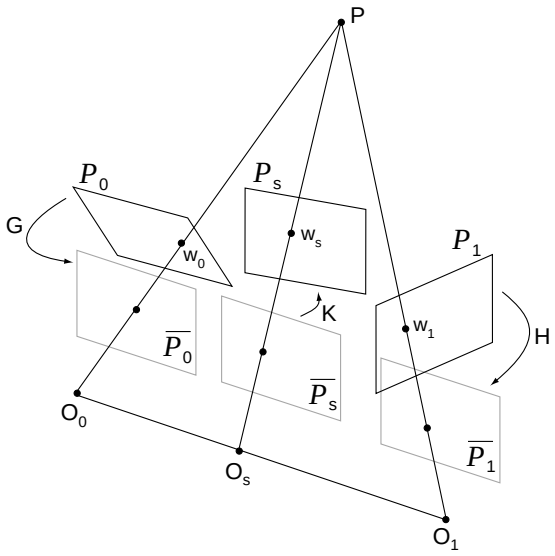


Figure 6.2: View morphing.

As we reviewed in Section 3.3, when the epipolar geometry is known between an image plane pair $(\mathcal{P}_0, \mathcal{P}_1)$, there are several methods for selecting a pair of rectifying projective transformations (G, H). The transformations G and H represent underlying rotations of the cameras $\mathcal{C}_0$ and $\mathcal{C}_1$ to new cameras $\bar{\mathcal{C}}_0$ and $\bar{\mathcal{C}}_1$ that have the same optical centers. The corresponding image planes $\mathcal{P}_0$ and $\mathcal{P}_1$ are rotated to new image planes $\bar{\mathcal{P}}_0$ and $\bar{\mathcal{P}}_1$. such that after rectification, $\bar{\mathcal{P}}_0$ and $\bar{\mathcal{P}}_1$ are parallel to each other and to the camera baseline, with their epipolar lines aligned and coincident with lines of constant y. Since this is precisely the configuration for view interpolation discussed above, a new view can be synthesized from the perspective of a camera $\bar{\mathcal{C}}_s$ whose origin O_s lies at the s-way point between O_0 and O_1, and whose image plane $\bar{\mathcal{P}}_s$ is parallel to $\bar{\mathcal{P}}_0$ and $\bar{\mathcal{P}}_1$.

The image plane $\mathcal{P}_s$ of an arbitrary camera $\mathcal{C}_s$ with origin O_s can be obtained from $\bar{\mathcal{P}}_s$ by application of an appropriate projective transformation K that effectively rotates the image plane from $\bar{\mathcal{P}}_s$ to $\mathcal{P}_s$. Then if (w_0, w_s, w_1) are the projections of a scene point P onto the image planes

$(\mathcal{P}_0, \mathcal{P}_s, \mathcal{P}_1)$, we have the central equation

$$w_s = K^{-1}((1-s)G(w_0) + sH(w_1)) \tag{6.6}$$

This result, illustrated in Figure 6.2, was first obtained by Seitz and Dyer [9], who called the process view morphing.

Since the focal lengths of the cameras $\bar{\mathcal{C}}_0$ and $\bar{\mathcal{C}}_1$ are equal by construction, the view interpolation equation (6.5) is valid for any value of s. Hence, we can use (6.6) to construct the projection onto the image plane of any camera whose optical center lies on the line through O_0 and O_1, not just cameras with $s \in [0, 1]$. This was not mentioned in Seitz's original work, though the extrapolation property was recognized by others, e.g. Scharstein [10].

6.2 Experimental Results: Virtual Images from Wide-Baseline Stills

We now return to the pair of test images illustrated in Figure 6.3, which is the same example from Chapter 5. In Section 5.5, we estimated the correspondence graph for each pair of conjugate epipolar lines using our proposed algorithm, and thus we possess a dense correspondence between the image planes $\mathcal{P}_0$ and $\mathcal{P}_1$. This is all we need to create virtual views of the same scene.

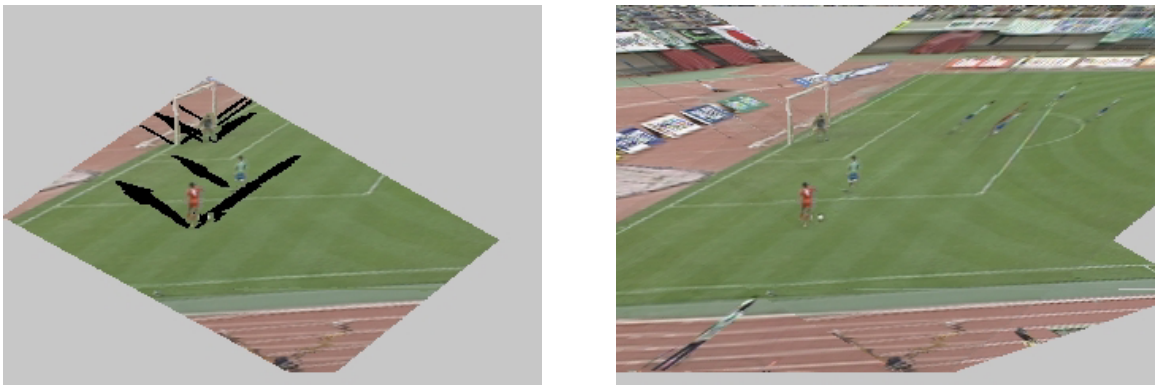
Figure 6.3: Original image pair $(\mathcal{I}_0, \mathcal{I}_1)$**.**

The view morphing equation (6.6) is a statement only about the positions of corresponding points in the image planes, not about their colors. Here we proceed from the Lambertian assumption that scene points have the same color regardless of the viewing angle, and that the color of an image

point is the same as the color of a single corresponding scene point. To compensate for deviations from these assumptions in real images, we will color points in the virtual images by a weighted average:

$$\mathcal{I}_s(w_s) = (1-s)\mathcal{I}_0(w_0) + s\mathcal{I}_1(w_1) \tag{6.7}$$

We use the choice of rectifying projective transformations suggested by Seitz [9] and detailed in Section 3.3. In the examples of this chapter, we will fix the postwarping transformation K to be the identity. However, one of our main interests in Chapter 7 will be the estimation of a projective transformation that aligns a virtual image with a real image as well as possible.



(a) (b)

Figure 6.4: Synthesized virtual image $\hat{\mathcal{I}}_s$ at $s = 0.5$. (a) no filling of occluded regions. (b) filling of occluded regions by planar assumption.

For each pair of conjugate epipolar lines (corresponding to rows of the rectified images) we estimated the correspondence graph, as described in Section 5.4 and illustrated in Figures 5.23-5.25. Each scene point that is visible in both images is rendered on the virtual image plane using the view morphing equations (6.6) and (6.7) with $s = 0.5$. Pixels are rendered in the order of decreasing disparity, that is, back to front. The result is illustrated in Figure 6.4a. While the rendered pixels appear realistic, the eye is drawn to two striking artifacts:

1. The black regions in the image plane that correspond to pixels visible in only one of the images $(\mathcal{I}_0, \mathcal{I}_1)$. For example, each soccer player has two "shadows" corresponding to the piece of the soccer field that was occluded from each perspective.

2. The limited extent of the virtual image compared to the originals. This is caused by the relatively small region visible in both image planes.

In this example, we can alleviate both of the above problems by supposing that the background is a planar surface.[2] Consider a scene point P that is visible in $\mathcal{I}_0$ at w_0 but is not visible in $\mathcal{I}_1$. We compute an estimate $\tilde{w}_1$ that is the image of w_0 under the projective transformation induced by the planar surface. Then $(w_0, \tilde{w}_1)$ can be treated as a correspondence, and the projection w_s of P in $\mathcal{P}_s$ can be estimated as

$$w_s = (1 - s)G(w_0) + sH(\tilde{w}_1)$$

However, in this case we should only use the color of the point in the image where it is visible, that is,

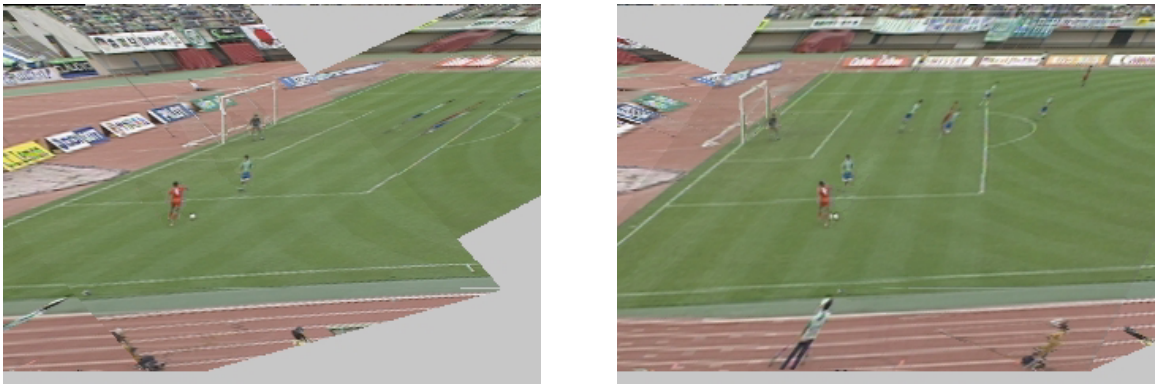
$$\mathcal{I}_s(w_s) = \mathcal{I}_0(w_0)$$

We take a similar tactic for points that are visible in $\mathcal{I}_1$ but not in $\mathcal{I}_0$. Of course, there may be regions that are visible in neither image due to occlusions by multiple objects. A correspondence estimate $(\tilde{w}_0, \tilde{w}_1)$ can be obtained for such a point from the planar assumption, but there is no color information for this point. In this case, we can interpolate the colors from either side of the missing piece, or use a default color. We note that the correspondences of occluded points induced by the planar assumption are displayed as dotted and dashed lines in Figures 5.23-5.25.

The result of filling in occluded regions by the planar assumption is illustrated in Figure 6.4b. Since the planar assumption is valid over many occluded pixels, the virtual image is much more realistic. Distortion is visible in several regions where the planar assumption is invalid, such as the stands in the upper left corner, and the soccer players at the upper right. However, the virtual image is a convincing rendition of the scene from a viewpoint that is halfway between the unknown optical centers of the original cameras. Interpolated views with $s = 0.25$ and $s = 0.75$ are illustrated in Figure 6.5, and extrapolated views with $s = -0.5$ and $s = 1.5$ are illustrated in Figure 6.6.

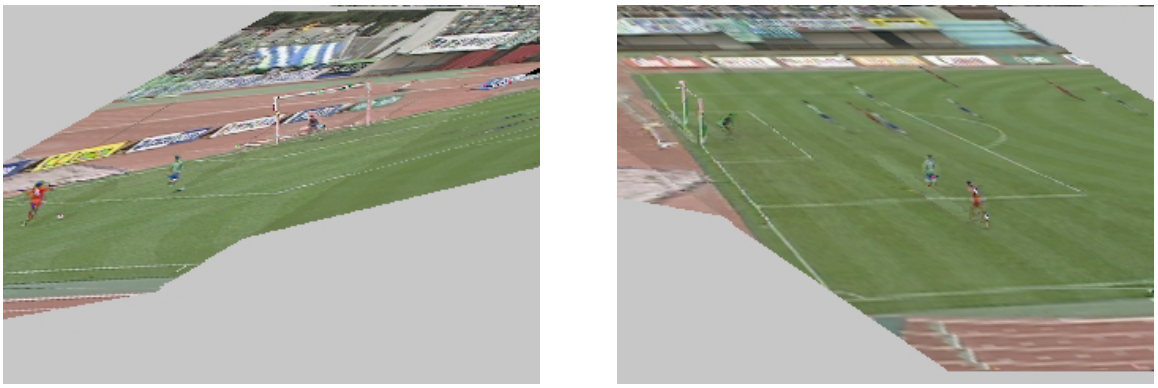
We emphasize that the realism of the virtual images is due to the complicated but physically correct correspondence encapsulated by the set of correspondence graphs. The original work by Seitz applied Beier-Neely morphing [11] or an epipolar-line-based correspondence algorithm [12]

[2] Recall that we introduced this assumption in Section 5.4.2 in order to construct the basic correspondence graphs.



(a) (b)

Figure 6.5: Interpolated virtual images $\hat{\mathcal{I}}_s$ at (a) $s = 0.25$, (b) $s = 0.75$.



(a) (b)

Figure 6.6: Extrapolated virtual images $\hat{\mathcal{I}}_s$ at (a) $s = -0.5$, (b) $s = 1.5$.

to obtain a dense correspondence between a pair of images. However, these techniques make the monotonicity assumption, which is clearly violated in this data set. Using correspondence graphs allows us to obtain a much richer set of virtual images than was previously demonstrated. We can see arrangements of objects in the virtual images (e.g. the position of the goalie with respect to the goalposts in Figure 6.5b) that never occurred in the original frames.

6.3 Virtual Video

In this section we present the main contribution of the chapter, a framework for the recursive propagation of correspondences between frames of two video sequences. Our motivating application is the efficient and accurate synthesis of virtual video, which will be demonstrated in the next section.

6.3.1 Notation

We consider a pair of rotating cameras, $\mathcal{C}_0$ and $\mathcal{C}_1$, taking images of a dynamic scene. The image taken by $\mathcal{C}_k$ at time i for $i = 0, 1, 2, \ldots$ is defined by $\mathcal{I}_k(i)$, which lies on a coordinatized image plane $\mathcal{P}_k(i)$. Our goal is to synthesize the virtual image sequence $\{\mathcal{I}_s(i), i = 0, 1, 2, \ldots\}$ of the scene from the perspective of a moving virtual camera $\mathcal{C}_s$.

We assume the cameras' centers of projection are not coincident, so that every pair of image planes $\mathcal{P}_0(i)$ and $\mathcal{P}_1(i)$ is related by a fundamental matrix $F(i)$. We also assume each camera's center of projection to be constant. Hence, the plane coordinates of $\mathcal{P}_k(i-1)$ and $\mathcal{P}_k(i)$ are related by a projective transformation, denoted by $P(i)$ and $Q(i)$ for $k = 0, 1$ respectively. These assumptions are reasonable in many domains of application such as sports video, where multiple cameras mounted on tripods simultaneously view a scene. The cameras can rotate and zoom, but the translational motion of the tripods is small with respect to the distance to the scene points.

As discussed in Chapter 5, we facilitate the estimation of correspondence along conjugate epipolar lines by rectifying the input image planes. The rectifying projective transformations chosen at time i are denoted as $G(i)$ and $H(i)$, which when applied to the image planes $\mathcal{P}_0(i)$ and $\mathcal{P}_1(i)$ produce image planes $\bar{\mathcal{P}}_0(i)$ and $\bar{\mathcal{P}}_1(i)$ respectively. The various relationships between image planes are illustrated in Figure 6.7.

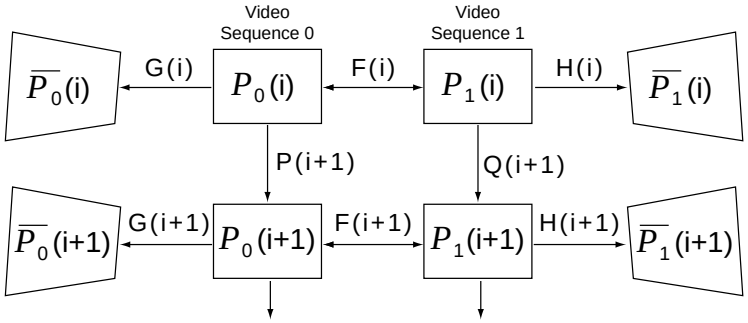


Figure 6.7: Relationships between image planes.

To ease the notation in this section, we will define $\chi^*(i)$ as the (true) correspondence between the image pair $\mathcal{P}_0(i)$ and $\mathcal{P}_1(i)$. To make this more formal, let the set of conjugate epipolar line pairs between $\mathcal{P}_0(i)$ and $\mathcal{P}_1(i)$ be parametrized by the real number β in an interval $I_i \subset \mathbb{R}$. The interval I_i is finite due to the finite extent of the image planes. As β ranges over I_i, it induces the family of conjugate epipolar lines $L(i) = \{(\ell_0^\beta(i), \ell_1^\beta(i)), \beta \in I_i\}$. Let $X(i)$ be the space of all possible correspondence graphs for the family $L(i)$. Making use of Proposition 5.4 from Chapter 5, $\chi \in X(i)$ if and only if $\chi = \{A_\beta, \beta \in I_i\}$, where each A_β is a Southeast set for $(\ell_0^\beta(i), \ell_1^\beta(i))$ satisfying the bounds (5.3)–(5.6). Clearly the true correspondence $\chi^*(i)$ is an element of the space X.

In order to quantify how close two estimates of correspondence are, we will define a metric d_{X_i} on the space X_i. We will use as a subcomponent the Hausdorff metric d_H induced by a metric d on $\mathbb{R}^2$. Let $\mathcal{H}$ be the collection of all nonempty, compact (in the sense of d) subsets of $\mathbb{R}^2$. The Hausdorff metric on $\mathcal{H}$ is defined as follows:

$$d_H(A, B) = \inf\{\delta \mid A \subset B^{(\delta)} \text{ and } B \subset A^{(\delta)}\}$$

where $A, B \in \mathcal{H}$ and $A^{(\delta)}$ is the dilation operator defined in terms of d by

$$A^{(\delta)} = \{x \in \mathbb{R}^2 | \inf_{y \in A} d(x, y) < \delta\}$$

Then we can define the metric d_{X_i} by

$$d_{X_i}(\chi, \chi') = \sup_\beta d_H(A_\beta, A'_\beta)$$

where $\chi = \{A_\beta, \beta \in I_i\}$ and $\chi' = \{A'_\beta, \beta \in I_i\}$. We can think of d_{X_i} as measuring the L^∞ distance between two estimates of correspondence. In this section, we will assume all correspondence graphs are closed sets.

Our goal is to efficiently obtain an estimate of $\chi^*(i)$ at every time step. In the next three sections we discuss our proposed algorithm in detail. In Section 6.3.5 we provide analysis to bound the accuracy of our estimates using the metric on X_i defined above.

6.3.2 Recursive Propagation

Let $\tilde{\chi}(i)$ be an estimate of $\chi^*(i)$ obtained by the application of a correspondence algorithm C^i. We assume that the application of the operator C^i is a time-consuming task, either because a lengthy search process or human intervention is required.

We wish to more efficiently estimate $\chi^*(i)$ at each time. We do so by exploiting the temporal regularity of the video, estimating the effect of camera motion, and using a computationally simpler approximation of C^i. Namely, let $\hat{\chi}(i \mid j)$ be an approximation of $\tilde{\chi}(i)$ based on information from time j. $\hat{\chi}(i \mid j)$ defined by:

$$\begin{aligned} \hat{\chi}(0 \mid 0) &= \tilde{\chi}(0) \\ \hat{\chi}(i+1 \mid i) &= T^{i+1}(\hat{\chi}(i \mid i)) \\ \hat{\chi}(i+1 \mid i+1) &= M^{i+1}(\hat{\chi}(i+1 \mid i)) \end{aligned}$$

Here, T^{i+1} is a time update operator that propagates the correspondence estimate from time i to $i+1$, and M^{i+1} is a measurement update operator that refines the estimate using new information that has become available at time $i+1$. The time-dependency of the update operators arises from their dependency on the images $\mathcal{I}_0(i+1)$ and $\mathcal{I}_1(i+1)$.

To make this algorithm more concrete, we now discuss the operators T^i and M^i in more detail.

6.3.3 Time Update

Given complete knowledge of the camera motion of Figure 6.7, the new position at time $i+1$ of a point match $(w_0(i), w_1(i)) \in \mathcal{P}_0(i) \times \mathcal{P}_1(i)$ is

$$(w_0(i+1), w_1(i+1)) = (P(i+1)w_0(i), Q(i+1)w_1(i))$$

That is, if the only difference between the frames at times i and $i+1$ is due to motion of the cameras, the coordinates of $\mathcal{P}_k(i)$ and $\mathcal{P}_k(i+1)$, $k = 0, 1$, are globally related by a projective transformation.

The time update for rectified image planes can be expressed in a particularly simple form. Suppose $(G(i), H(i))$ rectify $(\mathcal{P}_0(i), \mathcal{P}_1(i))$, such that $H(i)^{-T}F(i)G(i)^{-1} = F^*$. We would like to choose a pair of projective transformations $(G(i+1), H(i+1))$ that rectify $(\mathcal{P}_0(i+1), \mathcal{P}_1(i+1))$. Such a pair is given by the following lemma:

Lemma 6.1: *$(G(i)P(i+1)^{-1}, H(i)Q(i+1)^{-1})$ is a rectifying pair for $(\mathcal{P}_0(i+1), \mathcal{P}_1(i+1))$.*

Proof. It is easily proven that the fundamental matrix $F(i+1)$ relating $(\mathcal{P}_0(i+1), \mathcal{P}_1(i+1))$ is given by

$$F(i+1) = Q(i+1)^{-T}F(i)P(i+1)^{-1}$$

Since

$$(H(i)Q(i+1)^{-1})^{-T}F(i+1)(G(i)P(i+1)^{-1})^{-1} = H(i)^{-T}F(i)G(i)^{-1} = F^*$$

we conclude that $(G(i)P(i+1)^{-1}, H(i)Q(i+1)^{-1})$ is a rectifying pair. ∎

Therefore, we fix

$$G(i+1) = G(i)P(i+1)^{-1} \tag{6.8}$$

$$H(i+1) = H(i)Q(i+1)^{-1} \tag{6.9}$$

Using this special rectifying pair, a point match $(\bar{w}_0(i), \bar{w}_1(i))$ from the rectified images $\bar{\mathcal{P}}_0(i) \times \bar{\mathcal{P}}_1(i)$ is propagated to the rectified images $\bar{\mathcal{P}}_0(i+1) \times \bar{\mathcal{P}}_1(i+1)$ by

$$\begin{aligned}(\bar{w}_0(i+1), \bar{w}_1(i+1)) &= (G(i+1)P(i+1)G(i)^{-1}\bar{w}_0(i), H(i+1)Q(i+1)H(i)^{-1}\bar{w}_1(i)) \\ &= (\bar{w}_0(i), \bar{w}_1(i))\end{aligned}$$

That is, the propagating transformation is simply the identity. Given that we use the rectifying pair in (6.8)-(6.9), this leads us to define the time update operator T^{i+1} that operates on a correspondence estimate $\chi = \{A_\beta, \beta \in I_i\}$ to simply be

$$T^{i+1}(\chi) = \chi$$

This is well-defined since the coordinates of I_i and I_{i+1} agree by construction of the rectifying projective transformations. For the same reason, we can drop the subscript from the metric d_{X_i} since the epipolar lines agree, and refer simply to d_X.

Of course, the various projective transformations are generally estimated using a regression algorithm as described in Chapter 4, so in practice we use an approximation $\hat{T}^{i+1}$ of T^{i+1} given by

$$\hat{T}^{i+1}(\chi) = \chi$$

where the estimated rectifying projective transformations $(\hat{G}(i+1), \hat{H}(i+1))$ are compositions of other estimates given by

$$(\hat{G}(i+1), \hat{H}(i+1)) = (\hat{G}(i)\hat{P}(i+1)^{-1}, \hat{H}(i)\hat{Q}(i+1)^{-1})$$

In Section 6.3.5 we will analyze the implications of this approximation.

Objects that move independently of the camera can be time-updated using a separate segmentation and tracking algorithm if desired. We will discuss our implementation of time-updating in practice in Section 6.4.

6.3.4 Measurement Update

Let C^i be the operator that takes as input an image pair $(\mathcal{I}_0(i), \mathcal{I}_1(i))$ and produces an estimate $\tilde{\chi}(i)$ of the set of correspondence graphs for each pair of conjugate epipolar lines as described in Section 5.4, Algorithm 5.1. This requires the estimation of the basic topology of the correspondence graphs, followed by the solution of a set of monotonic matching problems over a series of series of rectangular domains (see Figure 6.8). We denote this set of domains as $\mathcal{D}^i$.

However, at times $i > 0$, we possess the set of time-updated correspondence graphs from time $i-1$, which we assume to be a good estimate of the set of correspondence graphs at time i. Hence,

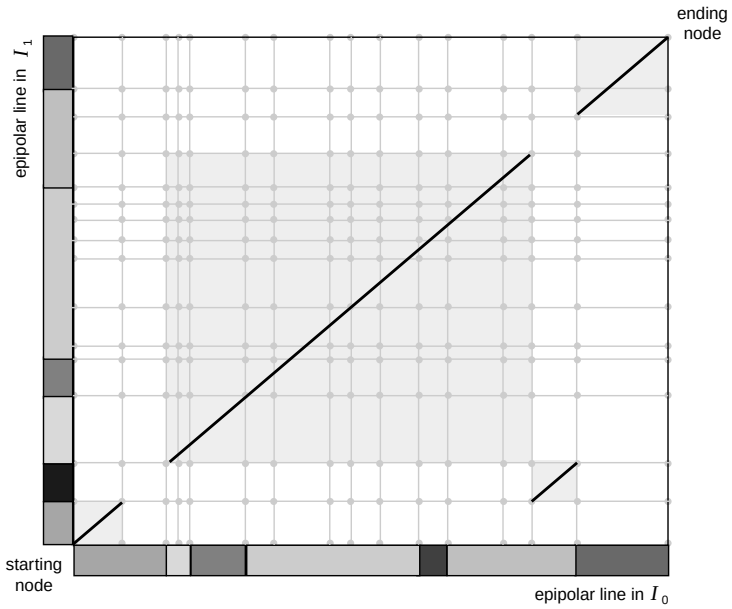


Figure 6.8: The set $\mathcal{D}^i$ of rectangular domains searched by the correspondence operator C^i given basic correspondence graph topology for one epipolar line pair.

we need not search over the set of all possible matching paths as we did at time 0. Instead, given an estimate of correspondence χ,we define the measurement update operator $M^i(\chi)$ to be C^i restricted to an ε-ball around χ. This is illustrated in Figure 6.9 for one epipolar line pair. We denote this set of domains as $\mathcal{B}^i$. Specifically, if χ is the set of Southeast sets $\{A_\beta, \beta \in I_i\}$, then

$$\mathcal{B}^i = \{A^{(\varepsilon)}_{\beta_i}\} \cap \mathcal{D}^i \tag{6.10}$$

Recall $A^{(\varepsilon)}$ is the ε-dilation operator introduced in Section 6.3.1. We intersect the ε-ball with $\mathcal{D}^i$ so that the output of the measurement update operator $M^i(\chi)$ is still a Southeast set with the same topology and endpoints as χ.

By construction, $\mathcal{B}^i \subset \mathcal{D}^i$, and if ε is small the area of $\mathcal{B}^i$ can be substantially smaller than the area of $\mathcal{D}^i$. Specifically, if $\mathcal{D}^i$ is the union of K rectangles with dimensions $M_k \times N_k$, $k = 1, \ldots, K$, then the ratio r of the area of $\mathcal{B}^i$ to the area of $\mathcal{D}^i$ is approximately

$$r = \sum_{k=1}^{K} \frac{2\varepsilon(M_k + N_k) - \varepsilon^2}{M_k N_k}$$

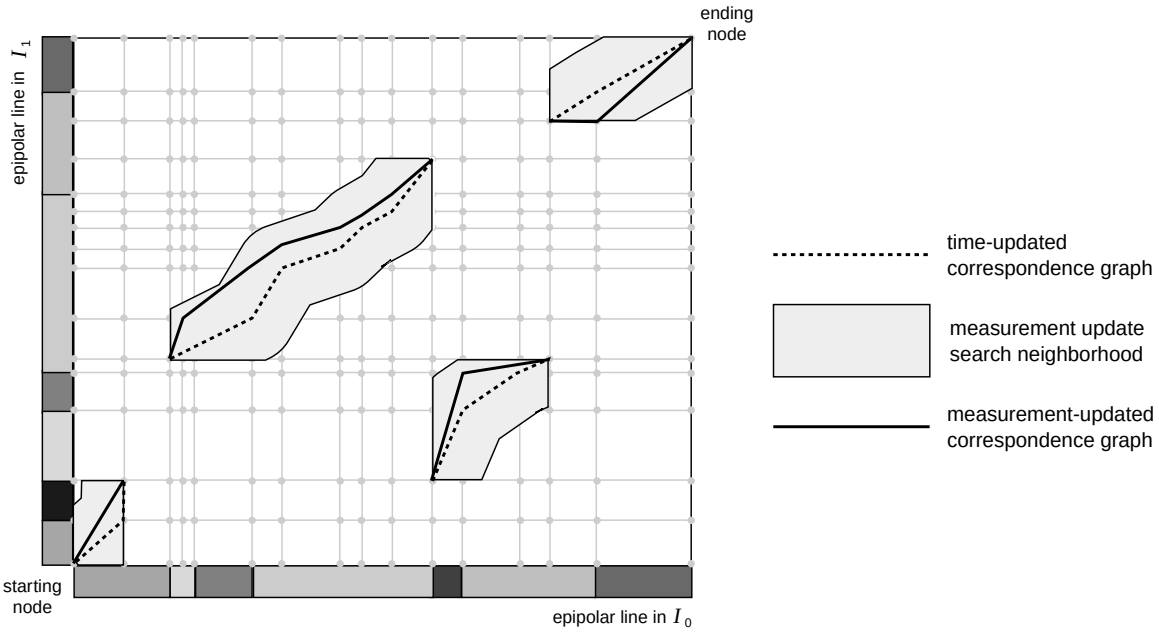


Figure 6.9: Measurement update by searching a local neighborhood $\mathcal{B}^i$ around the time-updated estimate.

if the dilation is based on the L_1 norm, and

$$r = \sum_{k=1}^{K} \frac{2\varepsilon\sqrt{M_k^2 + N_k^2} - \varepsilon^2 \frac{M_k^2+N_k^2}{M_k N_k}}{M_k N_k}$$

if the dilation is based on the L_2 norm. In either case, if $\varepsilon \ll M_k, N_k$, r becomes quite small. Thus, the measurement update M^i can be computed more efficiently than the full correspondence operator C^i, since the computation required to solve the correspondence estimation problem over a domain is proportional to the area of that domain.

6.3.5 Error Analysis

We use the recurrence $\hat{\chi}(i \mid i) = M^i \hat{T}^i(\hat{\chi}(i-1 \mid i-1))$, where $\hat{T}^i$ is an estimate of the true T^i induced by camera dynamics as in Section 6.3.3. We are interested in bounding the difference between the output of the $(\hat{T}, M)$ algorithm and the true correspondence $\chi^*(i)$. To this end, we define the estimation error at each time i as:

$$\epsilon_{TM}(i) = d_X(\chi^*(i), \hat{\chi}(i \mid i))$$

Then we can prove the following theorem on the dynamics of ϵ_{TM}:

Theorem 6.1: *Suppose there exist constants α, γ, δ, and ρ such that for all i,*

$$d_X(T^i(\chi), \hat{T}^i(\chi)) \leq \gamma \tag{6.11}$$

$$d_X(T^i(\chi), T^i(\chi')) \leq \alpha\, d_X(\chi, \chi') \tag{6.12}$$

$$d_X(\chi^*(i+1), T^i(\chi^*(i))) \leq \delta \tag{6.13}$$

$$d_X(\tilde{\chi}(i), \chi^*(i)) \leq \rho \tag{6.14}$$

and that

$$M^i(\tilde{\chi}(i)) = \tilde{\chi}(i) \tag{6.15}$$

Let ε be the radius of the ball used in the measurement update (6.10). Then provided $\alpha < 1$, the $(\hat{T}, M)$ algorithm is stable in the sense that

$$\limsup \epsilon_{TM}(i) \leq \rho + \frac{2\varepsilon + 3\gamma + (\alpha+1)\rho + \delta}{1-\alpha} \tag{6.16}$$

First we prove a simple lemma:

Lemma 6.2: *If ε is the radius of the ball used in the measurement update (6.10), then*

$$d_X(M^i(\chi), M^i(\chi')) \leq 2\varepsilon + d_X(\chi, \chi')$$

Proof. First we show that $d_X(\chi, M^i(\chi))$ is bounded. Let $\chi = \{A_\beta, \beta \in I_i\}$. Fix β and consider a monotonic piece a of A_β. By construction, $M^i(a) \subset a^{(\varepsilon)}$ since the measurement-updated path must lie within an ε-ball of a. Conversely, $a \subset M^i(a)^{(\varepsilon)}$. This can be seen from the diagram in Figure 6.10. For any point p on a, construct the ball of radius ε about p. Since the measurement-updated path is continuous and has the same endpoints as a, it must pass through this ball, and hence every point p of a is contained in an ε-ball about some point of $M^i(a)$.

Thus $d_H(a, M^i(a)) \leq \varepsilon$ for every monotonic piece of A_β and hence $d_H(A_\beta, M^i(A_\beta)) \leq \varepsilon$. Therefore, $d_X(\chi, M^i(\chi)) \leq \varepsilon$, and from the triangle inequality, it follows that $d_X(M^i(\chi), M^i(\chi')) \leq 2\varepsilon + d_X(\chi, \chi')$, as desired. ∎

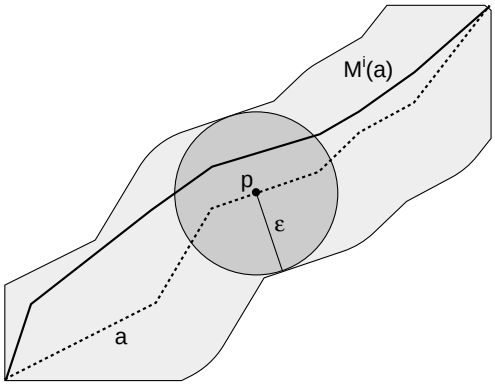


Figure 6.10: Proof that $a \subset M^i(a)^{(\varepsilon)}$.

Proof of Theorem. First, we define an auxiliary estimation error ϵ_D:

$$\epsilon_D(i) = d_X(\tilde{\chi}(i), \hat{\chi}(i \mid i))$$

By repeated applications of the triangle inequality, it is straightforward to show that

$$\begin{aligned} d_X(\hat{T}^i(\chi), \hat{T}^i(\chi')) &\leq 2\gamma + \alpha d_X(\chi, \chi') \\ d_X(T^i(\tilde{\chi}(i-1)), \tilde{\chi}(i)) &\leq (\alpha+1)\rho + \delta \end{aligned}$$

We can compute an upper bound on $\epsilon_D(i)$:

$$\begin{aligned} \epsilon_D(i) &= d_X(\tilde{\chi}(i), \hat{\chi}(i \mid i)) \\ &= d_X(M^i(\tilde{\chi}(i)), M^i(\hat{\chi}(i \mid i-1))) \\ &\leq 2\varepsilon + d_X(\tilde{\chi}(i), \hat{\chi}(i \mid i-1)) \\ &= 2\varepsilon + d_X(\tilde{\chi}(i), \hat{T}^i(\hat{\chi}(i-1 \mid i-1))) \\ &\leq 2\varepsilon + d_X(\tilde{\chi}(i), T^i(\tilde{\chi}(i-1))) + \\ &\qquad d_X(T^i(\tilde{\chi}(i-1)), \hat{T}^i(\tilde{\chi}(i-1))) + d_X(\hat{T}^i(\tilde{\chi}(i-1)), \hat{T}^i(\hat{\chi}(i-1 \mid i-1))) \\ &\leq 2\varepsilon + ((\alpha+1)\rho + \delta) + (\gamma) + (2\gamma + \alpha d_X(\tilde{\chi}(i-1), \hat{\chi}(i-1 \mid i-1)) \\ &= 2\varepsilon + 3\gamma + (\alpha+1)\rho + \delta + \alpha\epsilon_D(i-1) \end{aligned}$$

Hence,

$$\begin{aligned} \epsilon_D^\infty &\equiv \limsup \epsilon_D(i) \\ &\leq \frac{2\varepsilon + 3\gamma + (\alpha+1)\rho + \delta}{1-\alpha} \end{aligned}$$

Finally, since $\epsilon_{TM}(i) = d_X(\chi^*(i), \hat{\chi}(i)) \leq \rho + \epsilon_D(i)$, we have

$$\begin{aligned} \epsilon_{TM}^\infty &\equiv \limsup \epsilon_{TM}(i) \\ &\leq \rho + \limsup \epsilon_D(i) \\ &\leq \rho + \frac{2\varepsilon + 3\gamma + (\alpha+1)\rho + \delta}{1-\alpha} \end{aligned}$$

which is the statement of the theorem. ∎

The conditions of the theorem are not unusually stringent. We require that the output $\tilde{\chi}(i)$ of C^i is fixed by M^i, which is the case when M^i is a restriction of C^i over a smaller domain. The constant γ of (6.11) reflects the accuracy of the projective transformation estimation algorithm, which is a function of the algorithm itself as well as the noise in the point matches. For well-chosen feature extraction and transformation estimation algorithms, γ should be on the order of a few pixel widths. The constant δ of (6.13) reflects scene dynamics that are not modeled by the rotation of the cameras, and can be interpreted as the maximum distance objects can move after compensating for camera motion. If the frames are closely spaced in time, this is again on the order of a few pixel widths. The constant α of (6.12) is related to the relative distance two points can move apart after the application of the projective transformation embedded in T^i. Since the greatest relative expansion occurs at one edge of the finite-extent image plane, α is related not only to the underlying rotation and zoom parameters of the cameras between adjacent frames but also to the dimensions of the image plane. The parameter ε should be chosen proportional to α, δ, and γ. The smaller these parameters are, the more accurate the time update is, and the narrower the search neighborhood needs to be.

By (6.16), the error in the recursive propagation algorithm is uniformly bounded for all time. In particular, when $\rho = 0$, i.e. the operator C^i produces the true correspondence $\chi^*(i)$, the error in the $(\hat{T}, M)$ algorithm is bounded by a quantity that depends on the amount of object motion in the scene, the error in the approximation of T^i by $\hat{T}^i$, and the radius of the measurement update. As these quantities decrease to zero, so does the asymptotic error of the $(\hat{T}, M)$ algorithm.

6.4 Experimental Results: Virtual Video from Wide-Baseline Video

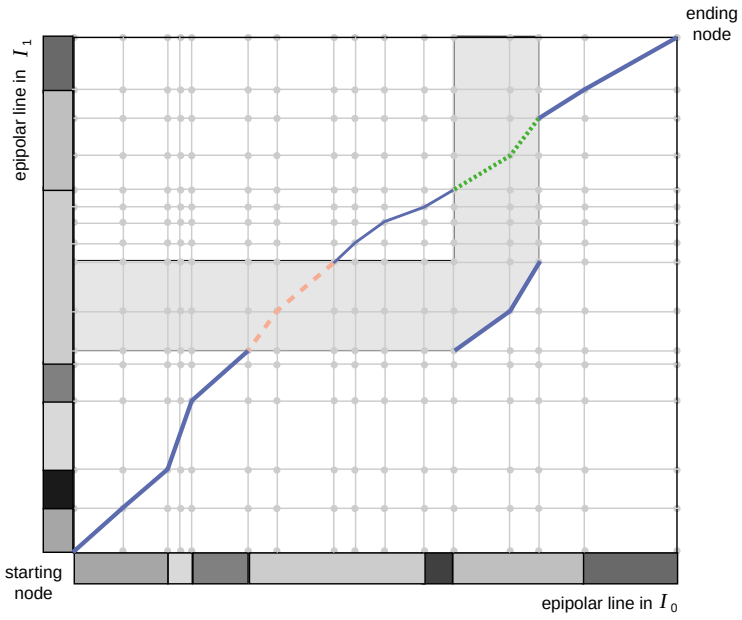
Here we demonstrate the results of the recursive propagation framework in the context of creating virtual video. Our test sequence is 43 frames long and constitutes a single event from a soccer game (a player attempts to kick the ball and is tripped). The frames are 340×240 pixels, and come from a high-quality digital video camera.

We make an implementational comment regarding the filling-in of occluded regions as the algorithm progresses. In Section 6.1 we discussed how the planar assumption could be used to estimate correspondence for points visible in only one image. While the planar assumption can be propagated to subsequent frames by composing projective transformations, we can do better by time-updating the correspondence for the entire background piece at each iteration, regardless of visibility. Parts of this background piece that had been seen in previous frames have had their correspondence estimated at prior steps, and may become visible again. Hence, our implementation of the time-update is:

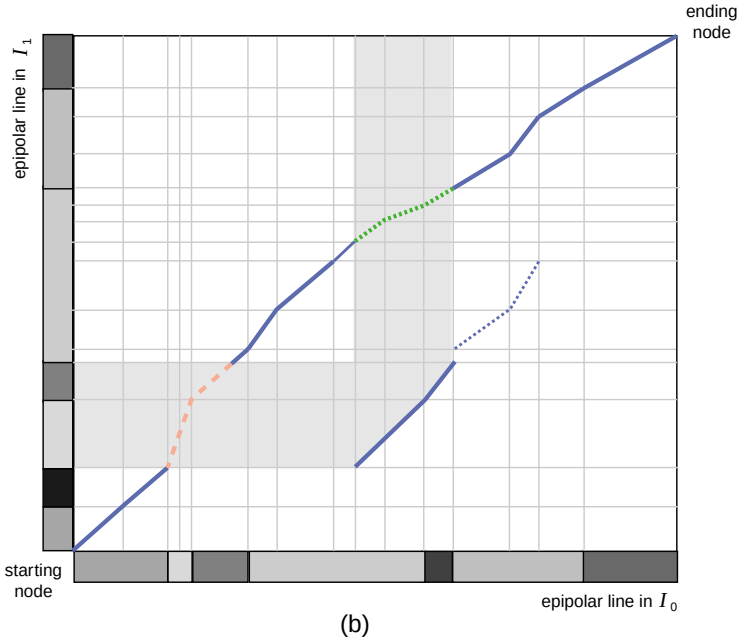
Algorithm 6.1: *Practical time-update.*

1. *Estimate the projective transformations $P(i)$ and $Q(i)$.*
2. *Rectify the image planes at time $i+1$ with the approximate rectifying pair $(\hat{G}(i)\hat{P}(i+1)^{-1}, \hat{H}(i)\hat{Q}(i+1)^{-1})$.*
3. *Initialize the background correspondence at time $i+1$ in the rectified images to be the same as the background correspondence at time i, discounting any previous information about which regions were occluded.*
4. *Track the foreground objects to their locations at time $i+1$.*
5. *Create a basic correspondence graph for each epipolar line pair by Southeasting the foreground pieces onto the background piece. Retain correspondence estimates for the occluded regions to use in rendering and in subsequent iterations.*

The last three steps of the algorithm are illustrated schematically in Figure 6.11.



(a)



(b)

Figure 6.11: Filling in occluded regions in the time update. (a) The regions of the correspondence graph from time i removed by Southeasting are retained. (b) At time $i+1$, the entire background is reinitialized as a visible piece and the new object is Southeasted onto it.

Our current implementation produces virtual video at about 20 frames per minute. The only user intervention required is a sparse set of point correspondences in the initial frame pair (used to estimate the fundamental matrix and the projective transformation relating the dominant plane in the image pair), and segmentation and tracking information for moving objects in each frame (used to construct correct correspondence graphs). In this example, to obtain the best possible results, this information was obtained by hand.

The projective transformations $P(i)$ and $Q(i)$ were estimated using the efficient algorithm described in Chapter 4, using point matches extracted by the automatic feature selection algorithm described in Section 3.4. The measurement update used an 8-pixel search neighborhood about the time-updated estimate.

Figure 6.12-6.14 illustrate the results of the algorithm on conjugate epipolar lines 71, 105, and 120 for the first and second frames of video (labeled Frame 415 and Frame 417). The upper left hand corner of each figure is the basic correspondence graph for Frame 415 induced by the planar assumption and object segmentation. The upper right hand corner is the refined correspondence graph for Frame 415 obtained by applying the measurement update operator to the basic correspondence graph. The lower left hand corner is the correspondence graph for Frame 417 obtained by the time update, and the lower right hand corner is the correspondence graph for Frame 417 obtained by the measurement update.

The correspondence graphs all seem rather similar (which is the point of the algorithm). However, it can be seen clearly in Figure 6.13 that the background correspondence from Frame 415 is time-updated to the same location in Frame 417 (note the "elbow" at the lower left end of the long piece). This correspondence is refined by the measurement update (and the elbow disappears).

More compelling are the virtual video frames rendered using this correspondence. Six such frames are illustrated in Figures 6.15-6.20. In each figure, the upper left and upper right images are real images $\{\mathcal{I}_0(i), \mathcal{I}_1(i)\}$ seen at time i, corresponding to locations along the baseline of $s = 0$ and $s = 1$. The lower left image is a rendition of the scene from a stationary camera with optical center fixed at $s = 0.5$. The lower right image is a rendition from a moving camera whose optical center moves at constant speed from $s = 0$ to $s = 1$. The figures are selected to be spaced apart along the baseline by roughly $\frac{s}{5}$. Over the course of the video clip, camera $\mathcal{C}_0$ undergoes a slow pan

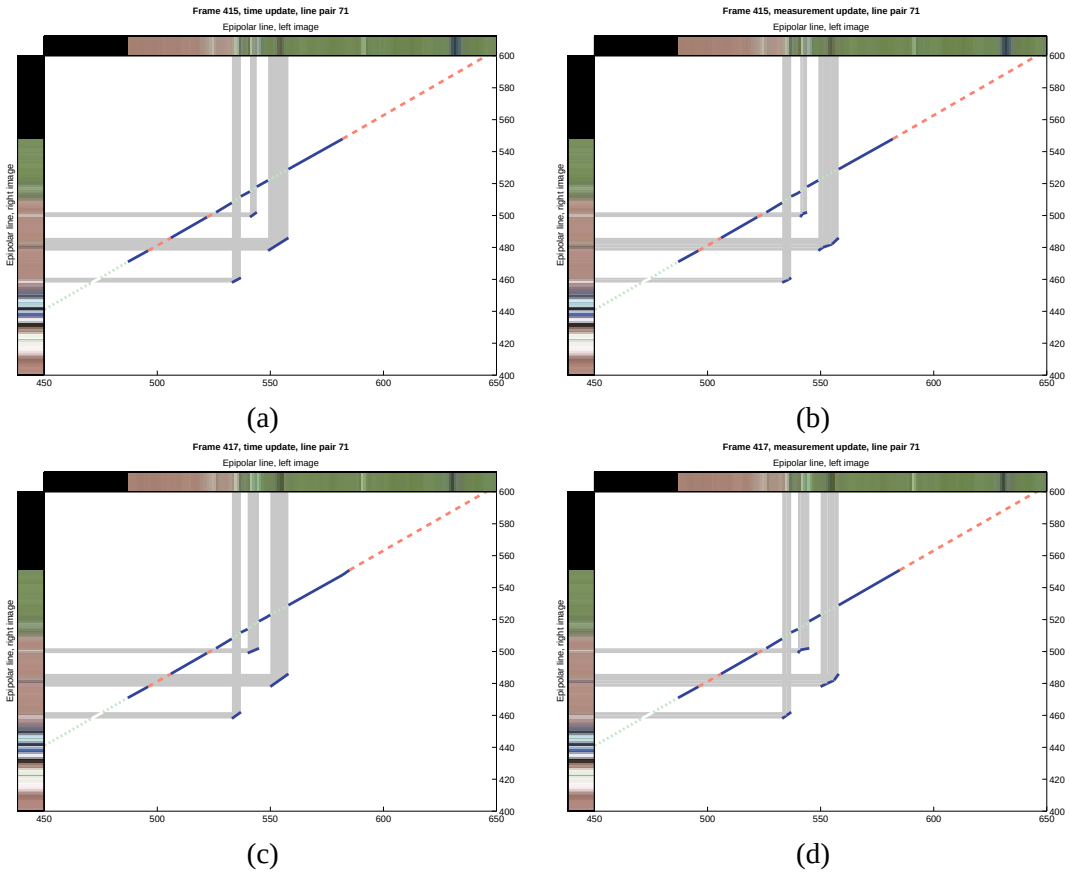


Figure 6.12: Correspondence graphs, line 71. (a) Frame 415 initialization. (b) Frame 415 measurement update. (c) Frame 417 time update. (d) Frame 417 measurement update.

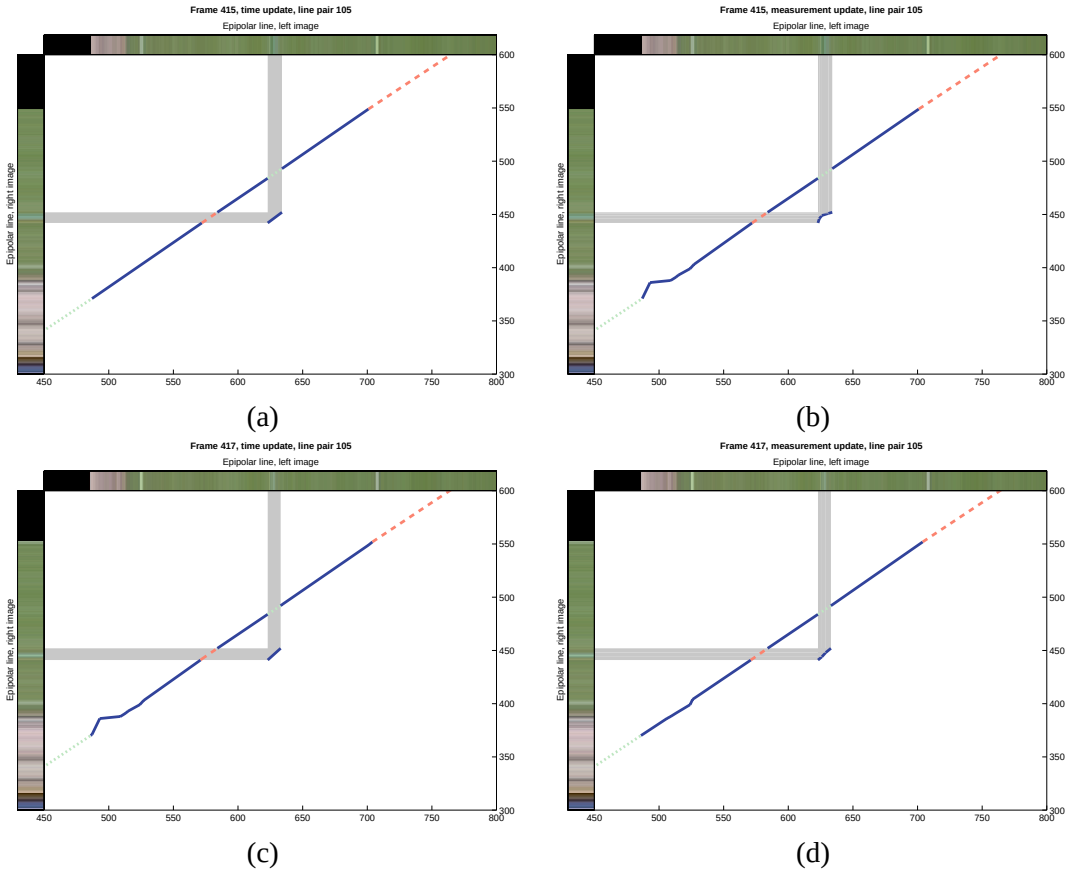


Figure 6.13: Correspondence graphs, line 105. (a) Frame 415 initialization. (b) Frame 415 measurement update. (c) Frame 417 time update. (d) Frame 417 measurement update.

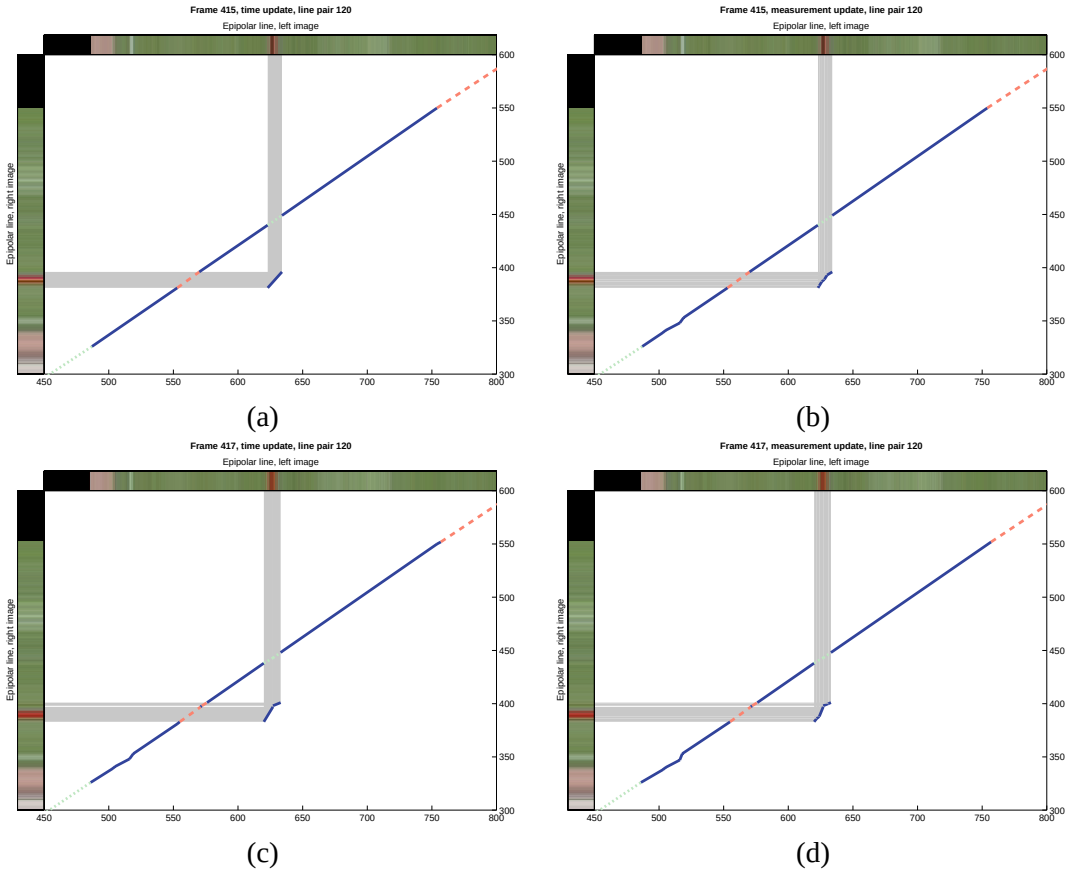


Figure 6.14: Correspondence graphs, line 120. (a) Frame 415 initialization. (b) Frame 415 measurement update. (c) Frame 417 time update. (d) Frame 417 measurement update.

to the right, while camera $\mathcal{C}_1$ slowly zooms in. The virtual camera has dynamics observed in neither of the source video clips.

Figure 6.15: Frame 415. Upper left, original $\mathcal{C}_0$ frame, $s = 0$. Upper right, original $\mathcal{C}_1$ frame, $s = 1$. Lower left, virtual $\mathcal{C}_s$ frame, $s = 0.5$. Lower right, virtual $\mathcal{C}_s$ frame, $s = 0$.

Unfortunately, it is difficult to convey the three-dimensional feeling of the rendered video from these still images. However, as in the single-frame-pair example we presented in Section 6.2, each of the virtual images is a convincing rendition of the dynamic from an intermediate viewpoint. We emphasize that the effects exhibited here are similar to those produced by specialized multicamera hardware. However, here we only require two uncalibrated cameras and no 3-D scene modeling. These results show that understanding the relationship between image correspondence and camera motion can be a powerful tool.

In later frames of the video, there are minor but visible artifacts. Notably, some of the players seem to "lose their heads"- the head of the player appears several pixels away from the correct

Figure 6.16: Frame 433. Upper left, original $\mathcal{C}_0$ frame, $s = 0$. Upper right, original $\mathcal{C}_1$ frame, $s = 1$. Lower left, virtual $\mathcal{C}_s$ frame, $s = 0.5$. Lower right, virtual $\mathcal{C}_s$ frame, $s = 0.2143$.

Figure 6.17: Frame 447. Upper left, original $\mathcal{C}_0$ frame, $s = 0$. Upper right, original $\mathcal{C}_1$ frame, $s = 1$. Lower left, virtual $\mathcal{C}_s$ frame, $s = 0.5$. Lower right, virtual $\mathcal{C}_s$ frame, $s = 0.3810$.

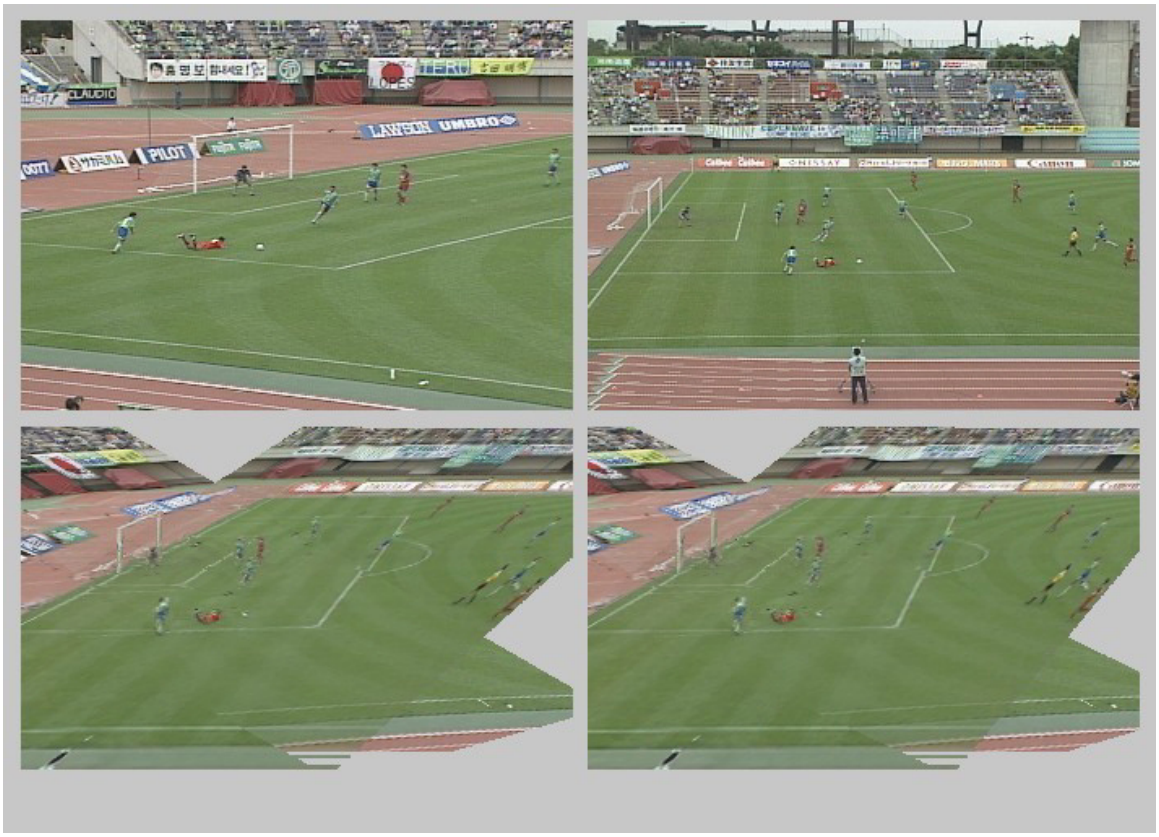


Figure 6.18: Frame 465. Upper left, original $\mathcal{C}_0$ frame, $s = 0$. Upper right, original $\mathcal{C}_1$ frame, $s = 1$. Lower left, virtual $\mathcal{C}_s$ frame, $s = 0.5$. Lower right, virtual $\mathcal{C}_s$ frame, $s = 0.5952$.

Figure 6.19: Frame 487. Upper left, original $\mathcal{C}_0$ frame, $s = 0$. Upper right, original $\mathcal{C}_1$ frame, $s = 1$. Lower left, virtual $\mathcal{C}_s$ frame, $s = 0.5$. Lower right, virtual $\mathcal{C}_s$ frame, $s = 0.8571$.

Figure 6.20: Frame 499. Upper left, original $\mathcal{C}_0$ frame, $s = 0$. Upper right, original $\mathcal{C}_1$ frame, $s = 1$. Lower left, virtual $\mathcal{C}_s$ frame, $s = 0.5$. Lower right, virtual $\mathcal{C}_s$ frame, $s = 1$.

location on top of the body. This is especially visible in Figure 6.19. This is largely due to the accumulation of errors in the estimation of the projective transformations $P(i)$ and $Q(i)$, which in turn affect the accuracy of the estimated rectifying projective transformations $G(i)$ and $H(i)$. Though our projective transformation estimation algorithm is generally quite accurate, after n iterations, the projective transformations $\hat{G}(n)$ and $\hat{H}(n)$ applied to $\mathcal{P}_0(n)$ and $\mathcal{P}_1(n)$ are compositions of n estimated transformations. In this video sequence, when n is more than about 25, $(\hat{G}(n), \hat{H}(n))$ are no longer close to a rectifying pair. This problem could be alleviated by a periodic re-estimation of the epipolar geometry. We address this issue briefly in the next section.

6.5 Conclusions

There are many directions for future work in the area of virtual video, both in improving the stability of the estimation algorithm and in the rendering of the synthetic images.

As addressed in the text, the propagation process eventually destabilizes, due to accumulation of errors in the estimation of the projective transformations. What is required is a reinitialization of the epipolar geometry. However, since this estimation requires the selection and matching of feature points between images with a substantial perspective difference, user intervention is generally required to obtain reliable results. Since some matching points are selected by the user for the first frame pair, one approach is to track these points through each image sequence, using a measure of feature similarity that is invariant to perspective distortion (e.g. based on corners). Periodically, the algorithm could be restarted with a new estimate of the fundamental matrix and rectifying projective transformations. Automatically detecting that restarting is necessary and maintaining continuity of the rectifying transformations and virtual images across the restarted frame would be problems to overcome.

The perceptual quality of the rendered images could be improved using techniques from computer graphics. For example, given appropriate texture models, planar surfaces in the scene could be rendered by texture mapping instead of interpolation of image intensities at each frame pair. However, a time-invariant texture-mapping method would perform poorly if the scene were undergoing a steady change in illumination.

The accuracy of the object segmentation also affects the perceptual quality of the rendered images. If an object is segmented too conservatively, pieces of the background will be erroneously removed and rendered along with the object. On the other hand, if an object is segmented too liberally, pieces of the object will be erroneously left behind on the background. On the whole, conservative segmentation is preferable to liberal segmentation in cases where the background is approximately uniform, as in our example.

We also note that the algorithm presented here depends crucially on the assumption that $\mathcal{P}_0(i)$ and $\mathcal{P}_1(i)$ are images of the same scene taken at exactly the same time. This type of synchronization is common in broadcast video, especially sports video, where an editor needs to be able to switch between different cameras with no noticeable discrepancies in timing. Obtaining synchronized video from multiple cameras of a dynamic scene without professional equipment is difficult, and the estimation of and compensation for synchronization offsets between multiple video sequences would be an interesting research problem.

This work can be thought of interpolation of video frames in the spatial (camera) domain. However, once we have estimated the correspondence graph between frames of video, it can be used to interpolate frames in the time domain as well. We develop this idea more in Chapter 7. One of the difficult issues here is the correct interpolation of locations of objects that move independently of the camera between frames. If a good solution is obtained, we can obtain virtual video at a higher frame rate than the original video sequences.

As an aside, we mention that the virtual video described here has no audio component, and one might naturally ask whether ideas from view interpolation apply to the problem of synthesizing virtual audio. This is in fact the case, and in Appendix D we supply a derivation and implementation of audio interpolation equations, for synthesizing the audio signal that would be received by a microphone located between two real microphones.

6.6 References

[1] M. Levoy and P. Hanrahan. Light Field Rendering. *Computer Graphics (SIGGRAPH '96)*, pp. 31–42, August 1996.

[2] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The Lumigraph. *Computer Graphics (SIGGRAPH '96)*, pp. 43–54, August 1996.

[3] S. Laveau and O.D. Faugeras. 3-D Scene Representation as a Collection of Images and Fundamental Matrices. Technical Report 2205, INRIA-Sophia Antipolis, February 1994.

[4] S. Avidan and A. Shashua. Novel View Synthesis by Cascading Trilinear Tensors. *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 4, October-December 1998.

[5] Y. Ma, S. Soatto, J. Košecká, and S. Sastry. Euclidean Reconstruction and Reprojection up to Subgroups. *International Journal of Computer Vision*, Vol. 38, No. 3, pp. 219–229, 2000.

[6] L. McMillan and G. Bishop. Plenoptic Modeling: An Image-Based Rendering System. *Computer Graphics (SIGGRAPH '95)*, pp. 39–46, August 1995.

[7] R. Radke, P. Ramadge, S. Kulkarni, T. Echigo, and S. Iisaku. Recursive Propagation of Correspondences with Applications to the Creation of Virtual Video. In *Proc. ICIP 2000*, Vancouver, Canada, September 2000.

[8] S.E. Chen and L. Williams. View Interpolation for Image Synthesis. *Computer Graphics (SIGGRAPH '93)*, pp. 279–288, July 1993.

[9] S.M. Seitz and C.R. Dyer. View Morphing. *Computer Graphics (SIGGRAPH '96)*, pp. 21–30, August 1996.

[10] D. Scharstein. Stereo Vision for View Synthesis. *Proc. CVPR '96*, pp. 852–858, 1996.

[11] T. Beier and S. Neely. Feature-Based Image Metamorphosis. *Computer Graphics (SIGGRAPH '92)*, pp. 35–42, July 1992.

[12] Y. Ohta and T. Kanade. Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. *IEEE PAMI*, Vol. 7, No. 2, pp. 139–154, March 1985.

Chapter 7

View Morphing for Time-Domain Interpolation of Low-Bit-Rate Video

In the previous chapter, we discussed several of the estimation problems involved with synthesizing virtual video, and demonstrated the graphical effects that can be achieved with virtual video techniques. However, since the main application of virtual video we discussed was to synthesize images from the perspective of cameras that did not exist in the original environment, there was no quantitative measure of the quality of the synthesized views.

In this final chapter, we show that in addition to enabling compelling graphical effects, virtual view-based algorithms can have benefits in other engineering applications. Specifically, we discuss the domain of low bit-rate video coding for wireless multimedia applications.

Current video coding algorithms exploit the fact that adjacent frames in a video shot are usually very similar. Typically some set of frames, a small fraction of the total number, are coded independently with high fidelity, and the rest of the frames are coded with reference to these anchor frames by motion compensation.

In Section 6.2 we demonstrated that given correspondence between two image planes, a certain class of physically correct perspective views can be constructed. Even when the perspective difference between the two source images is sizable, accurate intermediate views can be synthesized. This gives us a new perspective on video coding, since we no longer need adhere to the assumption that one frame can be well-predicted from another only if they look similar.

This chapter builds on all the previous ones to present an algorithm for synthesizing virtual

images of a scene that match frames from a source video clip. We use this algorithm for interpolation of video frames in the time domain, using a small amount of information to construct an approximation of the original video. Since this approach is based on estimating functions of the underlying camera motion parameters, and not on local block-based motion, it can capture relationships between image correspondences that extend across many (perhaps hundreds) of video frames. Each interpolated image can be rendered using only a few tens of bytes of side information, and the rendering process itself has low computational requirements. We present experimental results to demonstrate that for a 45 kbps (kilobits per second)[1] bit rate, our algorithm gives significant perceptual improvement over MPEG-1 coded video at a higher bit rate. Our approach is particularly amenable to representing computer-generated video, for which the correspondence and camera motion information required for view synthesis is readily available at render time. We note that while there has been some work on using "virtual views" for video coding (e.g. [1]), these are generally mesh-based methods aimed at the small-baseline case for compressing video teleconferencing data. Our results give a higher PSNR for a lower bit rate.

In Section 7.1 we briefly review existing video compression standards, which are generally based on block-based motion compensation. These can be thought of as algorithms for interpolation of video frames in the time domain from relatively high-fidelity, independently coded anchor frames. The various schemes differ in the positioning of the anchor frames and the type and amount of side information that is used to reconstruct the intermediate frames, but they share the underlying assumption that the intermediate frames should not look too different from the anchor frames.

In Section 7.2, we present an algorithm for time-domain interpolation of video frames that fits into the same framework as above. The advantage of our method is the use of view morphing to synthesize intermediate views between anchor frames, which removes the restriction that adjacent anchor frames are temporally close and visually similar. Consequently, in theory, the anchor frames can be taken much further apart than is common in current compression standards.

[1]Throughout this chapter, "kilobits" has the literal interpretation of 1000 bits, and not the common interpretation of 1024 bits.

To apply view interpolation methods, we require a dense and accurate estimate of correspondence between each pair of anchor frames. We discussed many of the issues involved in this estimation in Chapter 5, and make some additional comments in Section 7.3.

Once the anchor frames have been specified, the heart of the algorithm lies in synthesizing a virtual image that is a good approximation to an actual frame of a video sequence. In Section 7.4, we pose and solve the problem of jointly estimating the relative position of the virtual camera and the projective transformation that rotates the virtual image plane to align with the actual image plane.

We show some experimental results from our algorithm in Section 7.5 and discuss future work in Section 7.6. A shorter version of the results in this chapter appeared in [2].

7.1 Review of Video Compression Algorithms

While video compression algorithms differ considerably in details of implementation, the vast majority of them are based on block-based motion compensation and transform coding. The general approach, illustrated in Figure 7.1, can be described by:

Algorithm 7.1: *Block-based motion-compensated video compression.*

1. *Designate each video frame as intracoded (I), predictive (P), or bidirectional (B).*
2. *Encode the I frames independently with relatively high fidelity, e.g. with a transform-coding still image compression algorithm.*
3. *Split each P frame into smaller blocks. For each block, search for a matching block in the previous I or P frame that has low mean-squared error. Save the motion vector that points from the source block to its match, and encode the residual between the two blocks using transform coding.*
4. *Do the same for B frames, except search for the matching block in both the previous and subsequent I/P frames.*
5. *Efficiently code the data to be transmitted with a mixture of variable length coding algorithms.*

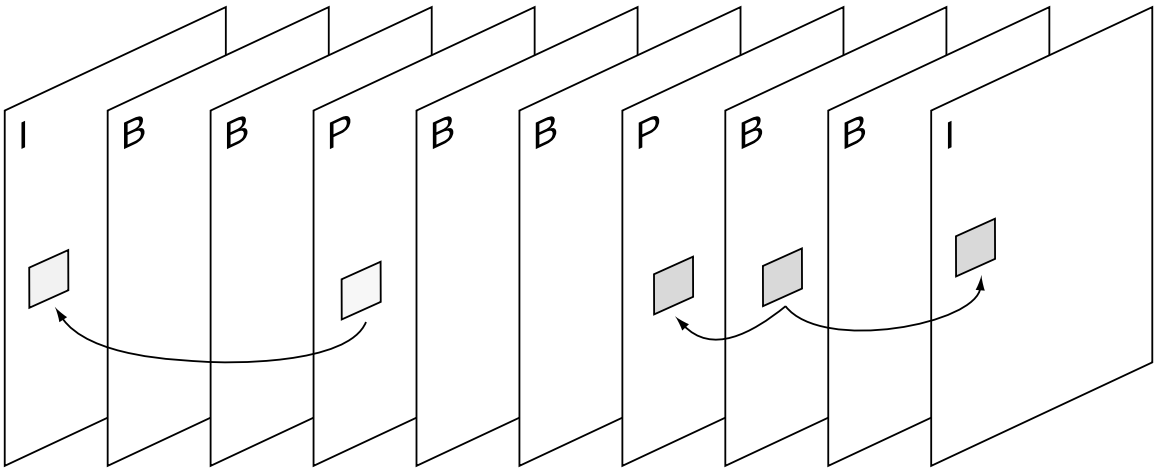


Figure 7.1: Schematic of block-based motion-compensation video compression algorithms. I frames are coded independently. P frames look to the previous I/P frame to match blocks. B frames look to the previous I/P frame and to the next I/P frame to match blocks.

The prevalent MPEG-x and H.26x standards both fall under this framework. MPEG-1 [3] was designed for general video compression at bit rates up to 1.5 Mb/sec. MPEG-2 [4] was a generalization of MPEG-1 to allow for interlacing and larger frames. MPEG-4 [5] is targeted at a flexible, object-oriented representation of video that allows different regions of pixels to be separately encoded at different rates. MPEG-4 also extends the block-based model to further allow the projective warping of a set of background pixels, and mesh-based modeling of video objects [6]. H.261 [7] is a standard comparable to MPEG-1 designed for ISDN lines at data rates up to 2 Mb/sec. H.263 [8] is similar to H.261, but geared for teleconferencing and designed for much lower bit rates, e.g. 64 kbps.

The various algorithms differ in several aspects (for example, the labeling of frames as I/P/B, the restrictions put on the motion vectors for each P/B frame, the frame sizes that are supported, the order and method by which the residuals of blocks are coded) but these details are unimportant for the discussion here. The main point is that current video compression schemes are almost universally based on block-based motion compensation, with the underlying assumption that the frames designated as P or B are visually similar to the frames designated as I on either side, so that the mean-squared error between a block and its predictor is small. In practice, the I frames are taken

(a)

(b)

Figure 7.2: (a) Two frames of test video, separated by 15 frames. (b) Their absolute luminance difference (darker pixels have a higher magnitude).

to be on the order of 10-30 frames apart. For example, the two frames in Figure 7.2a are 15 frames apart in the 30 frames per second (fps) clip of test video we will consider in Section 7.5. Figure 7.2b, the luminance difference between them, indicates that corresponding pixels are not more than 8 pixels apart after the 15 frames have elapsed.

The transmission and reconstruction of video in wireless multimedia poses a much more difficult problem than it does in a wired setting. There are three main issues that complicate matters:

1. The wireless multimedia channel has very limited bandwidth compared to a wired channel, so video data needs to be reduced in both frame size and frame rate. An uncompressed 24 bits-per-pixel (bpp) video at 320 x 240 pixel resolution, 30 fps, requires a data rate of 55 Mbps (megabits per second). In contrast, a typical set of H.263 parameters is QCIF resolution (176 x 144 pixels) at 10 fps, to be encoded at a bit rate of 11.36 kbps.

2. The wireless multimedia client has a limited power supply. Since the power required for one pass through an algorithm grows with the number of arithmetic operations that need to be executed, we require simple algorithms to reconstruct video from the transmitted data.

3. The wireless multimedia channel has very high bit error rates. The probability that a bit is corrupted may be as high as 0.01. For video compression schemes that use variable length coding, a single bit error can have damaging effects over several of the reconstructed video frames. Therefore, the compressed video bitstream needs to have robust error correction capabilities. Adding error correction further reduces the bit rate available for video data.

While issues of error correction cannot be ignored in a practical video-over-wireless scheme, here we take a higher-level approach that addresses the issues of limited bandwidth and complexity. Many of the algorithms that have been designed for error correction at the encoder or error concealment at the decoder [9] can be applied to the algorithm we propose. We will discuss aspects of error protection in Section 7.6. We emphasize that currently, this scheme is built on top of, and is compatible with, standard video compression algorithms. When a segment of video that is suitable for interpolation is encountered, the server can transmit the low-overhead side information concurrently with a standard video data stream. A “smart” receiver equipped with our algorithm can

take advantage of this information to render the video segment at higher quality, while a "normal" receiver ignores the side information and produces standard-quality video.

7.2 View Morphing as a Predictive Mechanism

We consider the camera path illustrated in Figure 7.3a. Here, dots and arrows represent the positions and orientations of a camera during a video shot of a static scene.

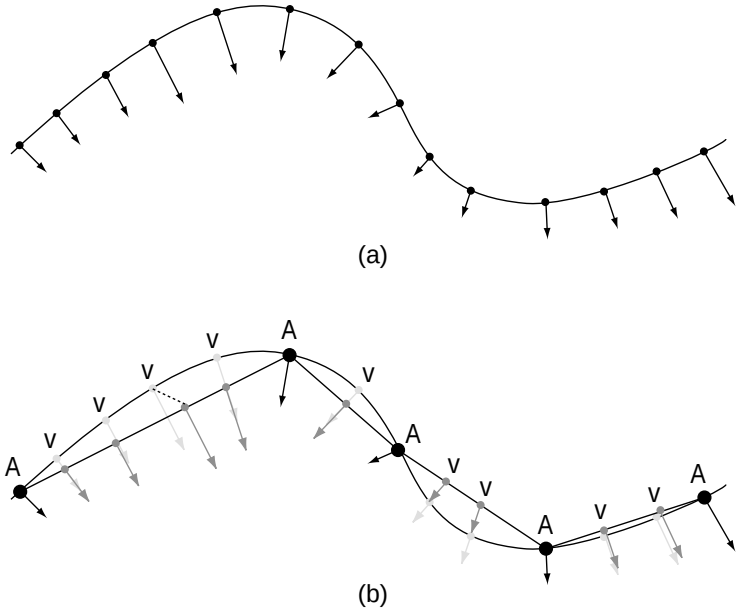


Figure 7.3: View morphing for interpolating video from a translating camera. (a) A video shot in which the camera translates. (b) Frames are designated as anchor (A) frames or virtual (V) frames.

Our approach to frame interpolation is illustrated in Figure 7.3b. A fraction of the frames are selected to be anchor, or "A", frames. These frames are the same as intracoded frames in the standard terminology, and are transmitted with good fidelity using a standard image compression scheme. The rest of the frames are designated as virtual, or "V", frames. These are interpolated between adjacent A frames using the view synthesis and registration algorithm described below.

We saw in Chapter 6 that for piecewise linear paths, we can synthesize virtual frames using view morphing that resemble real views of a scene, and are physically correct perspective views provided the estimate of correspondence between image planes is accurate. This is true even when

the perspective difference between the source frames is quite large. Hence, in principle, we may temporally subsample the video at wide intervals to obtain the A frames. The anchor frames need not be equally spaced along the image sequence; ideally they should automatically be chosen with respect to estimated camera dynamics.

In the remainder we assume the A frames have been chosen and consider one line segment of Figure 7.3b. We will work with a sequence of video frames $\{\mathcal{I}_t\}$, with $t = 0, \Delta t, 2\Delta t, \ldots, 1$, which are generated by a moving camera whose (unknown) parameters at time t are $\{\mathcal{C}_t\}$. We designate $(\mathcal{I}_0, \mathcal{I}_1)$ as the anchor frames, with associated image planes $(\mathcal{P}_0, \mathcal{P}_1)$. Our goal is to synthesize a virtual image $\hat{\mathcal{I}}_t$ using view morphing between $\mathcal{I}_0$ and $\mathcal{I}_1$ to approximate each intermediate frame $\mathcal{I}_t$. The information that comprises the transmitted video is thus:

1. The anchor frames $(\mathcal{I}_0, \mathcal{I}_1)$ (suitably compressed).

   and the following side information:

2. A pair of rectifying projective transformations (G, H) that rectify $(\mathcal{I}_0, \mathcal{I}_1)$.

3. The structured correspondence (i.e. set of correspondence graphs) between the image planes $\mathcal{P}_0$ and $\mathcal{P}_1$.

4. The camera position of each virtual frame, described as a fraction s_t of distance along the baseline connecting the optical centers of $\mathcal{C}_0$ and $\mathcal{C}_1$ (1 floating point number per V frame).

5. The projective transformation K_t that aligns each virtual frame with the corresponding actual frame $\mathcal{I}_t$ (8 floating point numbers per V frame).

Then for every estimated corresponding pair (w_0, w_1) in $\mathcal{P}_0 \times \mathcal{P}_1$, a pixel is rendered at position w_t in $\hat{\mathcal{I}}_t$ by view morphing:

$$w_t = K_t^{-1}((1 - s_t)G(w_0) + s_t H(w_1)) \tag{7.1}$$

$$\hat{\mathcal{I}}_t(w_t) = (1 - s_t)\mathcal{I}_0(w_0) + s_t \mathcal{I}_1(w_1) \tag{7.2}$$

The perceptual fidelity of video frames interpolated using the algorithm above depends crucially on the accuracy of the correspondence at step 3 above, and the estimates of s_t and K_t at steps 4 and 5. Our practical approach to these estimation problems is discussed in Sections 7.3 and 7.4 below.

Should the original video be synthetic and computer-rendered, the renderer can be easily modified to output precise information about correspondence and camera motion between frames, saving the trouble of estimating them later.

7.3 Correspondence Between Anchor Frames

We take the same approach to estimating correspondence between the pair of image planes $(\mathcal{P}_0, \mathcal{P}_1)$ as detailed in Chapter 5. First we estimate the epipolar geometry from a sparse set of point matches, and then we construct the correspondence graph for each pair of conjugate epipolar lines (ℓ_0, ℓ_1).

In Section 5.4.2 we discussed how the assumption that a planar surface comprises the entire scene induces correspondence between a pair of image planes. The result generalizes to the case when the scene is composed of a set of planar facets. For any such facet, consider its image in a pair of rectified image planes $(\mathcal{P}_0, \mathcal{P}_1)$. Let $((x_0^1, y), (x_1^1, y))$ and $((x_0^2, y), (x_1^2, y))$ be two pairs of corresponding points on the image of the facet in $\mathcal{P}_0 \times \mathcal{P}_1$. The same argument from Section 5.4.2 applies to each facet. That is, linear interpolation between corresponding points produces correct correspondence in the interior of the delimited interval, and

$$(((1-\alpha)x_0^1 + \alpha x_0^2, y), ((1-\alpha)x_1^1 + \alpha x_1^2, y))$$

is a correct correspondence for $\alpha \in [0, 1]$. Again, we note that this interpolation is only valid on a planar facet, for rectified image planes. The construction of a correspondence graph for a scene composed of planar facets is illustrated in Figure 7.4. Interpolation as discussed above can be used to initialize correspondence between anchor frames when we possess a set of several control line segments which delimit planar facets of the scene, e.g. the edges of the rectangular solid in Figure 7.4 or the line segments in Figure 7.6.

A correspondence graph can itself be "compressed" by simplification. For example, if bit rate requirements are particularly stringent, a basic correspondence graph obtained by linearly interpolating between control lines can be transmitted, instead of the refined version obtained by dynamic programming that contains more nodes, and hence requires more bits to describe.

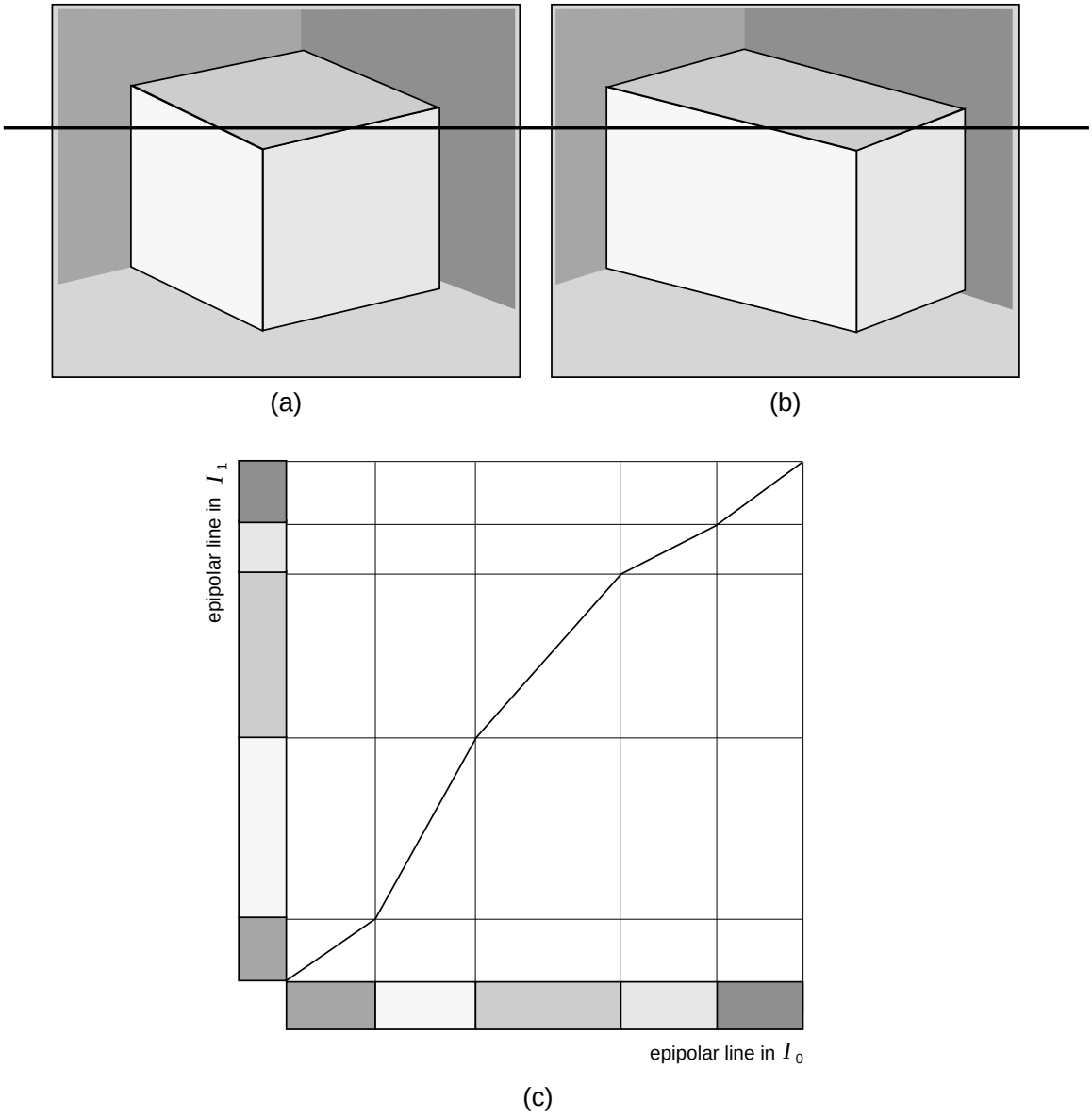


Figure 7.4: (a) and (b) Rectified image planes of a scene composed of planar facets. (c) Correspondence graph for highlighted epipolar line pair, constructed by linear interpolation between endpoints of facets.

7.4 Aligning the Virtual and Actual Frames

Given the correspondence between $(\mathcal{P}_0, \mathcal{P}_1)$ and a pair of rectifying projective transformations (G, H), it remains to describe the approximation $\hat{\mathcal{I}}_t$ by the nine parameters (s_t, K_t) required by the view morphing equation 7.1. Hence, we can write $\hat{\mathcal{I}}_t$ as an explicit function of the parameters:

$$\hat{\mathcal{I}}_t = \hat{\mathcal{I}}_{s_t, K_t}$$

The problem of estimating (s_t, K_t) can naturally be posed as a minimization problem:

$$\min_{\substack{s \in [0,1] \\ K \in GL(3)}} \sum_{w \in \mathcal{P}_t} \left(\hat{\mathcal{I}}_{s,K}(w) - \mathcal{I}_t(w) \right)^2$$

Solving this problem for s and K simultaneously is difficult due to the complicated dependence of $\hat{\mathcal{I}}_{s,K}$ on s through the correspondence relating $(\mathcal{P}_0, \mathcal{P}_1)$. In practice, we separate the minimization problem as:

$$\min_{s \in [0,1]} \min_{K \in GL(3)} \sum_{w \in \mathcal{P}_t} \left(\hat{\mathcal{I}}_{s,K}(w) - \mathcal{I}_t(w) \right)^2$$

Since s is fixed for the interior problem, it is simply the problem of finding the best projective transformation relating $\hat{\mathcal{I}}_{s,I}$ and $\mathcal{I}_t$. We can apply the efficient estimation algorithms discussed in Chapter 4. When $\hat{\mathcal{I}}_{s,I}$ and $\mathcal{I}_t$ are similar, automatic feature extraction techniques such as the one described in Section 3.4 can be applied to obtain data points for the optimizations. However, depending on the choice of rectifying projective transformations (G, H), the synthetic image that results from view morphing may have a much different orientation than the frame we wish to predict. For this reason, we apply an initial projective transformation to $\hat{\mathcal{I}}_{s,I}$ to better align the two images for feature matching. The previous estimate of $K_{t-\Delta t}$ is a natural choice (at $t = 0$, we have the zero-error solution $K_0 = G^{-1}$). The same applies to the search neighborhood for $\hat{s}_t$; since we assume the camera motion is continuous and its velocity is bounded, we only need to refine the initial estimate $\hat{s}_t = \hat{s}_{t-\Delta t}$. This leads to our final algorithm for the estimation of $(\hat{s}_t, \hat{K}_t)$:

Algorithm 7.2: *View morphing for matching video frames.*

1. *Initialize* $\hat{s}_0 = 0$, $\hat{K}_0 = G^{-1}$.

2. *For* $t = \Delta t, 2\Delta t, \ldots, N\Delta t$*:*

   (a) *Fix a search range* $[s_{\min}, s_{\max}]$ *about* $\hat{s}_{t-\Delta t}$.

   (b) *Initialize* $\hat{s}_t = \hat{s}_{t-\Delta t}$, $\hat{K}_t = \hat{K}_{t-\Delta t}$, $err = \infty$.

   (c) *For* $s \in [s_{\min}, s_{\max}]$*:*

      i. *Construct the virtual image* $\hat{\mathcal{I}}_{s,\hat{K}_{t-\Delta t}}$.

      ii. *Extract matching features between* $\hat{\mathcal{I}}_{s,\hat{K}_{t-\Delta t}}$ *and* $\mathcal{I}_t$.

      iii. *Estimate the projective transformation* $\tilde{K}$ *relating* $\hat{\mathcal{I}}_{s,\hat{K}_{t-\Delta t}}$ *and* $\mathcal{I}_t$ *using the extracted features.*

      iv. *Construct the virtual image* $\hat{\mathcal{I}}_{s,\tilde{K}\hat{K}_{t-\Delta t}}$.

      v. *If* $\displaystyle\sum_{w \in \mathcal{P}_t} \left(\hat{\mathcal{I}}_{s,\tilde{K}\hat{K}_{t-\Delta t}}(w) - \mathcal{I}_t(w)\right)^2 < err$,

         A. $err \leftarrow \displaystyle\sum_{w \in \mathcal{P}_t} \left(\hat{\mathcal{I}}_{s,\tilde{K}\hat{K}_{t-\Delta t}}(w) - \mathcal{I}_t(w)\right)^2$

         B. $\hat{s}_t \leftarrow s$

         C. $\hat{K}_t \leftarrow \tilde{K}\hat{K}_{t-\Delta t}$

If the video to be interpolated is in MPEG format, an alternative initial estimate of s_t could be obtained by rapidly estimating the magnitude and direction of camera motion from MPEG motion vectors [10]. Experimental results from our algorithm are demonstrated in the next section.

7.5 Experimental Results

We applied our interpolation method to a 180-frame, single-shot, 24-bit color, 320 x 240 test sequence captured with a digital video camera. The first and last frames (Figure 7.5) were designated as anchor frames, and the remaining 178 frames were designated as "V" frames to be interpolated. The camera motion is roughly linear, though the speed is not uniform. From Figure 7.5 we can see that the perspective difference between the anchor frames is substantial, and that a block from

(a) (b)

Figure 7.5: The two anchor frames: (a) Frame 0, (b) Frame 179.

an intermediate frame would probably have a poor (i.e. high error) match in either of the anchor frames.

The anchor frames with the 66 feature points used to initialize the epipolar geometry and the 28 control lines used to initialize the correspondence are illustrated in Figure 7.6. To achieve maximal compression, we did not refine the correspondence obtained by linearly interpolating between the control lines. An additional set of features was used to estimate the projective transformations corresponding to the left and right walls (Figure 7.7). These are used to determine the correct starting and ending points in the conjugate epipolar line matching graphs.

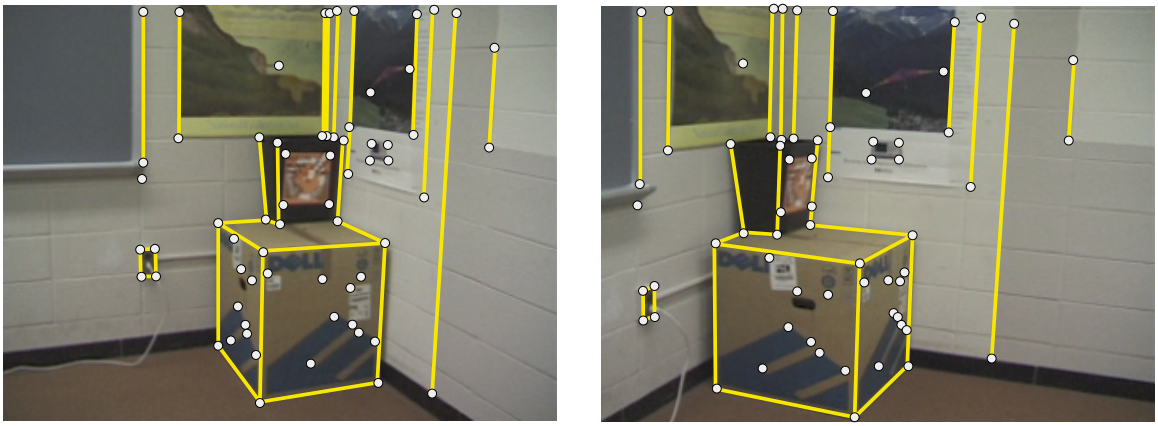


Figure 7.6: Anchor frames, with feature points and control lines used to initialize correspondence.

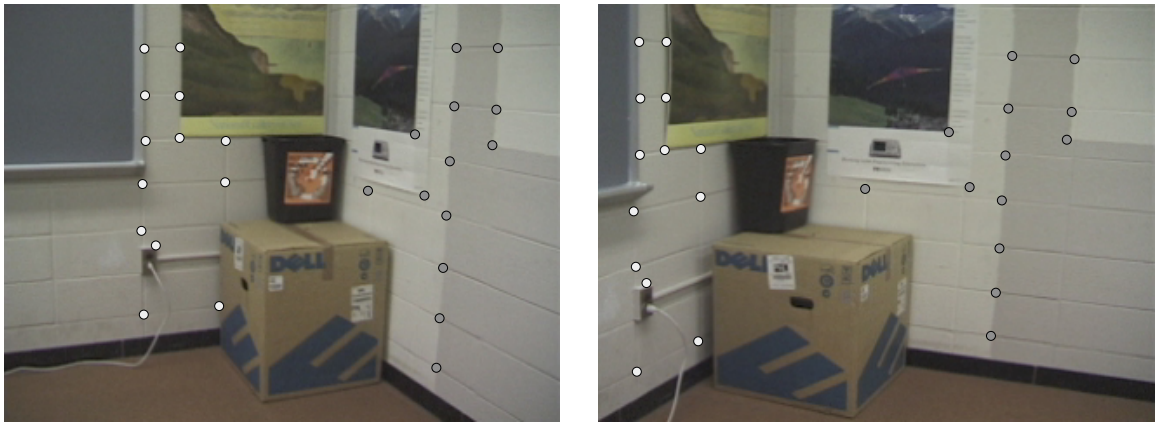


Figure 7.7: Anchor frames, with feature points used to specify the left (light-colored dots) and right (dark-colored dots) walls.

In this example, we set the search neighborhood for $\hat{s}_t$ to be

$$[\hat{s}_{t-\Delta t} - 0.05, \hat{s}_{t-\Delta t} + 0.05] \quad \cap \quad [0, 1]$$

This interval was sampled in steps of $\frac{1}{180} = 0.0056$. The resulting estimates $\hat{s}_t$ for the entire sequence are displayed in Figure 7.8.

Figure 7.9 illustrates the original, interpolated, and luminance difference frames for frames taken at every second (30 frames) of the test sequence. From the difference images we can see that the interpolated images align quite well with the original frames of video. The errors around edges are largely due to the blurriness of the virtual images introduced by several steps of image resampling. The other major artifacts are the black regions around the borders of the interpolated images that correspond to areas of the virtual frame visible in neither of the anchor frames. In this example, these areas are not too large and could be filled in by the type of error-concealment algorithms devised for other video compression schemes. The total file size of the information required to interpolate is roughly 35400 bytes (18000 bytes total for the two JPEG-coded anchor frames, 11000 bytes for the compressed correspondence information, 36 bytes for the parameters of each virtual frame). Clearly the number of interpolated frames has a negligible effect on the size of the transmitted data. For video in which the camera moves slowly along an approximately piecewise linear path, we therefore expect reasonable performance for a small amount of side information. The total bit rate in this example is 47.22 kbps.

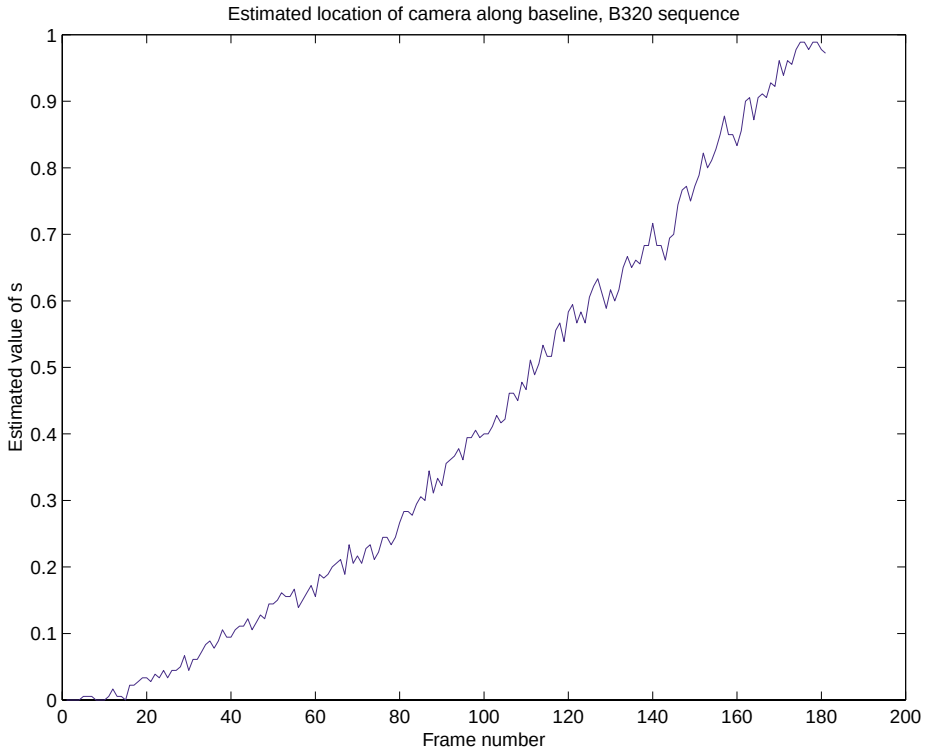


Figure 7.8: Estimates of the camera location s_t.

Figure 7.10 displays the luminance peak signal-to-noise ratio (PSNR) over the video sequence, defined for each frame as:

$$\mathrm{PSNR} = 10 \log_{10} \frac{1}{|R|} \sum_{w \in R} \frac{255^2}{\left(\mathcal{I}_t(w) - \hat{\mathcal{I}}_t(w) \right)^2}$$

where R is the subset of $\mathcal{P}_t$ that is visible in both of the anchor frames. The luminance images are assumed to have pixel intensities ranging from 0 to 255. The PSNR is lowest in the middle of the sequence, which stands to reason since the images here are the least similar to the anchor frames. The mean PSNR over the entire sequence is 30.2 dB, which seems competitive with the H.263 simulations reported in [11, Fig. 12]. The PSNR could be increased by reducing the blurriness of the virtual images through postprocessing or by reducing the number of image resampling steps (see Section 7.6). We also note that we have made little effort to represent the correspondence graph estimates and (s_t, K_t) pairs in highly compressed forms that reflect the correlations between adjacent epipolar line pairs and nearby frames. Better compression of these quantities would also increase the PSNR.

For comparison, we constructed an MPEG-1 video of the same test sequence, constrained to have the smallest bit rate the coder would produce. In this case, the size of the MPEG video is more than twice as large as the amount of information required for view morphing, at 111 KB, for a bit rate of 148 kbps. At this bit rate, the MPEG blocking and compression artifacts are severe, especially in high-detail, perceptually significant areas of the image. In contrast, the virtual image is well-defined and relatively sharper in these areas. This can be seen clearly in the close-ups of a typical frame shown in Figures 7.11a-7.11c. We note that for comparison, this MPEG video has a mean luminance PSNR of 33.1 dB. This is not surprising, since the bit rate is higher, and MPEG video is designed to minimize the mean-squared-error between original and reconstructed blocks. Figure 7.11 confirms that PSNR may be mathematically convenient, but it is a poor measure of perceptual quality.

In addition to comparing our results with MPEG-1 coded video, as illustrated here, we hope to process our test video with an MPEG-4 or H.263 encoder that is specifically targeted for low-bit-rate applications. However, at this time, there is no fully-featured source code for such encoders available for public-domain use.

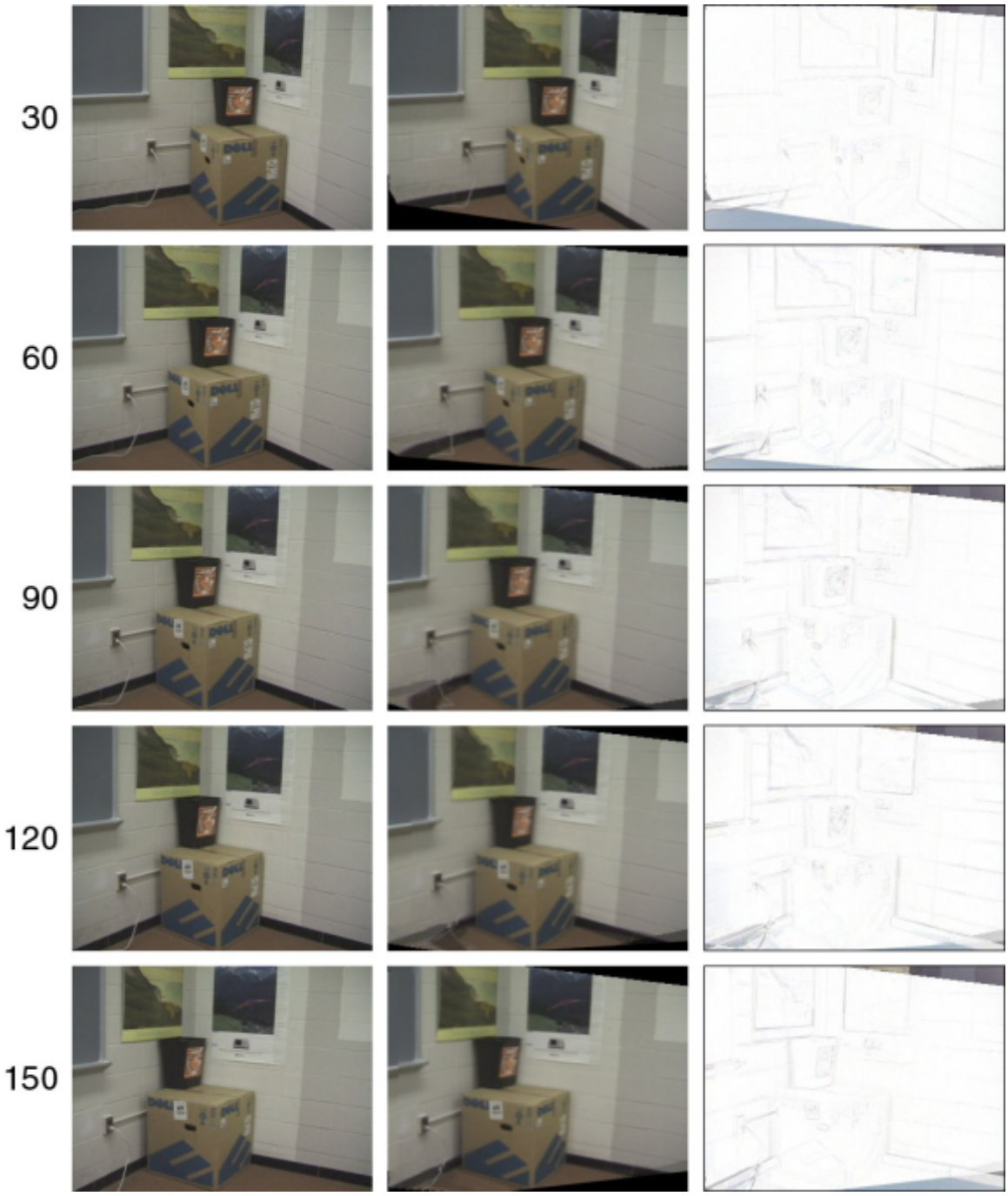


Figure 7.9: Frames 30, 60, 90, 120, and 150 of the B320 sequence. Left column: original images. Middle column: interpolated images. Right column: grayscale error images.

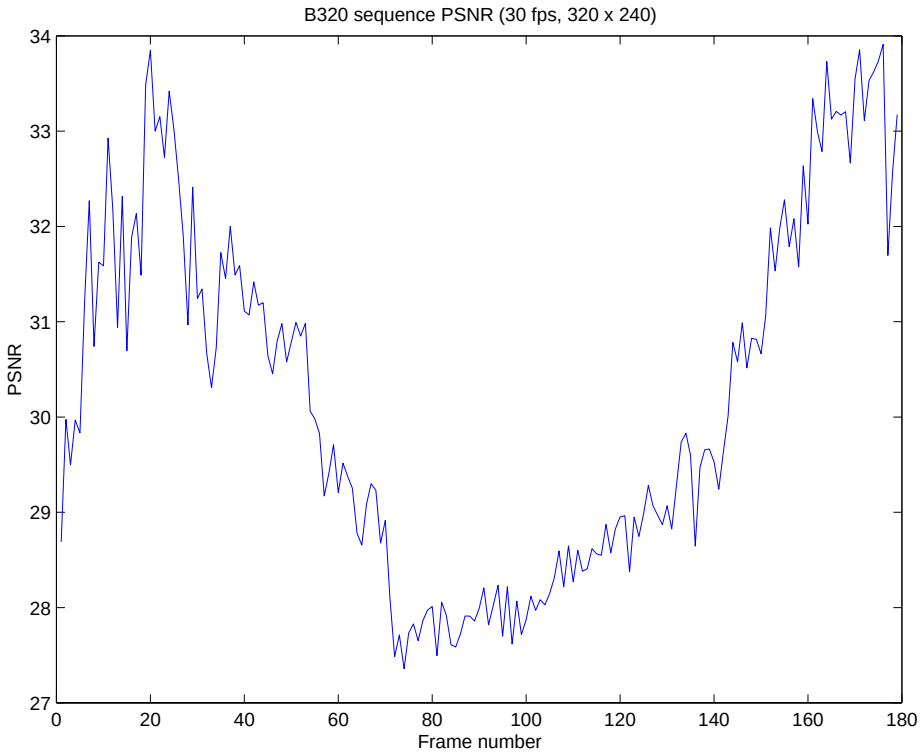


Figure 7.10: Peak signal to noise ratio over the test video sequence.

(a)

(b)

(c)

Figure 7.11: (a) Detail, original frame 90, (b) Detail, interpolated frame 90, (c) Detail, MPEG frame 90.

7.6 Conclusions

We suggest that methods, like the one introduced here, that exploit the relationship between camera motion and image correspondences can be profitably incorporated into a low bit-rate video scheme. As the technology matures and users expect more content over wireless multimedia devices, techniques that match the characteristics of the wireless multimedia network are necessary. The methods described here are not proposed as a final solution to the problem, or as a substitute for traditional video compression techniques at higher bit rates. However, we hope that these ideas provide a starting point for continuing research on more general video, and that tools such as these will be incorporated into future compression standards.

As demonstrated above, our approach is well-suited to wireless low bit-rate video in special cases of camera motion, and can be used to increase frame rate without much overhead. The computationally expensive (about 5 frames per minute) step of estimating parameters and encoding the video can be done once at the multimedia server, and amortized over a large number of downloads. The low bit-rate (e.g. 45 kbps) data stream can be easily transmitted over a wireless channel. The video can be rendered at the multimedia client for a low computational cost, since each rendered pixel is simply a weighted average of two pixels from the source images. The low complexity of the algorithm translates to low power consumption for a wireless device. Since power is also necessary for the error-correction decoding that would be required for a wireless channel, having a computationally simple reconstruction algorithm is even more important.

We emphasize that the algorithm is much less expensive if the source video comes from computer-generated imagery for which correspondence and camera motion information is easily obtained. Levoy [12] and Wallach et al. [13] made similar observations that using correspondence information for computer-generated video has significant benefits in MPEG compression. In the future, we would like to modify a ray-tracer to supply image correspondence and camera motion information to demonstrate these effects.

There are many improvements that could be made to the implementation of our coder, and several interesting research problems to address. The biggest problem with our current algorithm is the perceptually distracting jitter in the reconstructed video, since the estimation of (s_t, K_t) is

essentially independent for each frame. Even though the synthetic frames are individually good approximations to the original frames they represent, the result does not convey a sense of fluid camera motion as well as it could. Of course, since the original camera motion may be jerky, imposing too many smoothness constraints on the estimation could be unwise. This is a topic of current research.

The second most distracting artifact is the slight blurriness of the reconstructed images compared to the original video frames. This is caused by three steps of image resampling in our current implementation:

1. Applying the rectifying projective transformations (G, H) to $(\mathcal{I}_0, \mathcal{I}_1)$
2. Constructing $\hat{\mathcal{I}}_{s_t,I}$ with the view morphing equation
3. Applying the alignment projective transformation K_t to $\hat{\mathcal{I}}_{s_t,I}$ to construct $\hat{\mathcal{I}}_{s_t,K_t}$

The first and third resampling operations could be eliminated by removing the dependence of our rendering algorithm on explicitly rectified images. This leaves step 2 as the main rendering step, which as written in (7.2) is simply a weighted average of the intensities of two pixels from the original source images. Of course, the "pixels" might not have integer coordinates, which would require another step of bilinear interpolation. If even more sharpness is required, instead of using a weighted average in (7.2), the resampled image could just take its intensity from one or the other of the anchor frames. This would trade off sharpness for a lack of robustness to illumination changes in the scene. Additionally, standard post-processing techniques (e.g. unsharp masking) could be used to improve the perceptual quality of the video at the decoder.

Clearly, the success of view-interpolation-based methods depends greatly on the selection of good anchor frames. The automatic selection of such frames based on estimated camera dynamics is one of the next research problems we wish to consider.

We have not addressed the case when objects are moving in the scene independently of the camera. However, our method can be viewed as a generalization of one feature of the MPEG-4 standard, which allows a region of pixels to be designated as the "background", and warped projectively as if it lies on a plane. The foreground moving objects (called "sprites" in MPEG-4) can then be replaced on the warped background.

Finally, we comment on the type of unequal error protection that would be required for this scheme. The most important information in the video representation are, in order:

1. The parameters (s_t, K_t): a bit error here affects the entire reconstructed image $\hat{\mathcal{I}}_t$.
2. The estimated correspondence between $(\mathcal{I}_0, \mathcal{I}_1)$: a bit error here affects the same epipolar line in each reconstructed image
3. The images $(\mathcal{I}_0, \mathcal{I}_1)$ themselves: if they are JPEG compressed, a bit error here affects an 8×8 block of pixels, which may cause errors in a few pixels in each reconstructed image.

Of course, any video compression scheme based on variable length coding is equally sensitive to bit errors, especially to additions and deletions of bits.

7.7 References

[1] R-S. Wang and Y. Wang. Multiview Video Sequence Analysis, Compression, and Virtual Viewpoint Synthesis. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 3, pp. 397–410, April 2000.

[2] R. Radke, P. Ramadge, S. Kulkarni, and T. Echigo. Using View Interpolation for Low Bit-Rate Video. In *Proc. ICIP 2001*, Greece, October 2001.

[3] D. LeGall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, April 1991.

[4] ISO/IEC 13818. MPEG-2 International Standard, Information technology, Generic Coding of Moving Pictures and Associated Audio Information.

[5] T. Sikora. The MPEG-4 Video Standard Verification Model. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 19–31, 1997.

[6] A.M. Tekalp, P. van Beek, C. Toklu, and B. Günsel. Two-Dimensional Mesh-Based Visual-Object Representation for Interactive Synthetic/Natural Digital Video. *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1029–1051, June 1998.

[7] CCITT Recommendation H.261, Video Codec For Audiovisual Services At $p \times 64$ kbit/s, Genf, 1990.

[8] G. Cote, B. Erol, M. Gallant, and F. Kossentini. H.263+: Video Coding at Low Bit Rates. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, November 1998.

[9] Y. Wang and Q-F. Zhu. Error Control and Concealment for Video Communication: A Review. *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, May 1998.

[10] Y.P. Tan, D. D. Saur, S. R. Kulkarni, and P. J. Ramadge. Rapid Estimation of Camera Motion from Compressed Video With Application to Video Annotation. *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 133–146, February 2000.

[11] L. Hanzo. Bandwidth-Efficient Wireless Multimedia Communications. *Proceedings of the IEEE*, vol. 86, no. 7, pp. 1342–1382, July 1998.

[12] M. Levoy. Polygon-Assisted JPEG and MPEG Compression of Synthetic Images. *Computer Graphics (SIGGRAPH '95)*,pp. 21–28, August 1995.

[13] D. Wallach, S. Kunapalli, and M. Cohen. Accelerated MPEG Compression of Dynamic Polygonal Scenes. *Computer Graphics (SIGGRAPH '94)*, pp. 196–196, August 1994.

Chapter 8

Conclusions

Throughout this thesis, we returned frequently to the estimation and application of projective transformations. In Chapter 4, we posed the projective transformation estimation problem as a parameter estimation problem over noisy point matches. In prior work, this estimation problem is either simplified using affine or small-motion assumptions to produce a linear least-squares problem, or solved directly as a nonlinear minimization using numerical methods. We showed with extensive analysis that the cost function associated with the projective transformation estimate has considerable structure, and that this structure can be exploited to construct efficient minimization algorithms that are robust to measurement noise. Our minimization algorithms were shown to constitute a substantial improvement over the off-the-shelf methods that are typically used.

The correspondence induced between two image planes by a projective transformation is only a subset of the set of correspondences that can arise from two views of a scene taken by physical cameras. This set is rich and complicated, and much prior work on estimating correspondence only searches over the subset of monotonic correspondence, for which the ordering of correspondences along conjugate epipolar lines is invariant. In Chapter 5, we ventured beyond monotonic correspondence and fully characterized the structure of the elements of the entire set of viable correspondences. This led us to the problem of estimating correspondence in the general setting, and we showed how the formalism of correspondence graphs can be used to ensure that any estimated correspondence is consistent with a physical imaging system.

In Chapter 6, we used our correspondence estimates to synthesize virtual views of a scene from wide-baseline still images using view morphing. Since the estimated correspondences are both

geometrically and photometrically accurate, the synthetic images are realistic perspective views of the scene, rendered from novel camera positions. We then posed our third estimation problem, that of efficiently estimating correspondence between two video sequences. We exploited the temporal regularity of video to create a recursive framework for the propagation of correspondence estimates. We proved the stability of our algorithm in theory, and demonstrated its application to real video. The result is compelling virtual video that evolves dynamically along with the scene, constructed with minimal input from the user. This is a substantial improvement over the previous state of the art, which required either a static scene or an expensive hardware assembly to produce a similar effect.

Finally, in Chapter 7, we showed how virtual images, sometimes viewed only as graphical special effects, can be made useful in the context of time-domain interpolation of video frames for low-bitrate applications. Our final estimation problem was to synthesize the virtual image between two frames of video that best matches a real intermediate frame. Understanding the geometric connections between the positions of cameras and image correspondences allows us to store fewer intracoded frames and to synthesize realistic intermediate images that may resemble none of the intracoded frames. This is an exciting topic of current research that we hope will have applications in wireless multimedia.

The four main estimation problems in digital video we considered in this thesis, and the applications we discussed, are summarized in Table 8.1.

All of the estimation problems in this thesis are tied to the age-old problem of correspondence. In Chapter 5, we considered the fundamental question, “What sets are valid correspondences?”, which led to the definition of the correspondence graph. We might ask other questions. How can we quantify the “complexity” of a (scene,camera) pair? Is there a sampling theorem for scenes and cameras? That is, if we want enough information to describe a given scene with some fixed degree of accuracy, how finely do we need to space a set of perspective cameras? Conversely, for a fixed scene and finite number of cameras, where should the cameras be placed to be able to render the largest possible set of virtual views with some fixed degree of accuracy? The formalism of the correspondence graph may help answer some of these questions.

As we showed in Chapters 6 and 7, an estimate of correspondence that is consistent with the

Chapter	Estimation Problem	Application
4	Projective transformations from noisy point correspondences	Image and video mosaics
5	Correspondence between a general pair of image planes	Wide-baseline virtual images
6	Correspondence between a pair of video sequences of the same scene taken by rotating cameras	Virtual video
7	Position and orientation of a camera as it moves along a linear path during a video sequence	Virtual time-domain interpolants of real video

Table 8.1: Thesis contributions.

underlying geometry of a scene can have significant benefits in visualization and video coding. We contend that geometric correspondence is inherently superior to photometric correspondence in applications where a notion of physical correctness or consistency is important. Most video coding algorithms can match up points arbitrarily, disregarding the epipolar constraint, much less considerations of physical consistency. This may be good for mean-squared error over short time intervals, but it ignores the long-term connections between image correspondences that are induced by compactly parameterizable camera motion. When coding efficiency is an issue, we feel that more redundancy can be removed by exploiting geometric constraints using image-based algorithms. As a side benefit, the rendering component of an image-based algorithm is typically less computationally demanding than a transform-coding based algorithm such as MPEG decompression.

As applications that require a 3-D sense of scene develop, we expect that algorithms, such as the ones described in this thesis, that are based on well-founded estimates of parameters of camera motion will become increasingly important. We envision an algorithm that takes into account estimates of camera motion parameters, requirements for rendering quality, and constraints on bit rate, and decides that some sequences of video frames are best coded by a projective warping of a single frame, others can be synthesized using virtual view techniques, and so on, to create a hybrid video coder that operates in several modes.

Finally, we note that many research problems at the interface between equations and silicon must

be solved to bring the estimation algorithms we propose here from a powerful desktop computer to a portable wireless device. Issues of power control, fixed-point arithmetic, limited color depth, low available memory, and robust error correction would all be involved. Bringing effective and efficient video processing to wireless multimedia will challenge researchers for years to come.

Appendix A

Proof of Theorem 4.2

1. We split up the expressions (4.10)-(4.12) by separating out the first point (and noting that $q_1 = \alpha$):

$$\begin{aligned} W(c^* + \alpha h) &= \begin{bmatrix} \frac{w_1 w_1^T}{\alpha^2} + \sum_{j=2}^N \frac{w_j w_j^T}{q_j^2(\alpha)} & \frac{w_1}{\alpha^2} + \sum_{j=2}^N \frac{w_j}{q_j^2(\alpha)} \\ \frac{w_1^T}{\alpha^2} + \sum_{j=2}^N \frac{w_j^T}{q_j^2(\alpha)} & \frac{1}{\alpha^2} + \sum_{j=2}^N \frac{1}{q_j^2(\alpha)} \end{bmatrix} \\ V(c^* + \alpha h) &= \begin{bmatrix} \frac{w_1' w_1^T}{\alpha} + \sum_{j=2}^N \frac{w_j' w_j^T}{q_j(\alpha)} & \frac{w_1'}{\alpha} + \sum_{j=2}^N \frac{w_j'}{q_j(\alpha)} \end{bmatrix} \end{aligned}$$

Here $q_j(\alpha) = (c^* + \alpha h)^T w_j + 1$. We now rewrite the defining equation (4.9) as:

$$\begin{bmatrix} A(\alpha) & b(\alpha) \end{bmatrix} \left[\frac{1}{\alpha^2} \begin{bmatrix} w_1 w_1^T & w_1 \\ w_1^T & 1 \end{bmatrix} + W_2(\alpha) \right] = \left[\frac{1}{\alpha} w_1' p^T + V_2(\alpha) \right] \tag{A.1}$$

where

$$\begin{aligned} W_2(\alpha) &= \begin{bmatrix} \sum_{j=2}^N \frac{w_j w_j^T}{q_j^2(\alpha)} & \sum_{j=2}^N \frac{w_j}{q_j^2(\alpha)} \\ \sum_{j=2}^N \frac{w_j^T}{q_j^2(\alpha)} & \sum_{j=2}^N \frac{1}{q_j^2(\alpha)} \end{bmatrix} \\ V_2(\alpha) &= \begin{bmatrix} \sum_{j=2}^N \frac{w_j' w_j^T}{q_j(\alpha)} & \sum_{j=2}^N \frac{w_j'}{q_j(\alpha)} \end{bmatrix} \end{aligned}$$

We note that as $\alpha \to 0$, $W_2(\alpha)$ and $V_2(\alpha)$ converge to well-defined finite matrices $W_2(0)$ and $V_2(0)$. In the following, we will use the notations W_2, V_2 with the understanding that they are functions of α. While it is true that for $\alpha > 0$, $W_2(\alpha)$ and $V_2(\alpha)$ are also functions of h, the limiting values $W_2(0)$ and $V_2(0)$ are independent of h.

Taking (A.1) and isolating $A(\alpha)$ and $b(\alpha)$ on the left-hand side, we have

$$\left[\begin{array}{cc} A(\alpha) & b(\alpha) \end{array}\right] = \left[\frac{1}{\alpha} w_1' p^T + V_2\right] \left[\frac{1}{\alpha^2} \left[\begin{array}{cc} w_1 w_1^T & w_1 \\ w_1^T & 1 \end{array}\right] + W_2\right]^{-1} \tag{A.2}$$

Here we have introduced the abbreviation $p = [w_1^T 1]^T$. First consider the matrix that is inverted in (A.2) above. Using the matrix inversion lemma, we can write

$$\begin{aligned} \left[\frac{1}{\alpha^2} \left[\begin{array}{cc} w_1 w_1^T & w_1 \\ w_1^T & 1 \end{array}\right] + W_2\right]^{-1} &= \left[\frac{1}{\alpha} p \frac{1}{\alpha} p^T + W_2\right]^{-1} \\ &= W_2^{-1} - \frac{1}{\alpha^2} W_2^{-1} p \left[1 + \frac{1}{\alpha^2} p^T W_2^{-1} p\right]^{-1} p^T W_2^{-1} \\ &= W_2^{-1} - W_2^{-1} p \left[\alpha^2 + p^T W_2^{-1} p\right]^{-1} p^T W_2^{-1} \\ &= W_2^{-1} - \frac{W_2^{-1} p p^T W_2^{-1}}{\alpha^2 + p^T W_2^{-1} p} \\ &= W_2^{-1} \left[I - \frac{p p^T W_2^{-1}}{\alpha^2 + p^T W_2^{-1} p}\right] \end{aligned}$$

Therefore, we can rewrite (A.2) as

$$\begin{aligned} \left[\begin{array}{cc} A(\alpha) & b(\alpha) \end{array}\right] &= \left[\frac{1}{\alpha} w_1' p^T + V_2\right] W_2^{-1} \left[I - \frac{p p^T W_2^{-1}}{\alpha^2 + p^T W_2^{-1} p}\right] \\ &= w_1' p^T W_2^{-1} \left[\frac{\frac{1}{\alpha}\left(\alpha^2 + p^T W_2^{-1} p\right) I - \frac{1}{\alpha} p p^T W_2^{-1}}{\alpha^2 + p^T W_2^{-1} p}\right] \\ &\quad + V_2 W_2^{-1} \left[I - \frac{p p^T W_2^{-1}}{\alpha^2 + p^T W_2^{-1} p}\right] \\ &= w_1' p^T W_2^{-1} \frac{\alpha}{\alpha^2 + p^T W_2^{-1} p} + V_2 W_2^{-1} \left[I - \frac{p p^T W_2^{-1}}{\alpha^2 + p^T W_2^{-1} p}\right] \end{aligned}$$

Letting $\alpha \to 0$ in the above, we obtain

$$\begin{aligned} \left[\begin{array}{cc} A_o & b_o \end{array}\right] &= \lim_{\alpha \to 0} \left[\begin{array}{cc} A(\alpha) & b(\alpha) \end{array}\right] \\ &= V_2 W_2^{-1} \left[I - \frac{p p^T W_2^{-1}}{p^T W_2^{-1} p}\right] \end{aligned} \tag{A.3}$$

2. Consider the minimization problem

$$\begin{aligned} \min_{A,b} \quad & \frac{1}{2}\sum_{j=2}^{N}\left(w_j' - \frac{Aw_j+b}{c^{*T}w_j+1}\right)^T\left(w_j' - \frac{Aw_j+b}{c^{*T}w_j+1}\right) \\ s.t. \quad & Aw_1 + b = 0 \end{aligned} \tag{A.4}$$

The normal equations for the constrained problem are:

$$\begin{aligned} A\sum_{j=2}^{N}\frac{w_j w_j^T}{(c^{*T}w_j+1)^2} + b\sum_{j=2}^{N}\frac{w_j^T}{(c^{*T}w_j+1)^2} - \sum_{j=2}^{N}\frac{w_j' w_j^T}{c^{*T}w_j+1} + \lambda w_1^T &= 0 \\ A\sum_{j=2}^{N}\frac{w_j}{(c^{*T}w_j+1)^2} + b\sum_{j=2}^{N}\frac{1}{(c^{*T}w_j+1)^2} - \sum_{j=2}^{N}\frac{w_j'}{c^{*T}w_j+1} + \lambda &= 0 \\ Aw_1 + b &= 0 \end{aligned}$$

Here λ is a Lagrange multiplier in $\mathbb{R}^2$. Rewriting these normal equations in the notation of the previous section, we obtain

$$\begin{bmatrix} A & b \end{bmatrix} = V_2 W_2^{-1} - \lambda p^T W_2^{-1} \tag{A.5}$$

$$\begin{bmatrix} A & b \end{bmatrix} p = 0 \tag{A.6}$$

It is easy to see that (A.5) is satisfied by the choice of (A_o, b_o) in (A.3), with

$$\lambda = \frac{V_2 W_2^{-1} p}{p^T W_2^{-1} p}$$

Furthermore, (A_o, b_o) satisfy the constraint equation (A.6). Hence, by uniqueness of the solution of the linear least squares problem, we conclude that along singular lines, the solution of (4.9) converges to the solution of the constrained minimization problem (A.4) posed over the data set minus the offending point.

3. From the expression (A.3), we can also obtain an expression for the point to which (A_o, b_o, c^*) maps w_1:

$$\begin{aligned}
\lim_{\alpha\to 0} \frac{A(\alpha)w_1 + b(\alpha)}{(c^* + \alpha h)^T w_1 + 1} &= \lim_{\alpha\to 0} \frac{1}{\alpha}[\ A(\alpha) \quad b(\alpha)\ ]p \\
&= \lim_{\alpha\to 0} \frac{1}{\alpha} w_1' p^T W_2^{-1} \frac{\alpha}{\alpha^2 + p^T W_2^{-1} p} \\
&\quad + V_2 W_2^{-1} \left[I - \frac{p p^T W_2^{-1}}{\alpha^2 + p^T W_2^{-1} p} \right] p \\
&= \lim_{\alpha\to 0} w_1' p^T W_2^{-1} \frac{p}{\alpha^2 + p^T W_2^{-1} p} \\
&\quad + V_2 W_2^{-1} \left[\frac{1}{\alpha} p - \frac{1}{\alpha} \frac{p p^T W_2^{-1} p}{\alpha^2 + p^T W_2^{-1} p} \right] \\
&= \lim_{\alpha\to 0} w_1' \frac{p^T W_2^{-1} p}{\alpha^2 + p^T W_2^{-1} p} + V_2 W_2^{-1} \frac{\alpha p}{\alpha^2 + p^T W_2^{-1} p} \\
&= w_1'
\end{aligned}$$

Appendix B

Newton Methods

In Chapter 4, we referred to the Newton and Gauss-Newton methods for minimizing a nonlinear function of several parameters. We describe these methods now. The problem we wish to solve is:

Problem 1: *Given measurement data* $\{(x_j, y_j) \in \mathbb{R}^M \times \mathbb{R}, j = 1, \ldots, N\}$ *and a continuously differentiable function* $f : \mathbb{R}^{M+K} \to \mathbb{R}$*, determine the parameter values* $\{\hat{\theta}_1, \ldots, \hat{\theta}_K\}$ *that minimize the cost functional*

$$J(\theta_1, \ldots, \theta_K) = \sum_{j=1}^{N} (y_j - f(x_j; \theta_1, \ldots, \theta_K))^2 \tag{B.1}$$

We collect the measurement data and parameters into vectors:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \qquad \boldsymbol{\theta} = [\theta_1, \cdots, \theta_K]$$

Since the measurement data that appear inside the function f are fixed, we suppress the dependence of f on the data x_j and write

$$\mathbf{f}(\boldsymbol{\theta}) = \begin{bmatrix} f(x_1, \boldsymbol{\theta}) \\ \vdots \\ f(x_N, \boldsymbol{\theta}) \end{bmatrix}$$

The cost functional (B.1) can now be written

$$J(\boldsymbol{\theta}) = [\mathbf{y} - \mathbf{f}(\boldsymbol{\theta})]^T [\mathbf{y} - \mathbf{f}(\boldsymbol{\theta})] \tag{B.2}$$

Necessary conditions for $\boldsymbol{\theta}^*$ to minimize (B.2) are

$$\frac{\partial J}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^*) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_K} \end{bmatrix} (\boldsymbol{\theta}^*) = 0 \tag{B.3}$$

$$\frac{\partial^2 J}{\partial \boldsymbol{\theta}^2}(\boldsymbol{\theta}^*) = \begin{bmatrix} \frac{\partial^2 J}{\partial \theta_1^2} & \cdots & \frac{\partial^2 J}{\partial \theta_1 \partial \theta_K} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial \theta_1 \partial \theta_K} & \cdots & \frac{\partial^2 J}{\partial \theta_K^2} \end{bmatrix} (\boldsymbol{\theta}^*) > 0 \tag{B.4}$$

The notation in (B.4) is shorthand for the positive definiteness of the matrix $\frac{\partial^2 J}{\partial \boldsymbol{\theta}^2}(\boldsymbol{\theta}^*)$.

The condition (B.3) can be written:

$$\frac{\partial J}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^*) = -2[\mathbf{y} - \mathbf{f}(\boldsymbol{\theta}^*)]^T \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^*) = 0$$

or equivalently,

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}^T (\boldsymbol{\theta}^*)[\mathbf{y} - \mathbf{f}(\boldsymbol{\theta}^*)] = 0 \tag{B.5}$$

Note that $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}$ is an $N \times K$ matrix. When $\mathbf{f}$ is a linear function of the parameters $\boldsymbol{\theta}$ given by $\mathbf{f}(\boldsymbol{\theta}) = F\boldsymbol{\theta}$, $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}$ is a matrix $F \in \mathbb{R}^{N \times K}$ that doesn't depend on $\boldsymbol{\theta}$, and (B.5) is a linear equation in $\boldsymbol{\theta}^*$, the solution of which is:

$$\hat{\boldsymbol{\theta}} = (F^T F)^{-1} F^T \mathbf{y}$$

However, in general, (B.5) is a nonlinear system of equations in $\boldsymbol{\theta}$ that must be solved by numerical means.

We begin by expanding the cost function $J(\boldsymbol{\theta})$ in a Taylor series about some point $\boldsymbol{\theta}^*$:

$$J(\boldsymbol{\theta}) = J(\boldsymbol{\theta}^*) + \frac{\partial J}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^*)(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \frac{\partial^2 J}{\partial \boldsymbol{\theta}^2}(\boldsymbol{\theta}^*)(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + h.\ o.\ t. \tag{B.6}$$

$$\begin{aligned} \frac{\partial^2 J}{\partial \boldsymbol{\theta}^2}(\boldsymbol{\theta}^*) &= -2\left[\sum_{k=1}^{N} [y_k - f(x_k; \boldsymbol{\theta})] \frac{\partial^2 f(x_k; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2}(\boldsymbol{\theta}^*)\right] + 2\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^*)^T \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^*) \\ &= -2\left[\sum_{k=1}^{N} [y_k - f(x_k; \boldsymbol{\theta})] \frac{\partial^2 f(x_k; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2}(\boldsymbol{\theta}^*)\right] + 2H^T H \end{aligned} \tag{B.7}$$

In (B.7) we have denoted $H = \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^*)$. Ignoring the higher order terms in (B.6), minimizing the resulting quadratic with respect to $\boldsymbol{\theta}$, and solving for the minimizer gives

$$\boldsymbol{\theta} = \boldsymbol{\theta}^* - \left[\frac{\partial^2 J}{\partial \boldsymbol{\theta}^2}(\boldsymbol{\theta}^*)\right]^{-1} \frac{\partial J}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^*)^T \tag{B.8}$$

When we add a step size parameter to (B.8), and substitute the expansion (B.7), we obtain:

$$\boldsymbol{\theta} = \boldsymbol{\theta}^* + \rho \left[H^T H - \sum_{k=1}^{N} [y_k - f(x_k;\boldsymbol{\theta})] \frac{\partial^2 f(x_k;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2}(\boldsymbol{\theta}^*)\right]^{-1} H^T[\mathbf{y} - f(\boldsymbol{\theta}^*)] \tag{B.9}$$

Under certain conditions, if $\boldsymbol{\theta}^*$ is a good estimate of the minimizer $\hat{\boldsymbol{\theta}}$, the parameters $\boldsymbol{\theta}$ produced by (B.9) are an incrementally better estimate of $\hat{\boldsymbol{\theta}}$. The new estimate can then replace the old ($\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}^*$) and (B.9) reapplied. The iterations terminate when the term $H^T[\mathbf{y} - f(\boldsymbol{\theta}^*)]$ in (B.9) becomes vanishingly small (which is the condition (B.5)). The iteration suggested by (B.9) is called the *Newton-Raphson method*. The convergence of the algorithm is governed by the choice of the step size parameter ρ. Usually, ρ is chosen to reduce (not necessarily minimize) J along the search direction $(H^T H)^{-1} H^T(\mathbf{y} - \mathbf{f}(\boldsymbol{\theta}^*))$. This can be accomplished by a backtracking algorithm based on cubic interpolation [1].

A simplification of (B.9) is obtained by dropping the second term in the inverted matrix, producing the update equation

$$\boldsymbol{\theta} = \boldsymbol{\theta}^* + \rho (H^T H)^{-1} H^T (\mathbf{y} - \mathbf{f}(\boldsymbol{\theta}^*))$$

This simplified iteration is known as *Gauss's method* or the *Gauss-Newton method*. Gauss's method can be alternately derived by considering the modeling assumption that

$$\mathbf{y} = \mathbf{f}(\boldsymbol{\theta}) + \mathbf{v} \tag{B.10}$$

where $\mathbf{v}$ is a small noise term. By ignoring $\mathbf{v}$ and expanding (B.10) in a Taylor series about a point $\boldsymbol{\theta}^*$, we obtain

$$\begin{aligned} \mathbf{y} &= \mathbf{f}(\boldsymbol{\theta}^*) + \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^*)(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + h.\ o.\ t. \\ &= \mathbf{f}(\boldsymbol{\theta}^*) + H(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + h.\ o.\ t. \end{aligned} \tag{B.11}$$

If we ignore the higher order terms in (B.11), we obtain a linear equation in $\boldsymbol{\theta}$ that may be rewritten as:

$$\boldsymbol{\theta} = \boldsymbol{\theta}^* + (H^T H)^{-1} H^T (\mathbf{y} - \mathbf{f}(\boldsymbol{\theta}^*)) \tag{B.12}$$

When a step size parameter ρ is added to (B.12) we again obtain Gauss's method.

A third algorithm is obtained by assuming that the inverted matrix in (B.9) is equal to the identity. Substituting and adding a step size parameter, we obtain

$$\boldsymbol{\theta} = \boldsymbol{\theta}^* + \hat{\rho} H^T [\mathbf{y} - f(\boldsymbol{\theta}^*)] \tag{B.13}$$

The iteration suggested by (B.13) is the well-known *steepest descent method*. The descent direction $H^T[\mathbf{y} - f(\boldsymbol{\theta}^*)]$ is simply the gradient of the function J.

Finally, the *Levenberg-Marquardt method* is obtained as a cross between the Gauss and steepest descent methods. The Levenberg-Marquardt iteration is based on the recursion:

$$\boldsymbol{\theta} = \boldsymbol{\theta}^* + \rho \left[H^T H + \lambda I \right]^{-1} H^T [\mathbf{y} - f(\boldsymbol{\theta}^*)]$$

When $\lambda = 0$, the Levenberg-Marquardt iteration is the same as a Gauss iteration. As $\lambda \rightarrow \infty$, the Levenberg-Marquardt iteration tends to a step along the gradient. Careful tuning of the parameter λ as the algorithm progresses generally leads to a quicker convergence rate than either of the two sub-methods. The goal is an attempt to make Gauss's method globally convergent by ensuring that the "Hessian" is positive definite. This falls into a more general class of techniques called *trust region methods*.

More details about these algorithms (e.g. choice of step size, termination criteria, search directions, pitfalls) are beyond the scope of this paper and can be found in Dennis and Schnabel [1], Seber and Wild [2], Sorenson [3] and Ortega and Rheinbolt [4]. Implementations of the algorithms in the C programming language can be found in [5]. Implementational details are also discussed in [6].

References

[1] J.E. Dennis, Jr. and R.B. Schnabel. *Numerical Methods for Unconstrianed Optimization and Nonlinear Equations* SIAM, Philadelphia, 1996.

[2] G.A.F. Seber and C.J. Wild. *Nonlinear Regression*. John Wiley and Sons, 1989.

[3] H.W. Sorenson. *Parameter Estimation: Principles and Problems.* Marcel Dekker, Inc., New York, 1980.

[4] J.M. Ortega and W.C. Rheinbolt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, London, 1970.

[5] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[6] The MathWorks, Inc. *Matlab Optimization Toolbox User's Guide, Version 5.* May, 1997.

Appendix C

The Hessian of $J(c)$

Here we compute the Hessian of $J(c)$, and derive its Gauss-Newton approximation. Due to the implicit dependence of A and b on c through (4.13), the forms are quite complex.

C.1 The Hessian Itself

Recall from (4.18) we have:

$$\nabla J(c) = \sum_{j=1}^{N} \left(w_j' - \frac{A(c)w_j + b(c)}{c^T w_j + 1} \right)^T \frac{A(c)w_j + b(c)}{c^T w_j + 1} \frac{w_j}{c^T w_j + 1}$$

We rewrite this slightly as $DJ(c)(h)$ to reflect that the derivative $DJ(c)$ acts on a direction $h \in \mathbb{R}^2$ to produce a scalar by:

$$\begin{aligned} DJ(c)(h) &= \nabla J(c)^T h \\ &= \sum_{j=1}^{N} \left(w_j' - \frac{A(c)w_j + b(c)}{c^T w_j + 1} \right)^T \frac{A(c)w_j + b(c)}{c^T w_j + 1} \frac{w_j^T h}{c^T w_j + 1} \end{aligned} \tag{C.1}$$

Similarly, assuming sufficient regularity of J, the second derivative $D^2 J(c)$ acts on a direction $h \in \mathbb{R}^2$ to produce a scalar by:

$$D^2 J(c)(h) = h^T \frac{\partial^2 J}{\partial c^2} h \tag{C.2}$$

Our goal is to obtain the Hessian $\frac{\partial^2 J}{\partial c^2}$, a symmetric matrix in $\mathbb{R}^{2\times 2}$. Differentiating (C.1) with respect to c, we obtain

$$\begin{aligned} D^2J(c)(h) &= \sum_{j=1}^{N}\left(w_j' - \frac{A(c)w_j + b(c)}{c^Tw_j + 1}\right)^T D\left(\frac{A(c)w_j + b(c)}{c^Tw_j + 1}\right)(h)\frac{w_j^Th}{c^Tw_j + 1} \\ &\quad -\sum_{j=1}^{N} D\left(\frac{A(c)w_j + b(c)}{c^Tw_j + 1}\right)(h)^T \frac{A(c)w_j + b(c)}{c^Tw_j + 1}\frac{w_j^Th}{c^Tw_j + 1} \\ &\quad -\sum_{j=1}^{N}\left(w_j' - \frac{A(c)w_j + b(c)}{c^Tw_j + 1}\right)^T \frac{A(c)w_j + b(c)}{c^Tw_j + 1}\frac{(w_j^Th)^2}{(c^Tw_j + 1)^2} \end{aligned}$$

Introducing the abbreviations $\hat{w}_j = \frac{A(c)w_j + b(c)}{c^Tw_j + 1}$ and $\varepsilon_j = \left(w_j' - \hat{w}_j\right)$, we have

$$\begin{aligned} D^2J(c)(h) &= \sum_{j=1}^{N}\frac{\varepsilon_j^T}{cw_j + 1}D\hat{w}_j(h)w_j^Th - \sum_{j=1}^{N}\frac{D\hat{w}_j(h)^T\hat{w}_j(h)}{cw_j + 1}w_j^Th \\ &\quad -\sum_{j=1}^{N}\frac{\varepsilon_j^T\hat{w}_j}{(c^Tw_j + 1)^2}(w_j^Th)^2 \end{aligned}$$

Noting that

$$D\hat{w}_j(h) = \frac{DA(c)(h)w_j + Db(c)(h)}{cw_j + 1} - \frac{\hat{w}_jw_j^Th}{cw_j + 1}$$

we obtain

$$\begin{aligned} D^2J(c)(h) &= \sum_{j=1}^{N}\frac{\varepsilon_j^T(DA(c)(h)w_j + Db(c)(h))}{(c^Tw_j + 1)^2}w_j^Th - \sum_{j=1}^{N}\frac{h^Tw_j\varepsilon_j^T\hat{w}_jw_j^Th}{(c^Tw_j + 1)^2} \\ &\quad -\sum_{j=1}^{N}\left[\frac{DA(c)(h)w_j + Db(c)(h)}{cw_j + 1}\right]^T\frac{\hat{w}_j}{cw_j + 1}w_j^Th \\ &\quad +\left[\frac{\hat{w}_jw_j^Th}{cw_j + 1}\right]^T\frac{\hat{w}_j}{cw_j + 1}w_j^Th - \sum_{j=1}^{N}h^T\frac{w_j\varepsilon_j^T\hat{w}_jw_j^T}{(c^Tw_j + 1)^2}h \\ &= h^T\left[\sum_{j=1}^{N}\frac{(\hat{w}_j - 2\varepsilon_j)^T\hat{w}_j}{(c^Tw_j + 1)^2}w_jw_j^T\right]h \qquad \text{(C.3)} \\ &\quad -\sum_{j=1}^{N}(\hat{w}_j - \varepsilon_j)^T\left[\frac{DA(c)(h)w_j + Db(c)h}{cw_j + 1}\right]\frac{w_j^Th}{cw_j + 1} \qquad \text{(C.4)} \end{aligned}$$

The first term (C.3) is already in the form we seek (C.2). Now we clarify how to express the second term (C.4) in the same form. We note that the approximation to the Hessian (4.20) comes

from assuming $DA(c)$ and $Db(c)$ are identically 0 in (C.4). Here is the expansion of (C.4):

$$\begin{aligned}
&\sum_{j=1}^{N}(\hat{w}_j-\varepsilon_j)^T\left[\frac{DA(c)(h)w_j+Db(c)h}{cw_j+1}\right]\frac{w_j^Th}{cw_j+1}\\
=&\sum_{j=1}^{N}\frac{(\hat{w}_j-\varepsilon_j)^T}{(c^Tw_j+1)^2}\left[\left(\frac{\partial A}{\partial c_1}h_1+\frac{\partial A}{\partial c_2}h_2\right)w_j+\left(\frac{\partial b}{\partial c_1}h_1+\frac{\partial c}{\partial c_2}h_2\right)\right]w_j^Th\\
=&\sum_{j=1}^{N}\frac{(\hat{w}_j-\varepsilon_j)^T}{(c^Tw_j+1)^2}\left[\left(\frac{\partial A}{\partial c_1}w_j+\frac{\partial b}{\partial c_1}\right)h_1+\left(\frac{\partial A}{\partial c_2}w_j+\frac{\partial b}{\partial c_2}\right)h_2\right]w_j^Th\\
=&\sum_{j=1}^{N}\frac{w_j^Th}{(c^Tw_j+1)^2}\left[(\hat{w}_j-\varepsilon_j)^T\left(\frac{\partial A}{\partial c_1}w_j+\frac{\partial b}{\partial c_1}\right)h_1+(\hat{w}_j-\varepsilon_j)^T\left(\frac{\partial A}{\partial c_2}w_j+\frac{\partial b}{\partial c_2}\right)h_2\right]\\
=&\sum_{j=1}^{N}\frac{w_j^Th}{(c^Tw_j+1)^2}\begin{bmatrix}h_1\\h_2\end{bmatrix}^T\begin{bmatrix}(\hat{w}_j-\varepsilon_j)^T\left(\frac{\partial A}{\partial c_1}w_j+\frac{\partial b}{\partial c_1}\right)\\(\hat{w}_j-\varepsilon_j)^T\left(\frac{\partial A}{\partial c_2}w_j+\frac{\partial b}{\partial c_2}\right)\end{bmatrix}\\
=&\ h^T\left[\sum_{j=1}^{N}\frac{1}{(c^Tw_j+1)^2}\begin{bmatrix}(\hat{w}_j-\varepsilon_j)^T\left(\frac{\partial A}{\partial c_1}w_j+\frac{\partial b}{\partial c_1}\right)\\(\hat{w}_j-\varepsilon_j)^T\left(\frac{\partial A}{\partial c_2}w_j+\frac{\partial b}{\partial c_2}\right)\end{bmatrix}w_j^T\right]h \qquad \text{(C.5)}
\end{aligned}$$

It remains to actually compute $\frac{\partial A}{\partial c_i}$ and $\frac{\partial b}{\partial c_i}$. We begin with the system of defining equations

$$A\sum_{j=1}^{N}\frac{w_jw_j^T}{(c^Tw_j+1)^2}+b\sum_{j=1}^{N}\frac{w_j^T}{(c^Tw_j+1)^2}-\sum_{j=1}^{N}\frac{w_j'w_j^T}{cw_j+1}=0 \qquad \text{(C.6)}$$

$$A\sum_{j=1}^{N}\frac{w_j}{(c^Tw_j+1)^2}+b\sum_{j=1}^{N}\frac{1}{(c^Tw_j+1)^2}-\sum_{j=1}^{N}\frac{w_j'}{cw_j+1}=0 \qquad \text{(C.7)}$$

and differentiate:

$$\begin{aligned}
\frac{\partial A}{\partial c_1}\quad&\sum_{j=1}^{N}\frac{w_jw_j^T}{(c^Tw_j+1)^2}+\frac{\partial b}{\partial c_1}\sum_{j=1}^{N}\frac{w_j^T}{(c^Tw_j+1)^2}= \qquad \text{(C.8)}\\
&2A\sum_{j=1}^{N}\frac{w_jw_j^T}{(c^Tw_j+1)^3}x_j+2b\sum_{j=1}^{N}\frac{w_j^T}{(c^Tw_j+1)^3}x_j-\sum_{j=1}^{N}\frac{w_j'w_j^T}{(c^Tw_j+1)^2}x_j\\
\frac{\partial A}{\partial c_1}\quad&\sum_{j=1}^{N}\frac{w_j}{(c^Tw_j+1)^2}+\frac{\partial b}{\partial c_1}\sum_{j=1}^{N}\frac{1}{(c^Tw_j+1)^2}= \qquad \text{(C.9)}\\
&2A\sum_{j=1}^{N}\frac{w_j}{(c^Tw_j+1)^3}x_j+2b\sum_{j=1}^{N}\frac{1}{(c^Tw_j+1)^3}x_j-\sum_{j=1}^{N}\frac{w_j'}{(c^Tw_j+1)^2}x_j
\end{aligned}$$

$$\frac{\partial A}{\partial c_2} \quad \sum_{j=1}^{N} \frac{w_j w_j^T}{(c^T w_j + 1)^2} + \frac{\partial b}{\partial c_2} \sum_{j=1}^{N} \frac{w_j^T}{(c^T w_j + 1)^2} = \tag{C.10}$$

$$2A \sum_{j=1}^{N} \frac{w_j w_j^T}{(c^T w_j + 1)^3} y_j + 2b \sum_{j=1}^{N} \frac{w_j^T}{(c^T w_j + 1)^3} y_j - \sum_{j=1}^{N} \frac{w_j' w_j^T}{(c^T w_j + 1)^2} y_j$$

$$\frac{\partial A}{\partial c_2} \quad \sum_{j=1}^{N} \frac{w_j}{(c^T w_j + 1)^2} + \frac{\partial b}{\partial c_2} \sum_{j=1}^{N} \frac{1}{(c^T w_j + 1)^2} = \tag{C.11}$$

$$2A \sum_{j=1}^{N} \frac{w_j}{(c^T w_j + 1)^3} y_j + 2b \sum_{j=1}^{N} \frac{1}{(c^T w_j + 1)^3} y_j - \sum_{j=1}^{N} \frac{w_j'}{(c^T w_j + 1)^2} y_j$$

Like the normal equations (C.6)-(C.7), these are systems of linear equations that can be solved for $\frac{\partial A}{\partial c_i}$ and $\frac{\partial b}{\partial c_i}$. The equations can be simplified as:

$$\begin{bmatrix} \frac{\partial A}{\partial c_1} & \frac{\partial b}{\partial c_1} \end{bmatrix} W(c) = X(c) \tag{C.12}$$

$$\begin{bmatrix} \frac{\partial A}{\partial c_2} & \frac{\partial b}{\partial c_2} \end{bmatrix} W(c) = Y(c) \tag{C.13}$$

where $W(c) \in \mathbb{R}^{3\times 3}$, $X(c) \in \mathbb{R}^{2\times 3}$, and $Y(c) \in \mathbb{R}^{2\times 3}$ are functions of $c \in \mathbb{R}^2$ and the data points, given by:

$$W(c) = \begin{bmatrix} \sum_{j=1}^{N} \frac{w_j w_j^T}{q_j^2(c)} & \sum_{j=1}^{N} \frac{w_j}{q_j^2(c)} \\ \\ \sum_{j=1}^{N} \frac{w_j^T}{q_j^2(c)} & \sum_{j=1}^{N} \frac{1}{q_j^2(c)} \end{bmatrix}$$

$$X(c) = \begin{bmatrix} \left(2A \sum_{j=1}^{N} \frac{w_j w_j^T}{q_j^3(c)} x_j + 2b \sum_{j=1}^{N} \frac{w_j^T}{q_j^3(c)} x_j - \sum_{j=1}^{N} \frac{w_j' w_j^T}{q_j^2(c)} x_j \right)^T \\ \left(2A \sum_{j=1}^{N} \frac{w_j}{q_j^3(c)} x_j + 2b \sum_{j=1}^{N} \frac{1}{q_j^3(c)} x_j - \sum_{j=1}^{N} \frac{w_j'}{q_j^2(c)} x_j \right)^T \end{bmatrix}^T$$

$$Y(c) = \begin{bmatrix} \left(2A \sum_{j=1}^{N} \frac{w_j w_j^T}{q_j^3(c)} y_j + 2b \sum_{j=1}^{N} \frac{w_j^T}{q_j^3(c)} y_j - \sum_{j=1}^{N} \frac{w_j' w_j^T}{q_j^2(c)} y_j \right)^T \\ \left(2A \sum_{j=1}^{N} \frac{w_j}{q_j^3(c)} y_j + 2b \sum_{j=1}^{N} \frac{1}{q_j^3(c)} y_j - \sum_{j=1}^{N} \frac{w_j'}{q_j^2(c)} y_j \right)^T \end{bmatrix}^T$$

Here $q_j(c) = c^T w_j + 1$. Note that $W(c)$ in (C.12)-(C.13) is the same matrix that appears in (4.9), which leads us to write the equations for A,b, and their partials with respect to c simultaneously as

a block triangular 9×9 system of equations:

$$\begin{bmatrix} A & b & \frac{\partial A}{\partial c_1} & \frac{\partial b}{\partial c_1} & \frac{\partial A}{\partial c_2} & \frac{\partial b}{\partial c_2} \end{bmatrix} \begin{bmatrix} W_2(c) & -W_{3x}(c) & -W_{3y}(c) \\ 0 & W_2(c) & 0 \\ 0 & 0 & W_2(c) \end{bmatrix} = \begin{bmatrix} V_1(c) & -V_{2x}(c) & -V_{2y}(c) \end{bmatrix}$$

where

$$\begin{aligned} W_2(c) &= \begin{bmatrix} \sum_{j=1}^N \frac{w_j w_j^T}{q_j^2(c)} & \sum_{j=1}^N \frac{w_j}{q_j^2(c)} \\ \sum_{j=1}^N \frac{w_j^T}{q_j^2(c)} & \sum_{j=1}^N \frac{1}{q_j^2(c)} \end{bmatrix} \\ W_{3x}(c) &= \begin{bmatrix} \sum_{j=1}^N \frac{w_j w_j^T}{q_j^3(c)} x_j & \sum_{j=1}^N \frac{w_j}{q_j^3(c)} x_j \\ \sum_{j=1}^N \frac{w_j^T}{q_j^3(c)} x_j & \sum_{j=1}^N \frac{1}{q_j^3(c)} x_j \end{bmatrix} \\ W_{3y}(c) &= \begin{bmatrix} \sum_{j=1}^N \frac{w_j w_j^T}{q_j^3(c)} y_j & \sum_{j=1}^N \frac{w_j}{q_j^3(c)} y_j \\ \sum_{j=1}^N \frac{w_j^T}{q_j^3(c)} y_j & \sum_{j=1}^N \frac{1}{q_j^3(c)} y_j \end{bmatrix} \\ V_1(c) &= \begin{bmatrix} \sum_{j=1}^N \frac{w_j' w_j^T}{q_j(c)} & \sum_{j=1}^N \frac{w_j'}{q_j(c)} \end{bmatrix} \\ V_{2x}(c) &= \begin{bmatrix} \sum_{j=1}^N \frac{w_j' w_j^T}{q_j^2(c)} x_j & \sum_{j=1}^N \frac{w_j'}{q_j^2(c)} x_j \end{bmatrix} \\ V_{2y}(c) &= \begin{bmatrix} \sum_{j=1}^N \frac{w_j' w_j^T}{q_j^2(c)} y_j & \sum_{j=1}^N \frac{w_j'}{q_j^2(c)} y_j \end{bmatrix} \end{aligned}$$

C.2 The Gauss-Newton Approximation

Though we know the Hessian is symmetric, this is unclear from the expansions of the second term (C.4) or (C.5). Furthermore, it is also not obvious which terms in the Hessian comprise the Gauss-Newton approximation.

We know that the cost function $J(c)$ can be written

$$J(c) = \frac{1}{2} \sum_{j=1}^N \varepsilon_{jx}(c)^2 + \varepsilon_{jy}(c)^2$$

where

$$\begin{aligned}\varepsilon_{jx}(c) &= x'_j - \frac{a_{11}x_j + a_{12}y_j + b_1}{c_1x_j + c_2y_j + 1} \\ \varepsilon_{jy}(c) &= y'_j - \frac{a_{21}x_j + a_{22}y_j + b_2}{c_1x_j + c_2y_j + 1}\end{aligned}$$

In this form, the gradient of J is given by

$$\nabla J(c) = \sum_{j=1}^{N} \varepsilon_{jx}(c) \cdot \nabla\varepsilon_{jx}(c) + \varepsilon_{jy}(c) \cdot \nabla\varepsilon_{jy}(c) \tag{C.14}$$

In the notation of the previous section, it is straightforward to derive

$$\begin{aligned}\nabla\varepsilon_{jx}(c) &= \frac{1}{q_j(c)}\left[\hat{x}_j w_j - \frac{\partial A_1}{\partial c}^T w_j - \frac{\partial b_1}{\partial c}\right] \\ \nabla\varepsilon_{jy}(c) &= \frac{1}{q_j(c)}\left[\hat{y}_j w_j - \frac{\partial A_2}{\partial c}^T w_j - \frac{\partial b_2}{\partial c}\right]\end{aligned}$$

where

$$\begin{aligned}\frac{\partial A_1}{\partial c} &= \begin{bmatrix} \frac{\partial a_{11}}{\partial c_1} & \frac{\partial a_{11}}{\partial c_2} \\ \frac{\partial a_{12}}{\partial c_1} & \frac{\partial a_{12}}{\partial c_1} \end{bmatrix} \\ \frac{\partial A_2}{\partial c} &= \begin{bmatrix} \frac{\partial a_{21}}{\partial c_1} & \frac{\partial a_{21}}{\partial c_2} \\ \frac{\partial a_{22}}{\partial c_1} & \frac{\partial a_{22}}{\partial c_1} \end{bmatrix} \\ \frac{\partial b_1}{\partial c} &= \begin{bmatrix} \frac{\partial b_1}{\partial c_1} \\ \frac{\partial b_1}{\partial c_2} \end{bmatrix} \\ \frac{\partial b_2}{\partial c} &= \begin{bmatrix} \frac{\partial b_2}{\partial c_1} \\ \frac{\partial b_2}{\partial c_2} \end{bmatrix}\end{aligned}$$

Substituting into (C.14), we obtain

$$\begin{aligned}\nabla J(c) = \sum_{j=1}^{N} \frac{1}{q_j(c)}\varepsilon_j^T \hat{w}_j w_j - \Bigg(&\frac{\partial A_1}{\partial c}^T \sum_{j=1}^{N} \frac{1}{q_j(c)}\varepsilon_{jx} w_j + \frac{\partial A_2}{\partial c}^T \sum_{j=1}^{N} \frac{1}{q_j(c)}\varepsilon_{jy} w_j + \\ &\frac{\partial b_1}{\partial c}\sum_{j=1}^{N} \frac{1}{q_j(c)}\varepsilon_{jx} + \frac{\partial b_2}{\partial c}\sum_{j=1}^{N} \frac{1}{q_j(c)}\varepsilon_{jy}\Bigg)\end{aligned}$$

Each of the terms $\sum_{j=1}^{N} \frac{1}{q_j(c)} \varepsilon_{jx} w_j$, $\sum_{j=1}^{N} \frac{1}{q_j(c)} \varepsilon_{jy} w_j$, $\sum_{j=1}^{N} \frac{1}{q_j(c)} \varepsilon_{jx}$, $\sum_{j=1}^{N} \frac{1}{q_j(c)} \varepsilon_{jy}$ is identically zero for the two-dimensional problem; these are the normal equations $D_A Q$, $D_b Q$ that are 0 by construction on the manifold. Hence we obtain

$$\nabla J(c) = \sum_{j=1}^{N} \frac{1}{q_j(c)} \varepsilon_j^T \hat{w}_j w_j$$

as before.

The second derivative of $J(c)$ can be written

$$\nabla^2 J(c) = \sum_{j=1}^{N} \nabla \varepsilon_{jx}(c) \cdot \nabla \varepsilon_{jx}(c)^T + \nabla \varepsilon_{jy}(c) \cdot \nabla \varepsilon_{jy}(c)^T + \varepsilon_{jx}(c) \cdot \nabla^2 \varepsilon_{jx}(c) + \varepsilon_{jy}(c) \cdot \nabla^2 \varepsilon_{jy}(c) \tag{C.15}$$

The first two terms comprise the Gauss-Newton approximation to the Hessian and can be written explicitly as:

$$\begin{aligned} H_{GN} &= \sum_{j=1}^{N} \frac{1}{q_j^2(c)} \left[\hat{x}_j w_j - \frac{\partial A_1}{\partial c}^T w_j - \frac{\partial b_1}{\partial c} \right] \left[\hat{x}_j w_j^T - w_j^T \frac{\partial A_1}{\partial c} - \frac{\partial b_1}{\partial c}^T \right] \\ &\quad + \sum_{j=1}^{N} \frac{1}{q_j^2(c)} \left[\hat{y}_j w_j - \frac{\partial A_2}{\partial c}^T w_j - \frac{\partial b_2}{\partial c} \right] \left[\hat{y}_j w_j^T - w_j^T \frac{\partial A_2}{\partial c} - \frac{\partial b_2}{\partial c}^T \right] \end{aligned}$$

This sum of rank-one matrices can be rearranged as

$$H_{GN} = \sum_{j=1}^{N} \frac{1}{q_j^2(c)} (N_j - \hat{w}_j w_j^T)^T (N_j - \hat{w}_j w_j^T)$$

where

$$N_j = \left[\frac{\partial A}{\partial c_1} w_j + \frac{\partial b}{\partial c_1} \quad \frac{\partial A}{\partial c_2} w_j + \frac{\partial b}{\partial c_2} \right]$$

To evaluate the second part of the Hessian, we require $\nabla^2 \varepsilon_{jx}(c)$ and $\nabla^2 \varepsilon_{jy}(c)$, given by

$$\begin{aligned} \nabla^2 \varepsilon_{jx}(c) &= \sum_{j=1}^{N} \frac{1}{q_j^2(c)} \left[w_j \left(w_j^T \frac{\partial A_1}{\partial c} + \frac{\partial b_1}{\partial c}^T - \hat{x}_j w_j^T \right) \right. \\ &\quad \left. + \left(\frac{\partial A_1}{\partial c}^T w_j + \frac{\partial b_1}{\partial c} - \hat{x}_j w_j \right) w_j^T \right] + \frac{1}{q_j(c)} \left(\nabla^2 A_1 w_j + \nabla^2 b_1 \right) \\ \nabla^2 \varepsilon_{jy}(c) &= \sum_{j=1}^{N} \frac{1}{q_j^2(c)} \left[w_j \left(w_j^T \frac{\partial A_2}{\partial c} + \frac{\partial b_2}{\partial c}^T - \hat{y}_j w_j^T \right) \right. \\ &\quad \left. + \left(\frac{\partial A_2}{\partial c}^T w_j + \frac{\partial b_2}{\partial c} - \hat{y}_j w_j \right) w_j^T \right] + \frac{1}{q_j(c)} \left(\nabla^2 A_2 w_j + \nabla^2 b_2 \right) \end{aligned}$$

It may be unclear how to evaluate $\nabla^2 A_1 w_j$ and $\nabla^2 A_2 w_j$ but as we shall see it will not be necessary. The second part of the Hessian is given by

$$\begin{aligned}
\bar{H} &= \varepsilon_{jx}(c)\cdot\nabla^2\varepsilon_{jx}(c) + \varepsilon_{jy}(c)\cdot\nabla^2\varepsilon_{jy}(c) \\
&= \sum_{j=1}^{N}\frac{1}{q_j^2(c)}\varepsilon_{jx}\left[w_j\left(w_j^T\frac{\partial A_1}{\partial c}+\frac{\partial b_1}{\partial c}^T-\hat{x}_j w_j^T\right)+\left(\frac{\partial A_1}{\partial c}^T w_j+\frac{\partial b_1}{\partial c}-\hat{x}_j w_j\right)w_j^T\right] \\
&\quad+\sum_{j=1}^{N}\frac{1}{q_j^2(c)}\varepsilon_{jy}\left[w_j\left(w_j^T\frac{\partial A_2}{\partial c}+\frac{\partial b_2}{\partial c}^T-\hat{y}_j w_j^T\right)+\left(\frac{\partial A_2}{\partial c}^T w_j+\frac{\partial b_2}{\partial c}-\hat{y}_j w_j\right)w_j^T\right] \\
&\quad+\sum_{j=1}^{N}\frac{1}{q_j(c)}\left(\nabla^2 A_1\varepsilon_{jx}w_j+\nabla^2 b_1\varepsilon_{jx}+\nabla^2 A_2\varepsilon_{jy}w_j+\nabla^2 b_2\varepsilon_{jx}\right) \qquad \text{(C.16)}
\end{aligned}$$

The terms in (C.16) are identically zero, since the parameters are constrained to lie on the manifold.

The remaining terms can be rearranged as

$$\bar{H}=\sum_{j=1}^{N}\frac{1}{q_j^2(c)}\left[w_j\varepsilon_j^T N_j+N_j^T\varepsilon_j w_j^T-2\varepsilon_j^T\hat{w}_j w_j w_J^T\right]$$

Writing the Hessian as the sum of H_{GN} and $\bar{H}$ gives

$$H=\sum_{j=1}^{N}\frac{1}{q_j^2(c)}\left[N_j^T N_j-N_j^T(\hat{w}_j-\varepsilon_j)w_j^T-w_j(\hat{w}_j-\varepsilon_j)^T N_j+(\hat{w}_j-2\varepsilon_j)^T\hat{w}_j w_j w_j^T\right]$$

In this form the Hessian is clearly symmetric. However, it can be shown that the equations in the partials (C.8)-(C.11) imply that

$$\sum_{j=1}^{N}\frac{1}{q_j^2(c)}\left[N_j-(\hat{w}_j-\varepsilon_j)w_j^T\right]^T N_j=0$$

which means that

$$H=\sum_{j=1}^{N}\frac{1}{q_j^2(c)}\left[(\hat{w}_j-2\varepsilon_j)^T\hat{w}_j w_j w_j^T-N_j^T(\hat{w}_j-\varepsilon_j)w_j^T\right]$$

This is the expression we derived earlier as (C.5).

Appendix D

Audio Interpolation

As an aside, we prove a simple result about when there is enough information contained in the audio signals received at two microphones to synthesize the audio received at a third "virtual" microphone. We show that when the virtual microphone is located along the line connecting the two real microphones (the "baseline"), the audio can be synthesized with no knowledge besides the distance between the two microphones.

We call this result "audio interpolation" as a direct analogy to the term "view interpolation" from computer vision. Instead of using two real images of a scene to synthesize a new, physically correct image of the scene from a different perspective, we seek to use two real sounds of an environment to synthesize new, physically correct audio of the environment from a different position. In terms of prior work, Slaney et al. proposed an algorithm called "audio morphing" [1], a method for automatically transitioning from one sound into another. While the intermediate signals may sound plausible, they do not correspond to sound produced by real underlying sources and microphones. This is in direct analogy to Beier-Neely morphing, which produces intermediate images that correspond to no real physical objects. By combining audio interpolation with view interpolation, we can create virtual video that contains both images and sound.

We consider the one-source, two-microphone scenario in Figure D.1. Note the similarity to Figures 6.1 and 6.2. We assume that the source, located at P, omnidirectionally broadcasts a signal $x(t)$ which is received at the omnidirectional microphones M_0 and M_1 as $x_0(t)$ and $x_1(t)$. We assume that the microphones are calibrated in the sense that the distance Δ between them is known. This is slightly less general than the results from view interpolation, for which the distance between

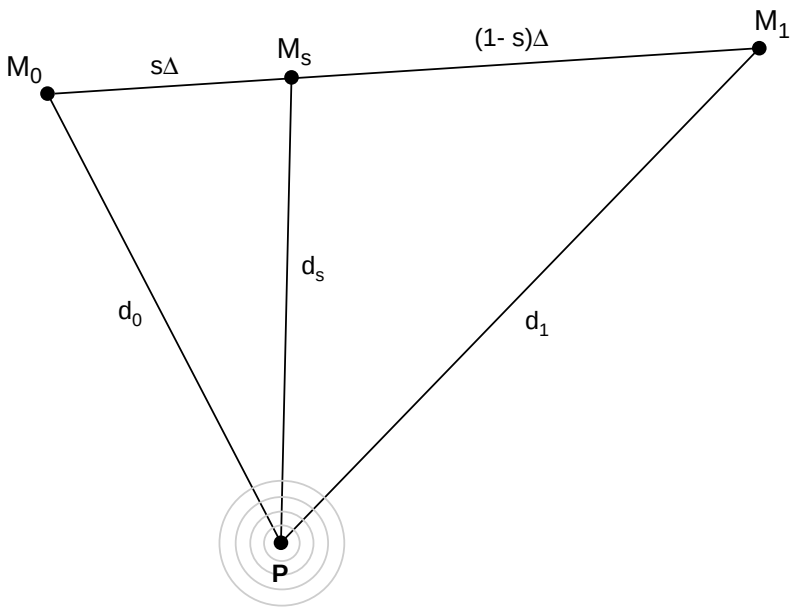


Figure D.1: Microphone configuration.

the cameras need not be known.

We define the distances of P to M_0 and M_1 to be d_0 and d_1 respectively. The location of the source is unknown; however, from the amplitude and delay differences of the signals $x_0(t)$ and $x_1(t)$ we can compute the difference $\delta_0 = d_1 - d_0$ and the ratio $a_0 = \frac{d_1}{d_0}$. The distances d_0 and d_1 can be easily recovered as

$$\begin{aligned} d_0 &= \frac{\delta_0}{a_0 - 1} \\ d_1 &= \frac{a_0 \delta_0}{a_0 - 1} \end{aligned}$$

We are interested in synthesizing the signal that would have been received at a microphone M_s placed a fraction s of the way along the line connecting M_0 to M_1 (the "baseline"). It is sufficient to calculate the distance d_s. This can be computed by a straightforward application of the law of cosines as

$$d_s = \sqrt{s(1-s)\Delta^2 + (1-s)d_0^2 + \alpha d_1^2} \tag{D.1}$$

This formula is correct for any value of s, not just $s \in [0, 1]$ as sketched in the figure. That is, the virtual microphone can range anywhere along the line through M_0 and M_1. This means that we

may take the original microphones to be as close together as we like. We can also see from (D.1) that knowledge of the microphone separation is necessary; the dependence on Δ is removed only in the trivial cases when the virtual microphone is at $s = 0$ or $s = 1$.

The signal from the virtual microphone can be reconstructed as:

$$x_s(t) = \frac{d_0}{d_s} x_0 \left(t - \frac{d_s - d_0}{\nu} \right)$$

where ν is the speed of sound. Unfortunately, the audio interpolation equation we derived is for a single source only. However, there has recently been substantial success on the problem of separating multiple sources from two stereo mixtures. In particular, Jourjine et al. [2] presented a novel method for blindly separating any number of sources using only two mixtures. The main assumption of the algorithm is that the sources are W-disjoint orthogonal, i.e. the supports of the windowed Fourier transforms of each pair $(x_i(t), x_j(t))$ of source signals are disjoint. This assumption was shown to be viable for mixtures of real sources, e.g. multiple voices speaking simultaneously. Mixing parameters of the sources are estimated by clustering ratios of the time-frequency representations of the mixtures. The estimates of the mixing parameters are used to partition the time-frequency representation of one mixture to recover the original sources. The technique is valid even when the number of sources is larger than the number of mixtures.

By coupling the audio interpolation equation with Jourjine's algorithm, we can synthesize realistic virtual audio even in the presence of multiple sources.

The baseline connecting M_0 and M_1 is unique in that it is the only location where the sound from a virtual microphone can be synthesized from only two microphones and multiple unknown sources. Using the signals $x_0(t)$ and $x_1(t)$, each source can be located only up to a point on a circle orthogonal to the baseline (Figure D.2). Three calibrated, non-colinear microphones can locate each source up to a pair of points; four calibrated, non-coplanar microphones can locate each source unambiguously. Thus, two calibrated microphones is really the only "interesting" case in which virtual audio can be synthesized from incomplete information.

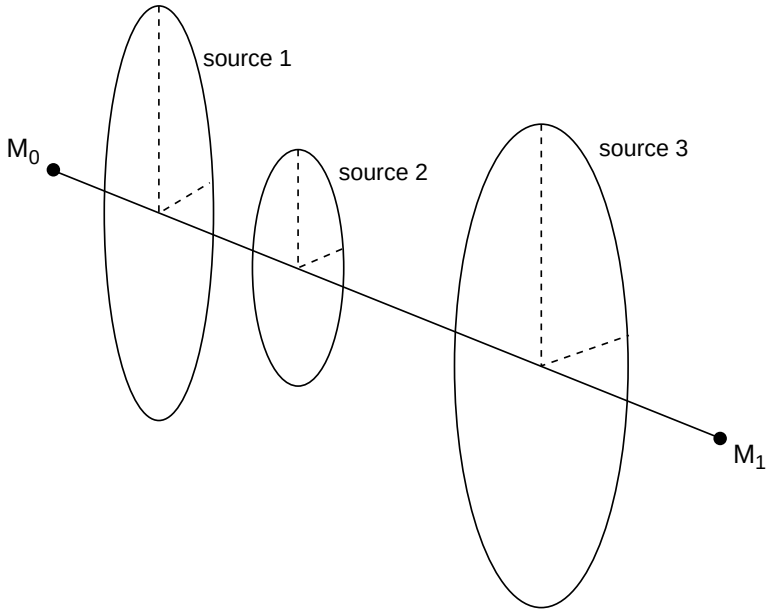


Figure D.2: Loci of sources that can be obtained with two microphones.

References

[1] M. Slaney, M. Covell and B. Lassiter. Automatic Audio Morphing. In *Proc. ICASSP 1996*, May 1996.

[2] A. Jourjine, S. Rickard, and O. Yilmaz. Blind Separation of Disjoint Orthogonal Signals: Demixing N Sources from 2 Mixtures. In *Proc. ICASSP 2000*, June 2000.