The Paraperspective and Projective Factorization Methods for Recovering Shape and Motion

Conrad J. Poelman

12 July 1995
CMU-CS-95-173

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis Committee:
Takeo Kanade, Chair
Steven Shafer
Paul Heckbert
Joseph Mundy, GE Corporate Research

Copyright © 1995 by Conrad J. Poelman

This research was sponsored by the Department of the Army, Army Research Office under grant number DAAH04-94-G-0006.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Army or the United States Government.

Keywords: Computer vision, motion, shape, time-varying imagery, iterative minimization

Carnegie Mellon

School of Computer Science

DOCTORAL THESIS
in the field of
Computer Science

The Paraperspective and Projective Factorization Methods for Recovering Shape and Motion

CONRAD J. POELMAN

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

ACCEPTED:

T Kanade — THESIS COMMITTEE CHAIR — July 12, 1995 — DATE

DEPARTMENT HEAD — 7/24/95 — DATE

APPROVED:

DEAN — 7/25/95 — DATE

Abstract

Sensing the shape and motion of an object from an image sequence is a central problem in computer vision, having applications in such diverse fields as autonomous navigation, cartography, and virtual reality systems. When an observer moves relative to an object, shape information is revealed through changes in the appearance of the object. In theory, the shape of an object and motion of the observer can be recovered by matching as few as eight points between two images. In practice, existing methods fail to work reliably in many noisy, real-world situations.

This thesis presents two novel techniques for recovering the shape and motion of a rigid object from a multi-image sequence. The paraperspective and projective factorization methods share a common approach with the orthographic factorization method developed by Tomasi and Kanade. The paraperspective factorization method is based on a closer approximation to image projection than orthography, enabling it to account for several important aspects of image projection that the original method could not. The projective factorization method is based on a more accurate model of image projection which not only accounts for standard perspective effects such as foreshortening, but can also model radial distortion, an important effect rarely considered in the shape from motion literature. The non-linear nature of the projection equations makes the projective factorization method more computationally intensive than the paraperspective method; however, its more accurate projection model allows its application to sequences in which the object is close to the camera or contains large depth disparities.

This thesis also addresses other issues vital to accurate shape recovery, such as robust tracking of features in sequences with large motions between images. Both methods are formulated to allow shape and motion recovery from sequences in which features become occluded or leave the field of view, and to account for the varying confidences of the feature tracking results.

The methods have been extensively tested and their behaviors systematically explored by controlled experimentation on over four thousand synthetically generated images. They have further been shown to successfully recover object shapes from real-world image sequences recorded using unsteady hand-held consumer-quality cameras, demonstrating their robustness to image noise and non-smooth camera motion.

Acknowledgments

Dedicated to my parents, Charlotte I. and R. Conrad Poelman, who always had enough confidence in me to support me unreservedly no matter where my interests led.

Thanks to my thesis committee: Takeo Kanade, who first instructed me the many arts of research; Steve Shafer, who sincerely tries to bring out the real scientist in all of us; Paul Heckbert, who read my thesis even more carefully than I did and offered countless helpful suggestions; and Joe Mundy, who always had a unique perspective to offer.

Thanks to my most recent officemates Jennifer Kay and Justin Boyan for putting up with my odd hours and strange habits during my thesis rush. Thanks to Denis Dancanet, Yalin Xiong, and Mark Maimone for helping me out of various last-minute emergencies. Thanks to Jim Rehg and Harry Shum for listening and helping me out with ideas at various stages of my research career.

Finally, thanks to Kay Reardon, without whose patience and love of teaching mathematics to high school students I could not have come this far.

Contents

List of Figures

List of Tables

1. Introduction

From a single image, little can be said for certain about the three-dimensional shape of the object or scene. Numerous methods have been developed for recovering shape using cues such as shading or defocus, or for recovering shape in limited domains where the images contain only right angles or certain types of smooth curves, yet these methods cannot reliably recover an object's shape from single images taken in unconstrained real-world situations. However, as an observer moves relative to an object, the image of the object changes in a way dependent on both the object's three-dimensional shape and the motion of the observer. In theory it is possible to recover both the shape of the object and the motion of the observer by matching observed points between as few as three images taken from different viewpoints. In practice, however, formulations based on a minimal number of observations have proven very sensitive to noise, and therefore unsuitable for real-world use.

Tomasi and Kanade [42][43] developed a robust and efficient method for accurately recovering the shape and motion of an object from a sequence of images, called the *factorization method*. Like most traditional techniques, the method requires that the image positions of point features (such as corners, bright spots, or surface markings) first be tracked throughout the stream. It achieves its accuracy and robustness by taking advantage of the large stream of redundant input by means of a well-understood numerical computation, the singular value decomposition (SVD), to process feature measurements from all images simultaneously. However, the original Tomasi-Kanade factorization method's applicability was limited due to its use of an orthographic projection model. Orthographic projection cannot account for several important effects of image projection, such as the scaling effect, whereby an object appears larger in the image as it approaches the camera, and therefore its application is restricted to certain ranges of motion. Furthermore it is unable to supply any estimation of translation along the camera's optical axis, which limits its usefulness for certain tasks.

This thesis describes two shape and motion recovery techniques which can be applied to a much wider range of situations and produce more accurate results than previous techniques. The first extends the factorization method to use a paraperspective projection model rather than an orthographic one. Paraperspective projection, first introduced by Ohta [28] and named by Aloimonos [2], models two effects of image projection that orthography does not model; the scaling effect, described above, and the position effect, whereby an object is viewed from different angles as it translates across the field of view. It can be modeled by bilinear equations similar to the orthographic equations, which allows for a factorization formulation of the shape and motion recovery process.

The second technique is the projective factorization method, which is based on a perspective projection model. By modeling the foreshortening effect of perspective projection in addition to the effects described above, this method can be applied to objects or scenes containing great depth disparities. While it shares a common approach and philosophy with the orthographic and paraperspective factorization methods, the non-linearity of the perspective projection equations necessitates the use of quite different techniques. The non-linear techniques applied to solve this problem tend to require more storage space and more computation than the bilinear methods. Therefore projective factorization is advantageous only when foreshortening effects make the use of the paraperspective method impossible.

Techniques are developed in this thesis to address several other practical issues vital to accurate shape recovery. One such technique enables robust tracking of features in sequences with large motions between images. Another extends the basic shape and motion recovery techniques to accommodate missing and uncertain data, which occur when points become occluded, leave the field of view, or for some other reason cannot be tracked throughout the entire sequence. This technique also improves the accuracy of shape and motion recovery by incorporating confidence measures for each feature measurement. The non-linear minimization technique developed for recovering shape and motion under a perspective projection model is also extended to account for unknown radial distortion in the image sequence, allowing the methods to be applied to a wider range of cameras and closer objects. The technique is also shown to enable the use of image intensities directly for shape and motion recovery rather than depending feature tracking measurements. While the technique is still "feature-based" in a sense, it allows the use of features which might not otherwise be tracked correctly, and therefore helps provide fuller shape recovery.

Successful techniques for accomplishing shape and motion recovery from image sequences could enable mobile robots to autonomously map and navigate through unknown environments, or aircraft to map terrain despite unknown and arbitrary, non-uniform motion, while simultaneously obtaining information about their own movement. Structure recovery techniques could be used in graphical simulation or virtual reality systems, replacing the currently painstaking task of defining 3D object models by hand with a system for automatically generating them from raw video footage. Such techniques also could prove powerful for image compression; once the three-dimensional scene geometry and image texture are transmitted, subsequent images could be reconstructed by sending only camera motion information.

Through extensive experiments on simulated, laboratory, and real video data, we demonstrate the performance of our methods, the relationships between them, and the particular advantages of each method. We show that the paraperspective and projective factorization methods can be used for shape and motion recovery in many practical scenarios.

1.1. Related Work

The problem of computing shape and motion from image streams finds its earliest roots in stereo vision research. In the stereo problem, two or more cameras image the same scene. The position and orientation of the second camera relative to the first is fixed, determined once through advance calibration. The depths of various points in the scene are generally computed by matching corresponding points in the two images, and triangulating to compute the depth.

Early structure from motion work by Longuet-Higgins and Tsai and Huang showed that it was possible to recover the depths of the points and the relative orientations of two cameras from two perspective views provided at least 8 point correspondences were known [23][45]. The solution put forth in [45], first solving for a set of essential parameters which can further be analyzed to reveal the camera motion, requires only solving a system of linear equations and computing the SVD of a small matrix. Faugeras, Lustman, and Toscani, as well as Spetsakis and Aloimonis developed similar approaches for recovering the positions of three-dimensional lines viewed in three images, the minimum necessary to provide a unique solution [16][35]. Demey, Zisserman, and Beardsley, Faugeras, and Shashua have developed numerous other techniques for recovering shape and motion from a small number of images and feature points [11][14][33], based on more recent insights into projective and affine geometry. However, without the advantage of data redundancy, these methods are highly sensitive to even low levels of image noise, and therefore frequently fail on sequences taken outside of a controlled laboratory environment.

Improved shape recovery can be achieved by restricting the range of allowable motions, by requiring that the motion be known *a priori,* as was done by Matthies, Kanade, and Szeliski [24], assuming the motion to be smooth, or by allowing only translational or only rotational motion, as was done by Horn and Weldon [22]. In the case of general motion, however, many reports have admitted that traditional methods have failed to produce reliable results in many situations [8][13]. Since methods using the minimal number of features or the minimal number of images tended to be highly sensitive to image noise, researchers began to look to longer sequences containing more points for methods that could provide improved robustness.

Soatto, Perona, and Frezza applied Kalman filtering techniques to combine a sequence of two-frame solutions to produce a single, more robust estimate [36], as did Azerbayejani and Pentland [3], Harris [19], Thomas, Hansen, and Oliensis [40], and many others. Xiong and Shafer have developed a efficient techniques for recovering a dense depth map as well as the motion from an image sequence using Kalman filter integration of two-frame depth estimates derived from optical flow values, in some ways an extension of the approach of Matthies, Kanade, and Szeliski [24] to the case of unknown motion [47].

The factorization method developed by Tomasi and Kanade [42][43] is a "batch" method which processes data from all frames simultaneously. It achieved its robustness by processing large numbers of orthographically projected images with many points tracked throughout the sequence. Basing the method on orthographic projection allowed them to avoid complicated non-linear solution methods. In a sense, their work parallels the "essential parameter" approach of Tsai and Huang [45]. In the decomposition stage, they use the SVD to solve for a set of "essential motion parameters" and an affine shape estimate. In the normalization stage, they solve for the affine transformation which transforms the affine shape into a Euclidean shape, by constraining the form of the essential motion parameter matrix. Finally in the motion recovery stage, they compute the camera motion from the essential parameters. In order to make use of many points and frames under a of perspective projection model, others have used non-linear optimization techniques. Taylor, Kriegman, and Anandan [39] developed a solution method for the case of 1D images which involved separately refining the shape and motion parameters. Szeliski and Kang [38] used Levenberg-Marquardt iteration to find a least-squares solution to the large system of non-linear perspective projection equations, using sparse matrix techniques to efficiently represent and invert the Hessian matrix.

Although methods based on projective geometry and projective invariants have recently become popular, many of these methods still address only the two-frame problem. Mohr, Veillon, and Quan put forth an iterative non-linear method inspired by projective geometry which simultaneously uses the information from all images and all frames [25]. They solve for the projective shape and motion, and in [6] Boufama, Mohr, and Veillon show how to convert this solution to Euclidean shape when additional scene knowledge is available. Ponce, Marimont, and Cass present an analogous method which requires the selection of five points to use as the basis for the projective coordinates and then reduces the problem to a minimization involving only polynomial functions [30]. Szeliski and Kang [38] also extend their framework to address projective shape and motion recovery, though they don't address the problem of Euclidean reconstruction from projective shape and motion.

Faugeras, Luong, and Maybank developed a method for recovering intrinsic and extrinsic camera calibration parameters from three images [15]. The epipolar geometry is estimated between pairs of images, and constraints on the form of the computed fundamental matrices are used to recover the camera parameters. They use the continuation method to solve a set of non-linear equations which is known to have many local minima. Since the epipolar geometry is only computed from pairs of images, scene rigidity throughout the three images is not strictly enforced. The method was shown to be extremely sensitive to noise, and was not demonstrated on real images. Hartley's method [18] involves first computing the projective shape and motion, and then converting them to a Euclidean solution using a series of steps including linear programming, Choleski decomposition, and Levenberg-Marquardt iteration.

1.2. Contributions

- The scaled orthographic factorization method for shape and motion recovery, which accounts for the scaling effect of the image projection.

- The paraperspective factorization method, which accounts for the scaling and position effects of image projection.

- The separate refinement method, which iteratively improves a shape and motion estimate using a perspective projection model to accurately account for the foreshortening effect.

- A clarification of the relationships between the paraperspective projection model, the general affine camera model, and the "fixed-intrinsic" affine camera model.
- A derivation of paraperspective projection as an mathematical first-order approximation to perspective projection.
- A robust hierarchical feature tracker which can track large feature motions.
- A method for accommodating occluded, missing, and uncertain feature tracking data within the factorization methods.
- The projective factorization method, which accounts for foreshortening and radial distortion effects, and has better convergence properties than the separate refinement method.
- Convincing experimental evidence that solving for projective shape and motion has superior convergence properties to directly solving for shape and motion using a Euclidean formulation.
- Introduction of the notion of fill fraction as an important measure which effects the accuracy and computational efficiency of shape and motion recovery in the presence of occlusion or missing data.
- A direct method for recovering shape and motion from an image sequence without first tracking feature points.
- Detailed examination of the behaviors of the paraperspective and projective factorization methods as functions of distance, noise level, fill fraction.
- Verification of the practicality of the factorization methods by demonstrating successful shape recovery from noisy sequences taken with consumer-quality hand-held video cameras.

1.3. Geometry and Notation

In a shape-from-motion problem, we are given a sequence of F images taken from a camera that is moving relative to an object. Imagine that we locate P prominent feature points in the first image. Each feature point corresponds to a single world point, located at position $\mathbf{s}_p$ in some fixed world coordinate system. This point will appear at varying positions in each of the following images, depending on the position and orientation of the camera in that image. We write the observed image position of point p in frame f as the two-vector $\mathbf{u}_{fp}$ containing its image x- and y-coordinates, which we will sometimes write as (u_{fp}, v_{fp}). These image positions are measured by tracking the feature from frame to frame using the tracking techniques describe in chapter 3.

The camera position and orientation in each frame is described by a rotation matrix R_f and a translation vector $\mathbf{t}_f$ representing the transformation from world coordinates to camera coordinates in each frame. We can physically interpret the rows of R_f as giving the orientation of the camera axes in each frame - the first row, $\mathbf{i}_f$, gives the orientation of the camera's x-axis, the second row, $\mathbf{j}_f$, gives the orientation of the camera's y-axis, and the third row, $\mathbf{k}_f$, gives the orientation of the camera's optical axis, which points along the camera's line of sight. The vector $\mathbf{t}_f$ indicates the position of the camera in each frame by pointing from the world origin to the camera's focal point. This formulation is illustrated in Figure 1.

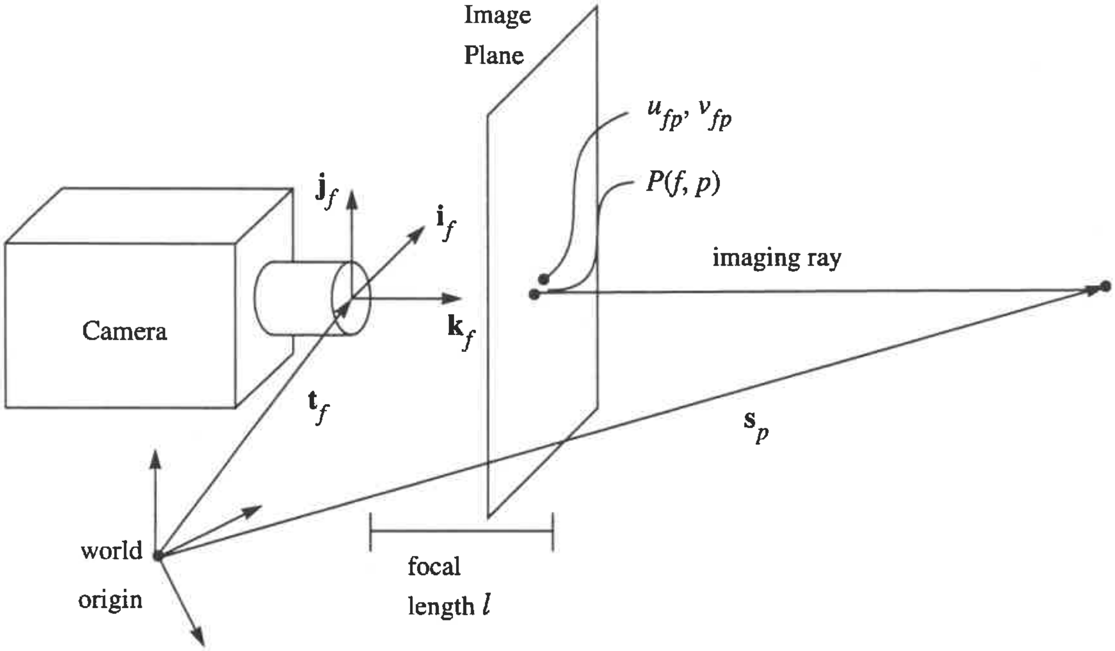


Figure 1. Coordinate system and Notation

The process of projecting a three-dimensional point onto the image plane in a given frame is referred to as projection. This process models the physical process by which light from a point in the world is focused on the camera's image plane, and mathematical projection models of various degrees of sophistication can be used to compute the expected or predicted image positions $P(f, p)$ as a function of $\mathbf{s}_p$, R_f, and $\mathbf{t}_f$. In fact, this process depends not only on the position of a point and the position and orientation of the camera, but also on the complex lens optics and image digitization characteristics. In this paper, at various points we use the orthographic projection model $P_o(f, p)$, the scaled orthographic projection model $P_{so}(f, p)$, the paraperspective projection model $P_{pp}(f, p)$, the perspective projection model $P_p(f, p)$, and the radial projection model $P_r(f, p)$. These models have varying degrees of mathematical sophistication and complexity, and account for the actual physics of image formation to increasingly accurate degrees. These projection models will be defined in Chapter 2, with the exception of the radial projection model which is introduced in Section 4.3, and are summarize for the reader's convenience in the section following this one.

Certain camera and digitizer parameters effect the way that images are transmitted from the world onto the image plane and into their final digital form. These parameters are the focal length l, the aspect ratio a, the image center (o_x, o_y), and the radial distortion κ. Some projection models use only a subset of these parameters. In some cases we simplify our equations by assuming "standard camera parameters", which means $l = a = 1$ and $o_x = o_y = \kappa = 0$. When the camera parameters are known, we can use them to transform the image feature point measurements into images taken with this "standard camera."

The shape from motion problem can be essentially stated as, given a sequence of images, recover the camera position in every frame f and the three-dimensional position of every point p. These values are computed so as to align the predicted position of each point in each frame $P(f, p)$ as closely as possible with the observed position $\mathbf{u}_{fp}$. We sometimes write our final estimated shape and motion, which of course due to noise may differ from the actual shape and motion, as $\hat{\mathbf{s}}_p$ for each object point, and $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, $\hat{\mathbf{k}}_f$ and $\hat{\mathbf{t}}_f$ for each frame in the sequence.

1.4. Imaging Effects and Projection Models

In the course of this thesis a number of different projection models are introduced, each modeling various effects of real image projection. The projection models will be described geometrically and derived mathematically as they are introduced. However, to help acquaint the reader with the terms and projection models, the following two tables summarize the effects of image projection and the various projection models used in this paper.

Table 1: Description of Imaging Effects

Effect	Description
scaling effect	An object appears larger when it is close to the camera than when it is far from the camera.
position effect	An object positioned away from the center of the image appears to be slightly rotated so that the rear of the object is closer to the image center.
foreshortening effect	The back of an object appears smaller than the front of the object.
radial distortion effect	Points far from the image center appear closer to the center that they would absent this effect. Straight lines not passing through the image center appear as curves, with the ends farthest from the image center bent towards the image center.

Table 2: Projection Models' Equations and Effects

projection model	projection equations	effects modeled
orthographic	$P_o(f,p) = \begin{bmatrix} l\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) + o_x \\ la\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) + o_y \end{bmatrix}$	
scaled orthographic	$P_{so}(f,p) = \begin{bmatrix} \dfrac{l\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} + o_x \\ \dfrac{la\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} + o_y \end{bmatrix}$	scaling
paraperspective	$P_{pp}(f,p) = \begin{bmatrix} \dfrac{l}{z_f}\left\{\left[\mathbf{i}_f + \dfrac{\mathbf{i}_f \cdot \mathbf{t}_f}{z_f}\mathbf{k}_f\right] \cdot \mathbf{s}_p - (\mathbf{t}_f \cdot \mathbf{i}_f)\right\} + o_x \\ \dfrac{la}{z_f}\left\{\left[\mathbf{j}_f + \dfrac{\mathbf{j}_f \cdot \mathbf{t}_f}{z_f}\mathbf{k}_f\right] \cdot \mathbf{s}_p - (\mathbf{t}_f \cdot \mathbf{j}_f)\right\} + o_y \end{bmatrix}$	scaling, position
perspective	$P_p(f,p) = \begin{bmatrix} l\dfrac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{\mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)} + o_x \\ la\dfrac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{\mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)} + o_y \end{bmatrix}$	scaling, position, foreshortening
radial	$P_r(f,p) = \begin{bmatrix} \dfrac{lP_{x_p}(f,p)}{\left(1 + \kappa_1 P_{x_p}(f,p)^2 + \kappa_1 P_{y_p}(f,p)^2\right)} + o_x \\ \dfrac{laP_{p_x}(f,p)}{\left(1 + \kappa_1 P_{x_p}(f,p)^2 + \kappa_1 P_{y_p}(f,p)^2\right)} + o_y \end{bmatrix}$ [a]	scaling, position, foreshortening, radial

[a] Radial projection first involves perspective projection using the "standard camera parameters" defined in section 1.3.

1.5. Thesis Overview

Chapter 2 reviews the original factorization method, which was based on orthography, and then extends the method to scaled orthography, and then to paraperspective. The chapter includes a comparison of paraperspective to the standard perspective projection model, and to the affine camera model. We show that the paraperspective results can be refined to account for perspective effects using a non-linear refinement step. We conclude with the results of numerous synthetic and real experiments demonstrating the paraperspective factorization method.

Chapter 3 addresses two practical issues of importance in applying the method to real image sequences: reliable feature tracking despite non-smooth motion and accommodating missing observations. The two techniques are related in the way they make use of image confidence information to weight different observations. Feature tracking is made robust by using a hierarchical method which weights motion estimates from different levels of the image pyramid. Missing observations, such as occur when features become occluded or enter or leave the field of view, are handled by extending the decomposition stage of the factorization method to allow an arbitrary confidence value to be assigned to each measurement. We demonstrate the performance of the feature tracker on two sequences, and explore the performance of the confidence-weighted factorization method as a function of the number of missing observations. We finally demonstrate the combined system on a real experiment using aerial image data.

Chapter 4 introduces the projective factorization method. While the projective decomposition step is not original to this work, we clearly explain how our implementation takes advantage of the sparsity of certain matrices to enable the method to be applied to problems with hundreds of points and images. The normalization step makes use of metric constraints very similar to the original factorization method to convert the projective shape and motion into a Euclidean solution. We experimentally compare this method to a method which directly computes the Euclidean shape and motion without first computing projective shape and motion. Finally we show that the general non-linear framework developed in the chapter can be extended to directly minimize image intensity errors to reduce reliance on tracked feature points.

Chapter 5 summarizes the contributions of the thesis and outlines directions for future research.

2. Paraperspective Factorization

The factorization method, developed by Tomasi and Kanade, assumes that world points are projected onto the image plane using orthographic projection. Orthographic projection is considered an appropriate model for use in many common imaging situations in which the object is very distant from the camera, precisely the situations in which traditional triangulation-based methods fail. Furthermore, it can be modeled by bilinear equations, which enable an efficient and robust solution.

The method was shown to perform extremely well in situations in which the orthographic assumption was valid. However, orthography cannot model several perspective effects such as the scaling, position, and foreshortening effects, which are illustrated in Figure 2. The scaling effect is apparent even in sequences in which the object is very distant from the camera, but in which the object translates significantly toward or away from the camera. The orthographic factorization method was unable to explain these scale changes, and therefore either failed to produce a result or produced a deformed shape, in which deformities in the object had been crafted to attempt to account for the scale change.

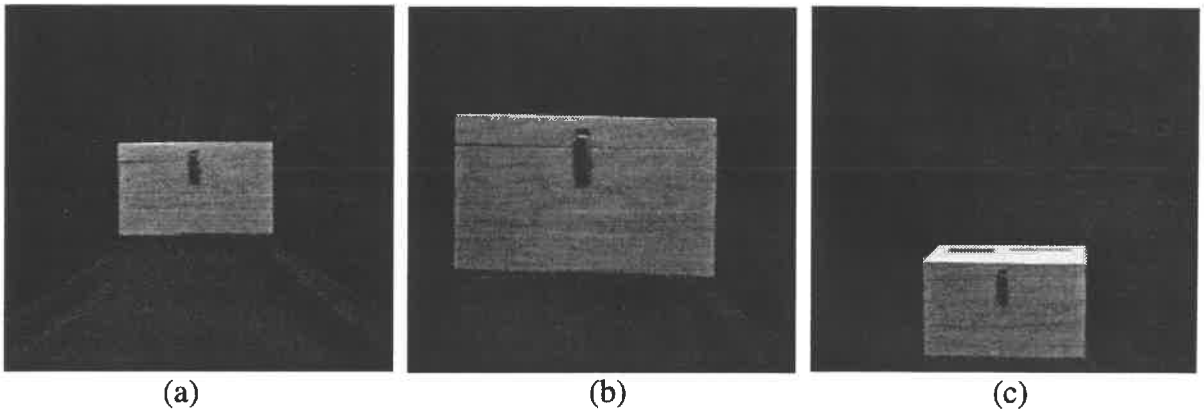


Figure 2. Effects of perspective projection

(a) the initial image (b) the image after translating the camera towards the object, demonstrating the scaling effect (c) the image after translating the camera vertically. Notice that in (c) the top of the object in has become visible; due to the position effect, the box is being effectively viewing from an angle. (c) also demonstrates the foreshortening effect, which causes the rear of the box to appear smaller than the front. This is the only effect of perspective projection that is not modeled by paraperspective projection.

In this chapter, we extend the factorization method to model some of the effects of perspective projection which are not modeled by orthography. The scaled orthographic factorization method accounts for the scaling effect and allows shape recovery from sequences containing depth translation. The paraperspective factorization method accounts for the scaling and position effects, allowing shape recovery from sequences in which the object is closer to the camera and not always centered in the image. Finally we present the separate refinement method, an iterative method which refines the results of paraperspective projection using a perspective model to account for the foreshortening effect.

2.1. The Orthographic Factorization Method

This section presents a summary of the orthographic factorization method developed by Tomasi and Kanade. A more detailed description of the method can be found in [43].

2.1.1. Orthographic Projection

The orthographic projection model assumes that rays are projected from an object point along the direction parallel to the camera's optical axis, so that they strike the image plane orthogonally, as illustrated in Figure 3.

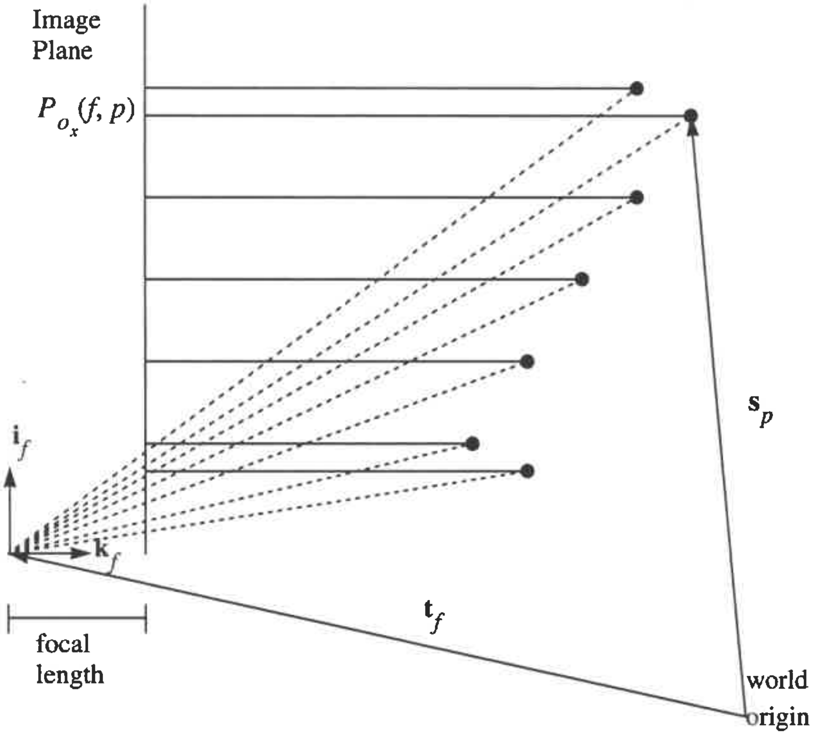


Figure 3. Orthographic projection in two dimensions
Dotted lines indicate perspective projection

A point p whose location is $\mathbf{s}_p$ will be observed in frame f at image coordinates $P_o(f, p)$, where

$$P_o(f,p) = \begin{bmatrix} P_{x_o}(f,p) \\ P_{y_o}(f,p) \end{bmatrix} = \begin{bmatrix} l\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) + o_x \\ la\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) + o_y \end{bmatrix} \tag{1}$$

Simplifying using standard camera parameters, these equations can be rewritten as

$$P_{x_o}(f,p) = \mathbf{m}_f \cdot \mathbf{s}_p + x_f \qquad P_{y_o}(f,p) = \mathbf{n}_f \cdot \mathbf{s}_p + y_f \tag{2}$$

where

$$x_f = -(\mathbf{t}_f \cdot \mathbf{i}_f) \qquad y_f = -(\mathbf{t}_f \cdot \mathbf{j}_f) \tag{3}$$

$$\mathbf{m}_f = \mathbf{i}_f \qquad \mathbf{n}_f = \mathbf{j}_f \tag{4}$$

2.1.2. Decomposition

All of the feature point coordinates (u_{fp}, v_{fp}) are entered in a $2F \times P$ *measurement matrix* W.

$$W = \begin{bmatrix} u_{11} & \cdots & u_{1P} \\ \cdots & \cdots & \cdots \\ u_{F1} & \cdots & u_{FP} \\ v_{11} & \cdots & v_{1P} \\ \cdots & \cdots & \cdots \\ v_{F1} & \cdots & v_{FP} \end{bmatrix} \tag{5}$$

Each column of the measurement matrix contains the observations for a single point, while each row contains the observed u-coordinates or v-coordinates for a single frame. Setting the observed positions equal to the predicted positions, equation (2) for all points and frames can now be combined into the single matrix equation

$$W = MS + T\begin{bmatrix}1 & \dots & 1\end{bmatrix} \tag{6}$$

where M is the $2F \times 3$ motion matrix whose rows are the $\mathbf{m}_f$ and $\mathbf{n}_f$ vectors, S is the $3 \times P$ shape matrix whose columns are the $\mathbf{s}_p$ vectors, and T is the $2F \times 1$ translation vector whose elements are the x_f and y_f.

Up to this point, Tomasi and Kanade placed no restrictions on the location of the world origin, except that it be stationary with respect to the object. Without loss of generality, they position the world origin at the center of mass of the object, denoted by $\mathbf{c}$, so that

$$\mathbf{c} = \frac{1}{P}\sum_{p=1}^{P} \mathbf{s}_p = 0 \tag{7}$$

Because the sum of any row of S is zero, the sum of any row i of W is PT_i. This enables them to compute the i^{th} element of the translation vector T directly from W, simply by averaging the i^{th} row of the measurement matrix. The translation is then subtracted from W, leaving a "registered" measurement matrix $W^* = W - T\begin{bmatrix}1 & \dots & 1\end{bmatrix}$.

W^* is bilinear in the motion variables M and the shape variables S because it equals a sum of products of them. The term "bilinear" refers to a problem whose variables can be partitioned into two sets such that the problem is linear with respect to one set of variables when the other set is held constant. Tomasi and Kanade pointed out that, since W^* is the product of a $2F \times 3$ motion matrix M and a $3 \times P$ shape matrix S, its rank should be at most 3. When noise is present in the input, W^* is not exactly of rank 3. Tomasi and Kanade used the Singular Value Decomposition (SVD) to compute $W^* = U\Sigma V^T$, where Σ is a diagonal matrix containing the singular values of the matrix [42]. In general, only three of these values are large, and the rest are extremely small, and are due primarily to noise in the measurement data. Therefore they use only the three largest singular values and their associated singular vectors to factor W^* into the product

$$W^* = \hat{M}\hat{S} \tag{8}$$

Using the SVD in this manner to perform the decomposition ensures that the product $\hat{M}\hat{S}$ is the best possible rank 3 approximation to the full measurement matrix W^*.

2.1.3. Normalization

The decomposition of equation (8) is only determined up to a linear transformation. In general, if the world origin had not been fixed at the mass center of the object, $\hat{M}$ and $\hat{S}$ would be determined up to an affine transformation, so for compatibility with standard terminology $\hat{M}$ and $\hat{S}$ may be referred to as the *affine motion* and *affine shape*. Any non-singular 3×3 matrix A and its inverse could be inserted between $\hat{M}$ and $\hat{S}$, and their product would still equal W^*. Thus the actual motion and shape are given by

$$M = \hat{M}A \qquad S = A^{-1}\hat{S} \tag{9}$$

with the appropriate 3×3 invertible matrix A selected. The correct A can be determined using the fact that the rows of the motion matrix M (which are the $\mathbf{m}_f$ and $\mathbf{n}_f$ vectors) represent the camera axes, and therefore they must be of a certain form. Since $\mathbf{i}_f$ and $\mathbf{j}_f$ are unit vectors, we see from equation (4) that

$$|\mathbf{m}_f|^2 = 1 \qquad |\mathbf{n}_f|^2 = 1 \tag{10}$$

and because they are orthogonal,

$$\mathbf{m}_f \cdot \mathbf{n}_f = 0 \tag{11}$$

Equations (10) and (11) give us $3F$ equations which we call the *metric constraints*. Using these constraints, we solve for the 3×3 matrix A which, when multiplied by $\hat{M}$, produces the motion matrix M that best satisfies these constraints. Once the matrix A has been found, the shape and motion are computed from equation (9).

2.2. The Scaled Orthographic Factorization Method

Scaled orthographic projection, also known as "weak perspective" [26], is a closer approximation to perspective projection than orthographic projection, yet not as accurate as paraperspective projection. It models the scaling effect of perspective projection, but not the position effect. The scaled orthographic factorization method can be used when the object remains centered in the image, or when the distance to the object is large enough relative to the size of the object that the position effect is negligible.

2.2.1. Scaled Orthographic Projection

Under scaled orthographic projection, object points are orthographically projected onto a hypothetical image plane parallel to the actual image plane but passing through the object's center of mass $\mathbf{c}$. This image is then projected onto the image plane using perspective projection, as shown in Figure 4.

Since the perspectively projected points all lie on a plane parallel to the image plane, they all lie at the same depth

$$z_f = (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{k}_f \qquad (12)$$

The scaled orthographic projection equations are very similar to the orthographic projection equations, except that the image plane coordinates are scaled by the ratio of the focal length to the depth z_f.

$$P_{so}(f, p) = \begin{bmatrix} P_{x_{so}}(f, p) \\ P_{y_{so}}(f, p) \end{bmatrix} = \begin{bmatrix} \dfrac{l\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} + o_x \\ \dfrac{la\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} + o_y \end{bmatrix} \qquad (13)$$

To simplify the equations we assume the standard camera parameters $l = 1$, $a = 1$, and $(o_x, o_y) = (0, 0)$. The world origin is arbitrary, so we fix it at the object's center of mass, so that $\mathbf{c} = 0$, and rewrite the above equations as

$$P_{x_{so}}(f, p) = \mathbf{m}_f \cdot \mathbf{s}_p + x_f \qquad P_{y_{so}}(f, p) = \mathbf{n}_f \cdot \mathbf{s}_p + y_f \qquad (14)$$

where

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \qquad (15)$$

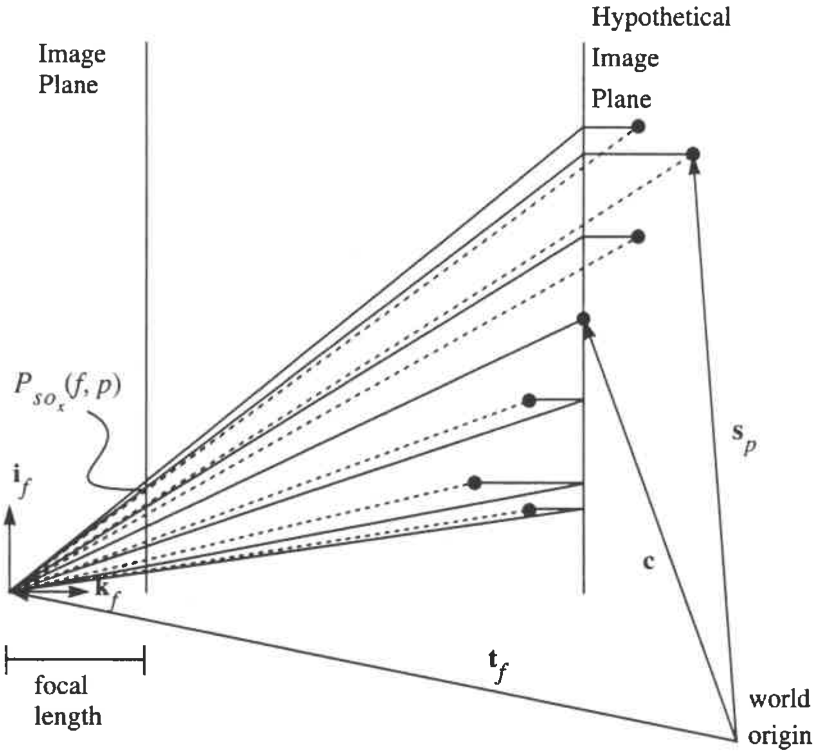


Figure 4 Scaled Orthographic Projection in two dimensions
Dotted lines indicate true perspective projection

$$x_f = -\frac{\mathbf{t}_f \cdot \mathbf{i}_f}{z_f} \qquad y_f = -\frac{\mathbf{t}_f \cdot \mathbf{j}_f}{z_f} \tag{16}$$

$$\mathbf{m}_f = \frac{\mathbf{i}_f}{z_f} \qquad \mathbf{n}_f = \frac{\mathbf{j}_f}{z_f} \tag{17}$$

2.2.2. Decomposition

Because equation (14) is identical to equation (2), the measurement matrix W can still be written as $W = MS + T$ just as in orthographic and paraperspective cases. We still compute x_f and y_f immediately from the image data by subtracting the center of gravity, and use singular value decomposition to factor the registered measurement matrix W^* into the product of $\hat{M}$ and $\hat{S}$.

2.2.3. Normalization

Again, the decomposition is not unique and we must determine the 3×3 matrix A which produces the actual motion matrix $M = \hat{M}A$ and the shape matrix $S = A^{-1}\hat{S}$. From equation

(17),

$$|\mathbf{m}_f|^2 = \frac{1}{z_f^2} \qquad |\mathbf{n}_f|^2 = \frac{1}{z_f^2} \tag{18}$$

The depth z_f is unknown, so we cannot impose individual constraints on $\mathbf{m}_f$ and $\mathbf{n}_f$ as we did in the orthographic case. Instead, we combine the two equations to impose the constraint

$$|\mathbf{m}_f|^2 = |\mathbf{n}_f|^2 . \tag{19}$$

Because $\mathbf{m}_f$ and $\mathbf{n}_f$ are just scalar multiples of $\mathbf{i}_f$ and $\mathbf{j}_f$, we can still use the constraint that

$$\mathbf{m}_f \cdot \mathbf{n}_f = 0 . \tag{20}$$

Equations (19) and (20) are homogeneous constraints, which could be trivially satisfied by the solution $M = 0$, so to avoid this solution we add the constraint that

$$|\mathbf{m}_1| = 1 . \tag{21}$$

Equations (19), (20), and (21) are the scaled orthographic version of the *metric constraints*. We can compute the 3×3 matrix A which best satisfies them very easily, because the constraints are linear in the 6 unique elements of the symmetric 3×3 matrix $Q = A^T A$.

2.2.4. Shape and Motion Recovery

Once the matrix A has been found, the shape is computed as $S = A^{-1}\hat{S}$. We compute the motion parameters as

$$\hat{\mathbf{i}}_f = \frac{\mathbf{m}_f}{|\mathbf{m}_f|} \qquad \hat{\mathbf{j}}_f = \frac{\mathbf{n}_f}{|\mathbf{n}_f|} . \tag{22}$$

Unlike the orthographic case, we can now compute z_f, the component of translation along the camera's optical axis, from equation (18).

2.3. The Paraperspective Factorization Method

Scaled orthography accurately models the scaling effect, and can therefore represents a significant improvement over the original factorization method. The method can successfully recover shape and motion from image sequences in which the object is relatively distant from the camera translates towards or away from the camera. In such cases, the foreshortening and position effects are small and induce only minor errors in the recovered shape and motion. Experiments indicate, however, that we must model the position effect in order to successfully recover shape and motion in sequences in which an object is closer to the camera and does not remain centered in the image. *Paraperspective projection*, introduced by Ohta [28] in order to solve a shape from texture problem, more closely approximates perspective projection by modeling both the scaling effect and the position effect, while retaining the bilinear algebraic properties of orthography. Based on this model, in this section we present a paraperspective factorization method similar to the original Tomasi-Kanade factorization method.

2.3.1. Paraperspective Projection

The paraperspective projection of an object onto an image, illustrated in Figure 5, involves two steps.

1. An object point is projected along a direction parallel to the line connecting the focal point of the camera to the object's center of mass, onto a hypothetical image plane parallel to the real image plane and passing through the object's center of mass.
2. The point is then projected onto the real image plane using perspective projection. Because the hypothetical plane is parallel to the real image plane, this is equivalent to simply scaling the point coordinates by the ratio of the camera focal length and the distance between the two planes.

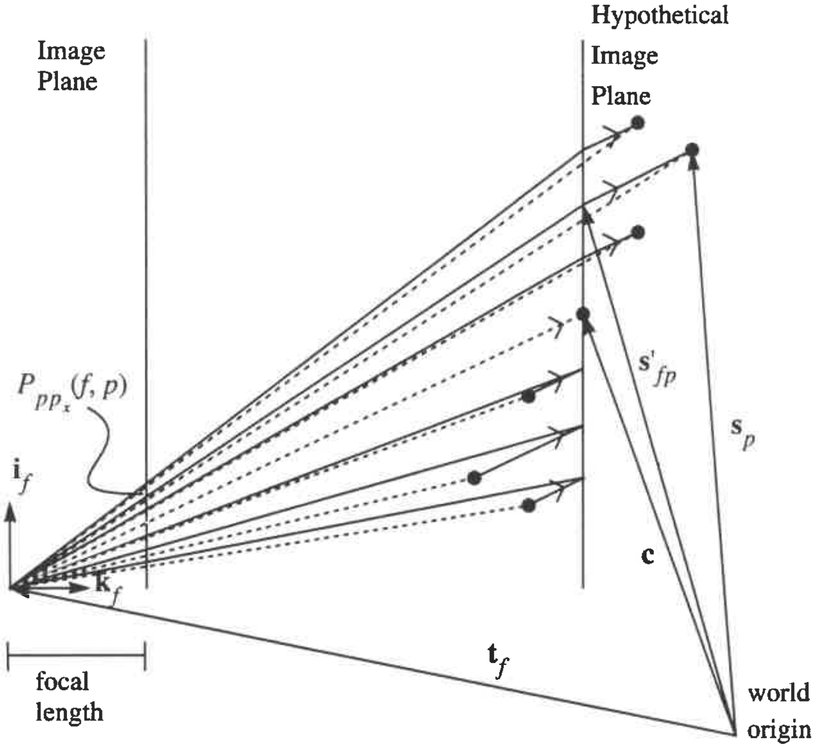


Figure 5. Paraperspective projection in two dimensions
Dotted lines indicate perspective projection
≻≻ indicates parallel lines

In general, the projection of a point $\mathbf{p}$ along direction $\mathbf{r}$, onto the plane defined by normal vector $\mathbf{n}$ and distance from the origin d, is given by the equation

$$\mathbf{p}' = \mathbf{p} - \frac{\mathbf{p} \cdot \mathbf{n} - d}{\mathbf{r} \cdot \mathbf{n}} \mathbf{r} . \tag{23}$$

In frame f, each object point $\mathbf{s}_p$ is projected along the direction $\mathbf{c} - \mathbf{t}_f$ (the direction from the camera's focal point to the object's center of mass) onto the plane defined by normal $\mathbf{k}_f$ and distance from the origin $\mathbf{c} \cdot \mathbf{k}_f$. The result $\mathbf{s}'_{fp}$ of this projection is

$$\mathbf{s}'_{fp} = \mathbf{s}_p - \frac{(\mathbf{s}_p \cdot \mathbf{k}_f) - (\mathbf{c} \cdot \mathbf{k}_f)}{(\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{k}_f} (\mathbf{c} - \mathbf{t}_f) \tag{24}$$

The perspective projection of this point onto the image plane is given by subtracting $\mathbf{t}_f$ from $\mathbf{s}'_{fp}$ to give the position of the point in the camera's coordinate system, and then scaling the result by the ratio of the camera's focal length l to the depth to the object's center of mass z_f. This yields the coordinates of the projection in the image plane,

$$P_{pp}(f, p) = \begin{bmatrix} P_{x_{pp}}(f, p) \\ P_{y_{pp}}(f, p) \end{bmatrix} = \begin{bmatrix} \dfrac{l\mathbf{i}_f \cdot (\mathbf{s}'_{fp} - \mathbf{t}_f)}{z_f} + o_x \\ \dfrac{la\mathbf{j}_f \cdot (\mathbf{s}'_{fp} - \mathbf{t}_f)}{z_f} + o_y \end{bmatrix}, \qquad \text{where } z_f = (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{k}_f \tag{25}$$

Substituting (24) into (25) gives the general paraperspective equations.

$$\begin{aligned} P_{x_{pp}}(f, p) &= \frac{l}{z_f} \left\{ \left[\mathbf{i}_f - \frac{\mathbf{i}_f \cdot (\mathbf{c} - \mathbf{t}_f)}{z_f} \mathbf{k}_f \right] \cdot (\mathbf{s}_p - \mathbf{c}) + (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{i}_f \right\} + o_x \\ P_{y_{pp}}(f, p) &= \frac{la}{z_f} \left\{ \left[\mathbf{j}_f - \frac{\mathbf{j}_f \cdot (\mathbf{c} - \mathbf{t}_f)}{z_f} \mathbf{k}_f \right] \cdot (\mathbf{s}_p - \mathbf{c}) + (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{j}_f \right\} + o_y \end{aligned} \tag{26}$$

Here we assume standard camera parameters $l = 1$, $a = 1$, and $(o_x, o_y) = (0, 0)$. The consequences of this action are discussed in section 2.3.9.

Without loss of generality we can simplify our equations by placing the world origin at the object's center of mass so that by definition

$$\mathbf{c} = \frac{1}{P} \sum_{p=1}^{P} \mathbf{s}_p = 0 . \tag{27}$$

This reduces (26) to

$$P_{x_{pp}}(f, p) = \frac{1}{z_f}\left\{\left[\mathbf{i}_f + \frac{\mathbf{i}_f \cdot \mathbf{t}_f}{z_f}\mathbf{k}_f\right] \cdot \mathbf{s}_p - (\mathbf{t}_f \cdot \mathbf{i}_f)\right\}$$
$$P_{y_{pp}}(f, p) = \frac{1}{z_f}\left\{\left[\mathbf{j}_f + \frac{\mathbf{j}_f \cdot \mathbf{t}_f}{z_f}\mathbf{k}_f\right] \cdot \mathbf{s}_p - (\mathbf{t}_f \cdot \mathbf{j}_f)\right\} \tag{28}$$

These equations can be rewritten as

$$P_{x_{pp}}(f, p) = \mathbf{m}_f \cdot \mathbf{s}_p + x_f \qquad P_{y_{pp}}(f, p) = \mathbf{n}_f \cdot \mathbf{s}_p + y_f \tag{29}$$

where

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \tag{30}$$

$$x_f = -\frac{\mathbf{t}_f \cdot \mathbf{i}_f}{z_f} \qquad y_f = -\frac{\mathbf{t}_f \cdot \mathbf{j}_f}{z_f} \tag{31}$$

$$\mathbf{m}_f = \frac{\mathbf{i}_f - x_f\mathbf{k}_f}{z_f} \qquad \mathbf{n}_f = \frac{\mathbf{j}_f - y_f\mathbf{k}_f}{z_f} \tag{32}$$

2.3.2. Relation of Paraperspective and Affine Models

The affine camera model has become popular among vision researchers because, like paraperspective projection, it can be described by linear equations. Many researchers have begun referring to the paraperspective model as “a special case of the affine camera model,” since both can be described by bilinear equations. This statement can be true or false depending on what is meant by “the affine camera model”. When only a single image is considered, one can use the phrase “the affine camera model” unambiguously. However, when multiple images are considered simultaneously one must be careful to distinguish between two variations of the affine camera model commonly in use; the unrestricted affine camera model, and the fixed-intrinsic affine camera model. This section examines both models and their relations to the paraperspective projection model. It points out that by modeling the position effect and enforcing the constraint that camera calibration parameters do not change throughout the sequence, paraperspective is a better model than affine for shape and motion applications.

In an unrestricted affine camera, the image coordinates are given by

$$P_{ua}(f, p) = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} s_{p1} \\ s_{p2} \\ s_{p3} \end{bmatrix} + \begin{bmatrix} x_f \\ y_f \end{bmatrix} \tag{33}$$

where the m_{ij} are free to take on any values. In motion applications, this matrix is commonly decomposed into a scaling factor, a 2×2 intrinsic parameter matrix, and a 2×3 rotation matrix. The intrinsic parameters matrix contains camera calibration parameters which are considered to remain constant throughout the sequence, while the rotation matrix and scaling factor are allowed to vary with each image. This fixed-intrinsic affine camera is given by

$$P_{fia}(f, p) = \frac{1}{z_f}\begin{bmatrix} 1 & 0 \\ s & a \end{bmatrix}\begin{bmatrix} i_{f1} & i_{f2} & i_{f3} \\ j_{f1} & j_{f2} & j_{f3} \end{bmatrix}\begin{bmatrix} s_{p1} \\ s_{p2} \\ s_{p3} \end{bmatrix} + \begin{bmatrix} x_f \\ y_f \end{bmatrix} \qquad (34)$$

These parameters have the following physical interpretations: the $\mathbf{i}_f$ and $\mathbf{j}_f$ vectors represent the camera orientation in each frame, x_f, y_f, and z_f represent the object translation (z_f is scaled by the camera focal length, x_f and y_f are offset by the image center), a is the pixel aspect ratio, and s is a skew parameter. The skew parameter is non-zero only if the projection rays, while still parallel, do not strike the image plane orthogonally.

The fixed-intrinsic affine model simply assumes that the parameters s and a correspond to intrinsic camera characteristics which do not change throughout the sequence. The other variables are considered to correspond to actual rotation and translation between the object and the camera, and are free to vary from one image to the next, provided $\mathbf{i}_f$ and $\mathbf{j}_f$ remain orthogonal unit vectors.

The paraperspective projection equations can be rewritten, retaining the camera parameters, as

$$P_{pp}(f, p) = \frac{l}{z_f}\begin{bmatrix} 1 & 0 & \dfrac{(o_x - x_f)}{l} \\ 0 & a & \dfrac{(o_y - y_f)}{l} \end{bmatrix}\begin{bmatrix} i_{f1} & i_{f2} & i_{f3} \\ j_{f1} & j_{f2} & j_{f3} \\ k_{f1} & k_{f2} & k_{f3} \end{bmatrix}\begin{bmatrix} s_{p1} \\ s_{p2} \\ s_{p3} \end{bmatrix} + \begin{bmatrix} x_f \\ y_f \end{bmatrix} \qquad (35)$$

or defining $b_f = \frac{o_x - x_f}{l}$ $c_f = \frac{o_y - y_f}{la}$, as

$$P_{pp}(f, p) = \frac{l}{z_f}\begin{bmatrix} 1 & 0 & b_f \\ 0 & a & ac_f \end{bmatrix}\begin{bmatrix} i_{f1} & i_{f2} & i_{f3} \\ j_{f1} & j_{f2} & j_{f3} \\ k_{f1} & k_{f2} & k_{f3} \end{bmatrix}\begin{bmatrix} s_{p1} \\ s_{p2} \\ s_{p3} \end{bmatrix} + \begin{bmatrix} x_f \\ y_f \end{bmatrix} \qquad (36)$$

In [37] Quan shows that this can be reduced to a form identical to that of the fixed-intrinsic affine camera by Householder transformation.

$$P_{pp}(f, p) = \frac{l\sqrt{1+b_f^2}}{z_f}\begin{bmatrix} 1 & 0 \\ a\dfrac{b_f c_f}{1+b_f^2} & a\dfrac{\sqrt{1+b_f^2+c_f^2}}{1+b_f^2} \end{bmatrix}\begin{bmatrix} i'_{f1} & i'_{f2} & i'_{f3} \\ j'_{f1} & j'_{f2} & j'_{f3} \end{bmatrix}\begin{bmatrix} s_{p1} \\ s_{p2} \\ s_{p3} \end{bmatrix} + \begin{bmatrix} x_f \\ y_f \end{bmatrix} \tag{37}$$

Here $\mathbf{i}'_f$ and $\mathbf{j}'_f$ are orthonormal unit vectors not necessarily equal to $\mathbf{i}_f$ and $\mathbf{j}_f$.

Both the fixed-intrinsic-parameter affine camera and the paraperspective models are specializations of the unrestricted affine camera model, yet they are different from each other. The former has a constant skew parameter s, and thus projects all rays onto the image plane at the same angle throughout the sequence. This can be an accurate model if the object does not translate in the image or if the angle is non-perpendicular due to a lens misalignment. Under paraperspective, equation (37) shows that the skew parameter and aspect ratio can vary with each image, meaning that the direction of the image projection varies from image to image. This projection direction depends in a physically realistic manner on the translation of the object in the image relative to the image center. This allows paraperspective to accurately model the position effect, which the fixed-intrinsic affine camera cannot do, while enforcing the constraint that the intrinsic camera calibration parameters remain constant in all images, which the unrestricted affine camera cannot do.

2.3.3. Paraperspective as a Perspective Approximation

Perspective projection is a common model of image projection in use by shape and motion researchers. It models the foreshortening effect as well as the scaling and position effects. However, the equations describing it are non-linear and therefore much more cumbersome. We show in this section that paraperspective projection can be derived mathematically as a linear approximation to the standard perspective projection model.

In Section 2.3.1., we defined paraperspective projection geometrically. We can derive the same equations mathematically as a first-order approximation to the perspective projection equations. The perspective projection of point p onto the image plane in frame f is given by

$$P_p(f, p) = \begin{bmatrix} P_{x_p}(f, p) \\ P_{y_p}(f, p) \end{bmatrix} = \frac{l}{z_{fp}}\begin{bmatrix} \mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) \\ \mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) \end{bmatrix} \tag{38}$$

where

$$z_{fp} = \mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) \tag{39}$$

For simplicity we assume unit focal length, $l = 1$.

We define the term

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \tag{40}$$

and then compute the Taylor series expansion of the above equations about the point

$$z_{fp} \approx z_f \tag{41}$$

yielding

$$\begin{aligned} P_{x_p}(f, p) &= \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(z_{fp} - z_f) + \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(z_{fp} - z_f)^2 + \ldots \\ P_{y_p}(f, p) &= \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(z_{fp} - z_f) + \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(z_{fp} - z_f)^2 + \ldots \end{aligned} \tag{42}$$

We combine equations (39) and (40) to determine that $z_{fp} - z_f = \mathbf{k}_f \cdot \mathbf{s}_p$, and substitute this into equation (42) to produce

$$\begin{aligned} P_{x_p}(f, p) &= \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p) + \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(\mathbf{k}_f \cdot \mathbf{s}_p)^2 + \ldots \\ P_{y_p}(f, p) &= \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p) + \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(\mathbf{k}_f \cdot \mathbf{s}_p)^2 + \ldots \end{aligned} \tag{43}$$

Ignoring all but the first term of the Taylor series yields the equations for scaled orthographic projection (See 2.2.) However, instead of arbitrarily stopping at the first term, we eliminate higher order terms based on the approximation that $|\mathbf{s}_p|^2 / z_f^2 = 0$, which will be accurate when the size of the object is smaller than the distance of the object from the camera. Eliminating these terms reduces the equation (43) to

$$\begin{aligned} P_{x_p}(f, p) &\approx \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{i}_f \cdot (-\mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p) \\ P_{y_p}(f, p) &\approx \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{j}_f \cdot (-\mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p) \end{aligned} \tag{44}$$

Factoring out the $1/z_f$ and expanding the dot-products $\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)$ and $\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)$ gives

$$\begin{aligned} P_{x_p}(f, p) &\approx \frac{1}{z_f}\left(\mathbf{i}_f \cdot \mathbf{s}_p + \frac{\mathbf{i}_f \cdot \mathbf{t}_f}{z_f}(\mathbf{k}_f \cdot \mathbf{s}_p) - (\mathbf{i}_f \cdot \mathbf{t}_f)\right) \\ P_{y_p}(f, p) &\approx \frac{1}{z_f}\left(\mathbf{j}_f \cdot \mathbf{s}_p + \frac{\mathbf{j}_f \cdot \mathbf{t}_f}{z_f}(\mathbf{k}_f \cdot \mathbf{s}_p) - (\mathbf{j}_f \cdot \mathbf{t}_f)\right) \end{aligned} \tag{45}$$

These equations are equivalent to the paraperspective projection equations given by equation (28).

The approximation that $|\mathbf{s}_p|^2 / z_f^2 = 0$ preserves the portion of the second term of the Taylor series expansion of order $(|\mathbf{s}_p||\mathbf{t}_f|) / z_f^2$, while ignoring the portion of the second term of order $|\mathbf{s}_p|^2/z_f^2$ and all higher order terms. Clearly if the translation that the object undergoes is also small, then there is little justification for preserving this portion of the second term and not the other. In such cases, the entire second term can be safely ignored, leaving only the equations for scaled orthographic projection.

Note that we did not explicitly set the world origin at the object's center of mass, as we did in Section 2.3.1. However, the assumption that $|\mathbf{s}_p|^2/z_f^2 = 0$ will be most accurate when the magnitudes of the $\mathbf{s}_p$ are smallest. Since the $\mathbf{s}_p$ vectors represent the vectors from the world origin to the object points, their magnitudes will be smaller when the world origin is located near the object's center of mass.

2.3.4. Decomposition

Notice that equation (29) has a form identical to its counterpart for orthographic projection, equation (2), although the corresponding definitions of x_f, y_f, $\mathbf{m}_f$, and $\mathbf{n}_f$ differ. This enables us to perform the basic decomposition of the matrix in the same manner that Tomasi and Kanade did for the orthographic case. Equating the predicted positions with the observed positions, we combine equation (29), for all points p from 1 to P, and all frames f from 1 to F, into the single matrix equation

$$\begin{bmatrix} u_{11} & \cdots & u_{1P} \\ \cdots & \cdots & \cdots \\ u_{F1} & \cdots & u_{FP} \\ v_{11} & \cdots & v_{1P} \\ \cdots & \cdots & \cdots \\ v_{F1} & \cdots & v_{FP} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1 \\ \cdots \\ \mathbf{m}_F \\ \mathbf{n}_1 \\ \cdots \\ \mathbf{n}_F \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_P \end{bmatrix} + \begin{bmatrix} x_1 \\ \cdots \\ x_F \\ y_1 \\ \cdots \\ y_F \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}, \tag{46}$$

or in short

$$W = MS + T\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}, \tag{47}$$

where W is the $2F \times P$ measurement matrix, M is the $2F \times 3$ motion matrix, S is the $3 \times P$ shape matrix, and T is the $2F \times 1$ translation vector.

Using equations (27) and (29) we can write

$$\begin{aligned} \sum_{p=1}^{P} u_{fp} &= \sum_{p=1}^{P} (\mathbf{m}_f \cdot \mathbf{s}_p + x_f) = \mathbf{m}_f \cdot \sum_{p=1}^{P} \mathbf{s}_p + Px_f = Px_f \\ \sum_{p=1}^{P} v_{fp} &= \sum_{p=1}^{P} (\mathbf{n}_f \cdot \mathbf{s}_p + y_f) = \mathbf{n}_f \cdot \sum_{p=1}^{P} \mathbf{s}_p + Py_f = Py_f \end{aligned} \tag{48}$$

Therefore we can compute x_f and y_f, which are the elements of the translation vector T, immediately from the image data as

$$x_f = \frac{1}{P}\sum_{p=1}^{P} u_{fp} \qquad y_f = \frac{1}{P}\sum_{p=1}^{P} v_{fp} \tag{49}$$

Once we know the translation vector T, we subtract it from W, giving the registered measurement matrix

$$W^* = W - T\begin{bmatrix}1 & \dots & 1\end{bmatrix} = MS \tag{50}$$

Since W^* is the product of two matrices each of rank at most 3, W^* has rank at most 3, just as it did in the orthographic projection case. When noise is present, the rank of W^* will not be exactly 3, but by computing the SVD of W^* and only retaining the largest 3 singular values, we can factor it into

$$W^* = \hat{M}\hat{S}, \tag{51}$$

where $\hat{M}$ is a $2F \times 3$ matrix and $\hat{S}$ is a $3 \times P$ matrix. Using the SVD to perform this factorization guarantees that the product $\hat{M}\hat{S}$ is the best possible rank 3 approximation to W^*, in the sense that it minimizes the sum of squared differences between corresponding elements of W^* and $\hat{M}\hat{S}$.

2.3.5. Normalization

Just as in the orthographic case, the decomposition of W^* into the product of $\hat{M}$ and $\hat{S}$ by equation (51) is only determined up to a linear transformation matrix A. Again, we determine this matrix A by observing that the rows of the motion matrix M (the $\mathbf{m}_f$ and $\mathbf{n}_f$ vectors) must be of a certain form. Taking advantage of the fact that $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ are unit vectors, from equation (32) we observe that

$$|\mathbf{m}_f|^2 = \frac{1 + x_f^2}{z_f^2} \qquad |\mathbf{n}_f|^2 = \frac{1 + y_f^2}{z_f^2} \tag{52}$$

The values of x_f and y_f were computed directly from the image measurement data using equation (49), but we do not know the value of the depth z_f. Thus we cannot impose individual constraints on the magnitudes of $\mathbf{m}_f$ and $\mathbf{n}_f$ as was done in the orthographic factorization method. Instead we adopt the following constraint on the magnitudes of $\mathbf{m}_f$ and $\mathbf{n}_f$

$$\frac{|\mathbf{m}_f|^2}{1 + x_f^2} = \frac{|\mathbf{n}_f|^2}{1 + y_f^2} \tag{53}$$

In the case of orthographic projection, one constraint on $\mathbf{m}_f$ and $\mathbf{n}_f$ was that they each have unit magnitude, as required by equation (10). In the above paraperspective case, we simply require that their magnitudes be in a certain ratio.

There is also a constraint on the angle relationship of $\mathbf{m}_f$ and $\mathbf{n}_f$. From equation (32), and the knowledge that $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ are orthogonal unit vectors,

$$\mathbf{m}_f \cdot \mathbf{n}_f = \frac{\mathbf{i}_f - x_f \mathbf{k}_f}{z_f} \cdot \frac{\mathbf{j}_f - y_f \mathbf{k}_f}{z_f} = \frac{x_f y_f}{z_f^2} \tag{54}$$

The problem with this constraint is that, again, z_f is unknown. We could use either of the two values given in equation (53) for $1/z_f^2$, but in the presence of noisy input data the two will not be exactly equal, so we use the average of the two quantities. We choose the arithmetic mean over the geometric mean or some other measure in order to keep the solution of these constraints linear. Thus our second constraint becomes

$$\mathbf{m}_f \cdot \mathbf{n}_f = x_f y_f \frac{1}{2}\left(\frac{|\mathbf{m}_f|^2}{1 + x_f^2} + \frac{|\mathbf{n}_f|^2}{1 + y_f^2} \right) \tag{55}$$

This is the paraperspective version of the orthographic constraint given by equation (11), which required that the dot product of $\mathbf{m}_f$ and $\mathbf{n}_f$ be zero.

Equations (53) and (55) are homogeneous constraints, which could be trivially satisfied by the solution $\forall f \ \mathbf{m}_f = \mathbf{n}_f = 0$, or $M = 0$. To avoid this solution, we impose the additional constraint

$$|\mathbf{m}_1| = 1 \tag{56}$$

This does not effect the final solution except by a scaling factor.

Equations (53), (55), and (56) gives us $2F + 1$ equations, which are the paraperspective version of the *metric constraints*. We compute the 3×3 matrix A such that $M = \hat{M}A$ best satisfies these metric constraints in the least sum-of-squares error sense. This is a simple problem because the constraints are linear in the 6 unique elements of the symmetric 3×3 matrix $Q = A^T A$. We use the metric constraints to compute Q, compute its Jacobi Transformation $Q = L \Lambda L^T$, where Λ is the diagonal eigenvalue matrix, and as long as Q is positive definite, $A = (L \Lambda^{1/2})^T$. The case of non-positive definite Q is discussed in section 2.3.7.

2.3.6. Shape and Motion Recovery

Once the matrix A has been determined, we compute the shape matrix $S = A^{-1}\hat{S}$ and the motion matrix $M = \hat{M}A$. For each frame f, we now need to recover the camera orientation vectors $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, and $\hat{\mathbf{k}}_f$ from the vectors $\mathbf{m}_f$ and $\mathbf{n}_f$, which are the rows of the matrix M. From equation (32) we see that

$$\hat{\mathbf{i}}_f = z_f \mathbf{m}_f + x_f \hat{\mathbf{k}}_f \qquad \hat{\mathbf{j}}_f = z_f \mathbf{n}_f + y_f \hat{\mathbf{k}}_f \tag{57}$$

From this and the knowledge that $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, and $\hat{\mathbf{k}}_f$ must be orthonormal, we determine that

$$\begin{aligned} \hat{\mathbf{i}}_f \times \hat{\mathbf{j}}_f{}' = \left(z_f \mathbf{m}_f + x_f \hat{\mathbf{k}}_f \right) \times \left(z_f \mathbf{n}_f + y_f \hat{\mathbf{k}}_f \right) &= \hat{\mathbf{k}}_f \\ |\hat{\mathbf{i}}_f| = |z_f \mathbf{m}_f + x_f \hat{\mathbf{k}}_f| &= 1 \\ |\hat{\mathbf{j}}_f| = |z_f \mathbf{n}_f + y_f \hat{\mathbf{k}}_f| &= 1 \end{aligned} \tag{58}$$

Again, we do not know a value for z_f, but using the relations specified in equation (53) and the additional knowledge that $|\hat{\mathbf{k}}_f| = 1$, equation (58) can be reduced to

$$G_f \hat{\mathbf{k}}_f = H_f, \tag{59}$$

where

$$G_f = \begin{bmatrix} (\tilde{\mathbf{m}}_f \times \tilde{\mathbf{n}}_f) \\ \tilde{\mathbf{m}}_f \\ \tilde{\mathbf{n}}_f \end{bmatrix} \qquad H_f = \begin{bmatrix} 1 \\ -x_f \\ -y_f \end{bmatrix} \tag{60}$$

$$\tilde{\mathbf{m}}_f = \sqrt{1 + x_f^2} \frac{\mathbf{m}_f}{|\mathbf{m}_f|} \qquad \tilde{\mathbf{n}}_f = \sqrt{1 + y_f^2} \frac{\mathbf{n}_f}{|\mathbf{n}_f|} \tag{61}$$

We compute $\hat{\mathbf{k}}_f$ simply as

$$\hat{\mathbf{k}}_f = G_f^{-1} H_f \tag{62}$$

and then compute

$$\hat{\mathbf{i}}_f = \tilde{\mathbf{n}}_f \times \hat{\mathbf{k}}_f \qquad \hat{\mathbf{j}}_f = \hat{\mathbf{k}}_f \times \tilde{\mathbf{m}}_f \tag{63}$$

There is no guarantee that the $\hat{\mathbf{i}}_f$ and $\hat{\mathbf{j}}_f$ given by this equation will be orthonormal, because $\mathbf{m}_f$ and $\mathbf{n}_f$ may not have exactly satisfied the metric constraints. Therefore we actually use the orthonormals which are closest to the $\hat{\mathbf{i}}_f$ and $\hat{\mathbf{j}}_f$ vectors given by equation (63). We further refine these values using a non-linear optimization step to find the orthonormal $\hat{\mathbf{i}}_f$ and $\hat{\mathbf{j}}_f$, as well as depth z_f which provide the best fit to equation (63). Due to the arbitrary world coordinate orientation, to obtain a unique solution we then rotate the computed shape and motion to align the world axes with the first frame's camera axes, so that $\hat{\mathbf{i}}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ and $\hat{\mathbf{j}}_1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$.

All that remain to be computed are the translations for each frame. We calculate the depth z_f from equation (53). Since we know x_f, y_f, z_f, $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, and $\hat{\mathbf{k}}_f$, we can calculate $\hat{\mathbf{t}}_f$ using equations (30) and (31).

2.3.7. Normalization Failure

In image sequences containing very high levels of noise, sometimes the normalization step fails. Solving the metric constraints in the manner described in section 2.3.5. produces a matrix Q which has negative eigenvalues. In this case the recovered Q cannot be written as a product A^TA for any A. This indicates a serious degeneracy in the tracking data, and how to handle such situations remains an open research question.

The problem of finding the matrix A which best satisfies the metric constraints was separated into two steps, first computing Q and then computing A from Q, primarily because that solution technique involved only linear operations. However, we should have added the constraint that, in order to be a valid solution, Q must have no negative eigenvalues, since a Q with negative eigenvalues does not correspond to a valid solution to the problem at hand.

Since the metric constraints are linear in the elements of Q, the sum of the squares of the error in the metric constraints as a function of the six elements of Q is quadratic in the elements of Q. A quadratic error surface has a single minimum, which can be computed using standard linear least squares techniques. Suppose we turn the problem into a constrained linear least squares problem by adding the constraint that Q have no negative eigenvalues. This would have the effect of marking some regions of the error surface as "off limits", since those portions of the error surface represent solutions which do not satisfy the additional constraint. If the absolute minimum of the quadratic error surface happens to lie in one of these regions, then the minimum subject to the additional constraint that Q have no negative eigenvalues must lie along the border of that constraint, since there are no other points on the error surface which could be the minimum. Solutions lying along the border of that constraint are solutions with one or more zero-valued eigenvalues.

To test this theory, we developed a system to solve the metric constraints directly for A rather than first solving for Q. This approach guarantees that all solutions are valid solutions, since there are no additional constraints on the members of A. Since these equations are non-linear, we solved them using Levenberg-Marquardt iteration and a random initial value. In cases where the original method produced a Q with non-negative eigenvalues, the nonlinear method produced exactly the same result. However, when the original method produced a Q with one or more negative eigenvalues, in every case the nonlinear method produced a matrix A with one or more zero eigenvalues. This shows that the occurrence of a Q matrix with negative eigenvalues is not an shortcoming in our linear approach to solving the metric constraints, but is fundamentally tied to the metric constraints themselves.

When A has one or more zero eigenvalues, the resulting motion matrix M becomes degenerate, having rank one or two instead of rank three. The physical interpretation of such a motion is a pure rotation of the camera about its optical axis. When the motion matrix has degenerate rank, then the corresponding shape matrix is under-determined; a variety of shape matrices will result in the same observation matrix W since they are being multiplied by a rank two matrix. This corresponds to our expectations, since such degenerate motions provide no shape information at all. In fact, images produced from a rotation about the camera's optical axis will all be simple two-dimensional rotations of each other, so the shape's depth values cannot be determined. In other words, a Q with one or more negative eigenvalues indicates that the best motion solution available is one that is insufficient to compute the object shape.

In practice, this occurs when the third column of the unnormalized motion matrix $\hat{M}$, which is the singular vector corresponding to the third largest singular values of the measurement matrix W, is so corrupted by noise that the normalization step can achieve the best fit by ignoring that column. In other words, the combination of image noise and insufficient camera rotation was large enough to make accurate shape recovery impossible. This also corresponds to our intuition. In the complete absence of noise, even a tiny rotation should suffice to recover the object's shape. When the tracking results are high contaminated by noise, the feature motion caused by a small rotation is indistinguishable from motion due to noise, so accurate shape and motion recovery cannot be expected.

Another possibility is that the object shape is a flat surface or a line. In this case, the shape matrix S will have a rank of 2 or 1, respectively. Even if the rank of the motion matrix M is 3, the measurement matrix, which is the product of M and S, will be of rank of 2 or 1. Therefore the unnormalized motion matrix $\hat{M}$ will also have a reduced rank, and the normalization step may produce a Q matrix with negative eigenvalues. The third component of M has been irretrievably lost by multiplying it by a degenerate matrix. At this point we should simply admit that information has been lost, assume the object is planar, and solve for a 2×2 Q matrix to rotate $\hat{M}$ into the correct form.

An examination of the singular values of the measurement matrix should provide some insight as to the source of the problem. If the third and fourth singular values are near the same magnitude as the second singular value, then noise or perspective distortion have corrupted the matrix and made shape and motion recovery impossible. If they are far smaller than the second singular value, then a flat object shape or insufficient rotation have made shape and motion recovery impossible.

This explanation coincides with observations by other researchers, who reported frequently recovering Q matrices with negative eigenvalues when applying the paraperspective factorization method to sequences with insufficient rotational motion [10]. They also applied nonlinear techniques to solve for the elements of A which best satisfy the metric constraints directly, without first solving for Q. They used Lagrange's method of indeterminate multiplier to replace the last constraint, which fixed the magnitude of $|\mathbf{m}_1| = 1$, with the constraint that $det(A) = 1$, and reported acceptable results. However, their failure to get accurate results using the linear versions of the constraints even in noiseless images with adequate motion makes it impossible to judge whether their method is actually an improvement over the linear approach.

2.3.8. Solution Ambiguity Removal

In order to solve for the shape and motion at the end of Section 2.3.5., we computed the matrix $Q = A^T A$ that best satisfied the metric constraints, and then computed A from Q.

There is a sign ambiguity in this process, since any matrix of the form $A\begin{bmatrix} \pm 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & \pm 1 \end{bmatrix}$ produces the same matrix Q when multiplied by its transpose. Thus there are actually several equally plausible motion and shape matrices, since changing the sign of a column of M and the corresponding row of S still produces a solution that satisfies the metric constraints equally well. This sign ambiguity in the first two columns of M was removed when we aligned the world coordinate axes with the first frame's camera axes, at the end of Section 2.3.6. However, the ambiguity in the third column of M and the third row of S is a genuine ambiguity. There are two equally valid solutions, whose shapes differ only by a reflection about the z-axis.

This is a genuine ambiguity due to the nature of paraperspective projection, not merely a mathematical one. This can be seen by looking at the motion sequence of a rotating cube shown in the first column of Figure 6. The reader can imagine that the thick square is the front square of a cube rotating clockwise when viewed from above, or that the thick square is the back of a cube rotating counterclockwise when viewed from above; both interpretations are plausible. However, in general real image sequences will contain additional queues which should clearly define which of the two solutions is the correct solution. For example, in real images some points will become occluded so the images will actually be either as shown in Figure 6(b), in which the thick square is clearly at the back of the cube, or as in Figure 6(c), in which the thick square is clearly at the front of the cube. By examining the pattern of occlusion in the image sequence, it should be possible to determine which of the two solutions provided by the paraperspective method is the correct solution. Alternatively, perspective foreshortening effects which were ignored during factorization could be used to determine which solution is correct. For example, if the actual images are as shown in Figure 6(d), clearly the thick square is at the front of the object, while if they are as shown in Figure 6(e), then the thick square is at the back of the object. (This cannot be determined by looking at a single image, since clearly the object could be a strangely shaped frustum, but if foreshortening effects are present, one of the two solutions will be more consistent with the feature measurements over entire sequence.) It should be a simple matter to analyze the fill pattern to determine which points are near and which points are far, or to determine which of the two possible solutions is most consistent with the measurement data using a perspective projection model.

2.3.9. Camera Calibration Requirements

When the paraperspective equations were derived in section 2.3.1, standard camera parameters were assumed. The scaling effect of perspective projection depends on the camera focal length, and the position effect of perspective projection depends significantly on the position of the object in the image relative to the center of projection, as well as the focal length. Since paraperspective projection models both of these effects, camera calibration data is

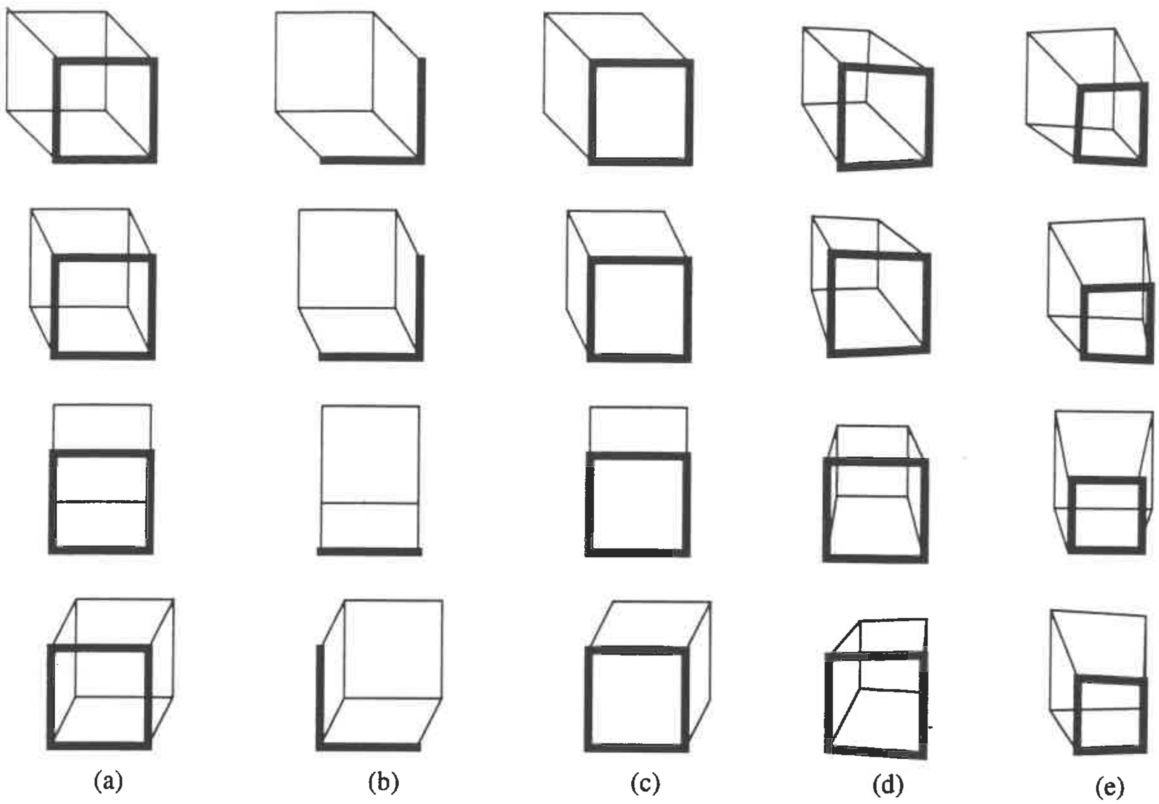


Figure 6 Ambiguity of Solution

(a) Sequence of images with two valid motion and shape interpretations.
(b), (c) Ambiguity removed due to occlusion information.
(d), (e) Ambiguity removed due to perspective distortion of the object.

needed for the paraperspective factorization method to work properly. The image data must be preprocessed by shifting the image so that the center of projection is at $(0, 0)$ and by scaling the measurements to have unit focal length and aspect ratio.

The scaled orthographic factorization method does not model the position effect, and therefore does not require that the center of projection be known. It does model the scaling effect, for which the focal length is required. However, the focal length enters the equations only multiplied by the depth z_f. Therefore an incorrect focal length does not effect the quality of the shape or rotation recovery; its only effect is to scale the recovered depth translation values by a constant factor. Thus the focal length is needed only to evaluate the magnitude of the depth translation relative to the other translations and the object size. The aspect ratio. however, must be known for both the orthographic and scaled orthographic factorization methods, since the $\mathbf{i}_f$ and $\mathbf{j}_f$ vectors will not have equal magnitudes if the aspect ratio is not unity.

As the focal length is increased, the magnitude of the position effect is diminished, and paraperspective projection gradually approaches scaled orthographic projection. Thus if the focal length is not precisely known, using an overestimate of the focal length with the paraperspective method will produce results no worse than using the scaled orthographic method.

2.4. Separate Perspective Refinement Method

This section presents an iterative method used to recover the shape and motion using a perspective projection model. The method requires accurate initial shape and motion estimates, as can be provided by paraperspective factorization, and then refines those estimates to remove distortion caused by unmodeled foreshortening. Shape and motion are refined separately, hence we call it "separate perspective refinement." First the motion is held fixed while the shape is refined, and subsequently the refined shape is held constant while the motion is refined. When the initial approximation is fairly accurate, this is a simpler and more efficient solution than [38], in which all parameters are refined simultaneously. However, it converges more slowly or even fails to definitively converge when the initial values are inaccurate. Although our algorithm was developed independently and handles the full three dimensional case, this method is quite similar to a two dimensional algorithm reported in [39].

2.4.1. Perspective Projection

Under perspective projection, often referred to as the pinhole camera model, object points are projected directly towards the focal point of the camera. An object point's image coordinates are determined by the position at which the line connecting the object point with the camera's focal point intersects the image plane, as illustrated in Figure 7.

Simple geometry using similar triangles produces the perspective projection equations

$$P_p(f,p) = \begin{bmatrix} P_{x_p}(f,p) \\ P_{y_p}(f,p) \end{bmatrix} = \begin{bmatrix} l\dfrac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{\mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)} + o_x \\ la\dfrac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{\mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)} + o_y \end{bmatrix} \tag{64}$$

Assuming standard camera parameters, we rewrite the equations in the form

$$P_{x_p}(f,p) = \frac{\mathbf{i}_f \cdot \mathbf{s}_p + x_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f} \qquad P_{y_p}(f,p) = \frac{\mathbf{j}_f \cdot \mathbf{s}_p + y_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f} \tag{65}$$

where

$$x_f = -\mathbf{i}_f \cdot \mathbf{t}_f \qquad y_f = -\mathbf{j}_f \cdot \mathbf{t}_f \qquad z_f = -\mathbf{k}_f \cdot \mathbf{t}_f \tag{66}$$

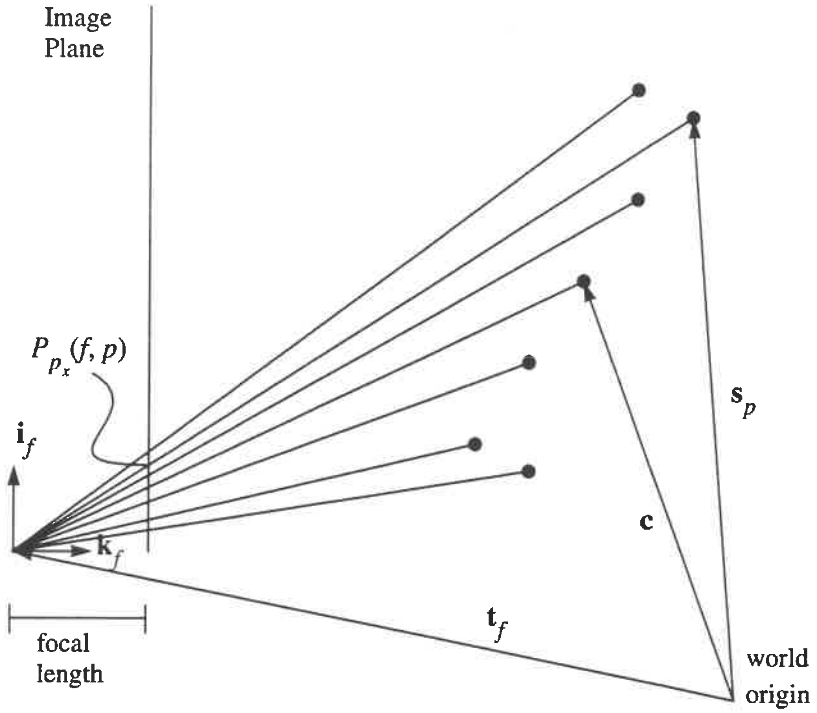


Figure 7. Perspective Projection in two dimensions

2.4.2. Review of Levenberg-Marquardt Minimization

In general, a non-linear least-squares minimization problem is formulated as finding the vector of unknowns $\mathbf{a}$ that minimizes differences between a set of observed values y_i, and the values $Y_i(\mathbf{a})$ predicted by the model and $\mathbf{a}$. The total error ε is measured by summing the squares of the differences between these values, possibly weighting each error term by an uncertainty measure σ_i, so that

$$\varepsilon = \sum_{i=1}^{N} \frac{[y_i - Y_i(\mathbf{a})]^2}{\sigma_i^2} \tag{67}$$

The goal is to find the set of variable values $\mathbf{a}$ which minimizes this total error sum. Except for global search techniques, which exhaustively search for an absolute minimum, most non-linear minimization methods require an initial value for the variables $\mathbf{a}$, and then iteratively adjust or refine that set of values to reduce the error ε.

Perhaps the most common technique for performing this error minimization is the gradient descent method. From the initial set of values, the derivative of the error with respect to each variable a_i is computed to form the gradient vector. The method then adjusts $\mathbf{a}$ by moving some distance in the direction of the gradient, so that the solution is adjusted in a “downhill” direction which reduces the error. Sometimes a constant step size is used, while more often it is varied over time. No matter how non-linear an error surface is, as long as the step size is made small enough, the error can always be decreased by moving in the direction of the gradient, until a minimum or point of zero gradient is reached.

A more complicated technique is the inverse-Hessian method, which approximates the local shape of the error surface as a multi-dimensional quadratic bowl. Given a point on the error surface, the slope of the surface in all directions, and the second derivatives, the method moves to the point on the surface which, if the surface were actually a quadratic bowl, would represent the minimum point of that bowl. This enables much quicker convergence than the gradient descent method, since as long as the error surface is accurately modeled by a quadratic, it can “jump” directly to the minimum point rather than wander towards the minimum through a series of steps. However, if the error surface in the vicinity of the current point is not accurately modeled by a quadratic bowl, using this method can cause very large jumps and instabilities.

Levenberg-Marquardt combines the two methods. When the solution is far from the minimum, in areas not well-modeled by the quadratic assumption, gradient descent is used to keep moving towards the minimum. As the minimum is approached, however, the quadratic approximation should become more accurate, so the inverse Hessian method is used. Marquardt’s insight was that a smooth weighting between the two methods can be achieved simply by multiplying the diagonal elements of the Hessian matrix by $1 + \lambda$, where the λ parameter is varied dynamically. High values of λ cause the method to behave mostly like the gradient descent method, while low values of λ cause the method to approach the inverse Hessian method.

Press, et al. [31] recommends using the approximate Hessian matrix α, defined by

$$\alpha_{kl} \equiv \frac{1 \partial^2 \varepsilon}{2 \partial a_k \partial a_l} \approx \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \frac{\partial Y_i \partial Y_i}{\partial a_k \partial a_l} \qquad (68)$$

rather than the full Hessian matrix. The full Hessian matrix includes the second derivatives of Y_i with respect to the a_k. Inclusion of these terms can sometimes improve the method’s convergence rate, but can also lead to unstable behavior. The weighting between the gradient descent method and the inverse Hessian method is performed by introducing the weighted Hessian matrix

$$\alpha_{kl}' \equiv \begin{cases} (1 + \lambda)\,\alpha_{kl} & l = k \\ \alpha_{kl} & l \neq k \end{cases} \qquad (69)$$

The vector β is defined by

$$\beta_k \equiv -\frac{1}{2}\frac{\partial \varepsilon}{\partial a_k} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2}(y_i - Y_i(\mathbf{a}))\frac{\partial Y_i}{\partial a_k} \tag{70}$$

At each iteration of the algorithm, the vector containing the variable values $\mathbf{a}$ is updated by solving a linear system of equations for the step vector $\delta\mathbf{a}$.

$$\alpha'\delta\mathbf{a} = \beta \tag{71}$$

The step vector is added to the parameter set $\mathbf{a}$ to determine the new position. If the value of ε at the new position $\mathbf{a} + \delta\mathbf{a}$ is lower than the prior value, then $\mathbf{a} + \delta\mathbf{a}$ is accepted as the new estimate and the parameter λ is decreased, so that the next step will more closely follow the inverse Hessian method. If the error at $\mathbf{a} + \delta\mathbf{a}$ is greater than the error at $\mathbf{a}$, then the step is rejected and λ is increased, so that the next step will be more steeply downhill. This process is repeated until convergence is achieved, defined as consecutive iterations producing little or no reduction in ε.

A more thorough review of the Levenberg-Marquardt method, as well as gradient descent and inverse Hessian methods, can be found in [31] or [12].

2.4.3. Iterative Solution Method

The perspective projection equations are non-linear in the shape and motion variables. We formulate the problem as an non-linear least squares problem in the motion and shape variables, in which we seek to minimize the error

$$\varepsilon = \sum_{f=1}^{F}\sum_{p=1}^{P}\{\left(u_{fp} - P_{x_p}(f,p)\right)^2 + \left(v_{fp} - P_{y_p}(f,p)\right)^2\} \tag{72}$$

or equivalently

$$\varepsilon = \sum_{f=1}^{F}\sum_{p=1}^{P}\left\{\left(u_{fp} - \frac{\mathbf{i}_f \cdot \mathbf{s}_p + x_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f}\right)^2 + \left(v_{fp} - \frac{\mathbf{j}_f \cdot \mathbf{s}_p + y_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f}\right)^2\right\}. \tag{73}$$

If each $\mathbf{i}_f$, $\mathbf{j}_f$, $\mathbf{k}_f$, x_f, y_f, and z_f were allowed to vary arbitrarily, there would be 12 motion variables for each frame, since each of the camera axis vectors contains three elements. However, we can enforce the constraint that $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ are orthogonal unit vectors by writing them as functions of three independent rotational parameters θ_f, φ_f, and ω_f.

$$\left[\mathbf{i}_f\ \mathbf{j}_f\ \mathbf{k}_f\right] = \begin{bmatrix} \cos\theta_f \cos\varphi_f & (\cos\theta_f \sin\varphi_f \sin\omega_f - \sin\theta_f \cos\omega_f) & (\cos\theta_f \sin\varphi_f \cos\omega_f + \sin\theta_f \sin\omega_f) \\ \sin\theta_f \cos\varphi_f & (\sin\theta_f \sin\varphi_f \sin\omega_f + \cos\theta_f \cos\omega_f) & (\sin\theta_f \sin\varphi_f \cos\omega_f - \cos\theta_f \sin\omega_f) \\ -\sin\varphi_f & \cos\varphi_f \sin\omega_f & \cos\varphi_f \cos\omega_f \end{bmatrix} \tag{74}$$

This gives six motion parameters for each frame (x_f, y_f, z_f, θ_f, φ_f, and ω_f) and three shape parameters for each point ($\mathbf{s}_p = \begin{bmatrix} s_{p1} & s_{p2} & s_{p3} \end{bmatrix}$) for a total of $6F + 3P$ variables.

We could apply any one of a number of non-linear techniques to minimize the error ε as a function of these $6F + 3P$ variables. Such methods begin with a set of initial variable values, and iteratively refine those values to reduce the error. If there are many points and many frames, however, this can become a huge optimization problem. Our method takes advantage of the particular structure of the equations by separately refining the shape and motion parameters. First the shape is held constant while solving for the motion parameters which minimize the error. Then the motion is held constant while solving for the shape parameters which minimize the error. This process is repeated until an iteration produces no significant reduction in the total error ε.

While holding the shape constant, the minimization with respect to the motion variables can be performed independently for each frame. Each of these minimizations requires solving an overconstrained system of six variables in P equations. Likewise while holding the motion constant, we can solve for the shape separately for each point by solving a system of $2F$ equations in three variables. This not only reduces the problem to manageable complexity, but as pointed out by Szeliski and Kang in [39], it lends itself well to parallel implementation.

We perform the individual minimizations, fitting six motion variables to P equations or fitting three shape variables to $2F$ equations, using the Levenberg-Marquardt method described in the previous section. Since we know the mathematical form of the expression of ε, the Hessian matrix is easily computed by taking derivatives of ε with respect to each variable.

A single iteration of the Levenberg-Marquardt method requires a single inversion of a 6×6 matrix when refining a single frame of motion, or a single inversion of a 3×3 matrix when refining the position of a single point. Generally about six iterations were required for convergence of a single point or frame refinement, so a complete refinement step requires $6P$ inversions of 3×3 matrices and $6F$ inversions of 6×6 matrices.

In theory we do not actually need to vary all $6F + 3P$ variables, since the solution is only determined up to a scaling factor, the world origin is arbitrary, and the world coordinate orientation is arbitrary. We could choose to arbitrarily fix each of the first frame's rotation variables at zero degrees, and similarly fix some shape or translation parameters to reduce the problem to $6F + 3P - 7$ variables. However, it was experimentally confirmed that the algorithm converged significantly faster when all shape and motion parameters are all allowed to vary. The final shape and translation are then adjusted to place the origin at the object's center of mass and scale the solution so that the depth in the first frame is 1. This shape and the final motion are then rotated so that $\hat{\mathbf{i}}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ and $\hat{\mathbf{j}}_1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$, or equivalently, so that $\theta_1 = \varphi_1 = \omega_1 = 0$.

A common drawback of iterative methods on complex non-linear error surfaces is that they do not find the global minimum, and therefore the final result can be highly dependent on the initial value. Taylor, Kriegman, and Anandan [39] require some basic odometry measurements as might be produced by a navigation system to use as initial values for their motion parameters, and use the 2D shape of the object in the first image frame, assuming constant depth, as their initial shape. To avoid the requirement for odometry measurements, which will not be available in many situations, we use the paraperspective factorization method to supply initial values to the iterative perspective refinement process.

2.5. Comparison

In this section we compare the performance of our new paraperspective factorization and perspective refinement methods with the previous orthographic factorization method. The comparison also includes the scaled orthographic projection in order to demonstrate the importance of modeling the position effect for objects at close range. Our results show that the paraperspective factorization method is a vast improvement over the orthographic method, and underscore the importance of modeling both the scaling and position effects. We also show the results of perspectively refining the paraperspective solution. This demonstrates that modeling of perspective distortion is important primarily for accurate shape recovery of objects at close range.

2.5.1. Data Generation

The synthetic feature point sequences used for comparison were created by moving a known "object" - a set of 3D points - through a known motion sequence. We tested three different object shapes, each containing approximately 60 points. Each test run consisted of 60 image frames of an object rotating through a total of 30 degrees each of roll, pitch, and yaw. The "object depth" - the distance from the camera's focal point to the front of the object - in the first frame was varied from 3 to 60 times the object size. In the sequences whose graphs are shown in the following sections, the object also translated across the field of view by a distance of one object size horizontally and vertically, and translated away from the camera by half its initial distance from the camera. For example, when the object's depth in the first frame was 3.0, its depth in the last frame was 4.5. Each "image" was created by perspectively projecting the 3D points onto the image plane, for each sequence choosing the largest focal length that would keep the object in the field of view throughout the sequence. The coordinates in the image plane were perturbed by adding gaussian noise, to model tracking imprecision. The standard deviation of the noise was 2 pixels (assuming a 512x512 pixel image), which we consider to be a rather high noise level from our experience processing real image sequences. For each combination of object, depth, and noise, we performed three tests, using different random noise each time. Two sample synthetic image sequences are shown in Figure 8.

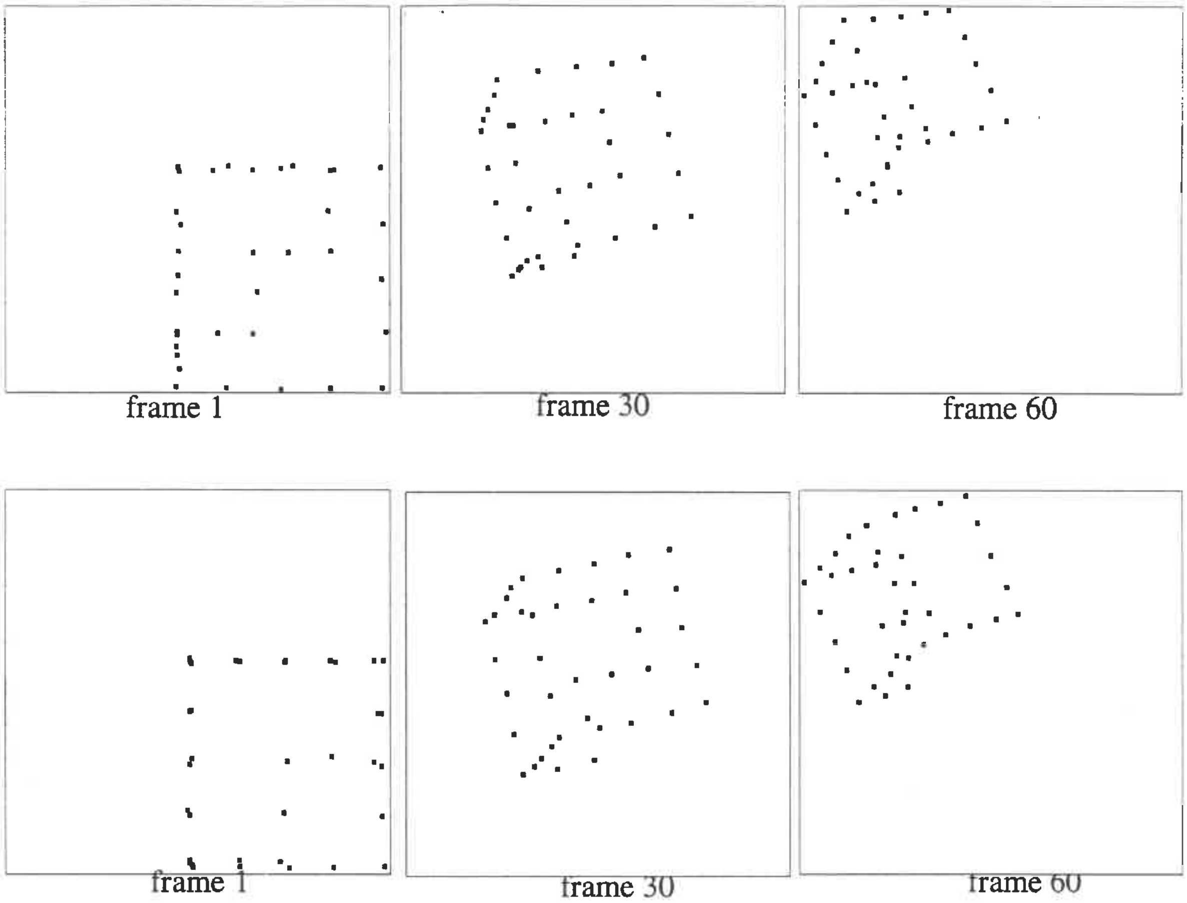


Figure 8. Sample Synthetic Image Sequences

The object used to generate these images consisted of the edges three sides of a cube of unit size. The far side was given a "notch" in one corner to make it easier to visually distinguish the front of the object from the back. In the top sequence, the depth (distance from the camera to the object) in the first frame was 3 and the depth in the last frame was 4.5. In the bottom sequence, the depth in the first frame was 30 and the depth in the last frame was 45. Gaussian noise with a standard deviation of 2 pixels was added to each image position.

2.5.2. Error Measurement

We ran each of the three factorization methods on each synthetic sequence and measured the rotation error, shape error, X-Y offset error, and Z offset (depth) error. The rotation error is the root-mean-square (RMS) of the size in radians of the angle by which the computed camera coordinate frame must be rotated about some axis to produce the known camera orientation. The shape error is the RMS error between the known and computed 3D point coordinates. Since the shape and translations are only determined up to scaling factor, we first scaled the computed shape by the factor which minimizes this RMS error. The term "offset" refers to the translational component of the motion as measured in the camera's coordinate frame rather than in world coordinates; the X offset is $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{i}}_f$, the Y offset is $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{j}}_f$, and the Z offset is $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{k}}_f$. These error measures are used directly rather than using them to first solve for $\mathbf{t}_f$ so that errors reported in the translation are not also influenced by the errors in the recovered orientation variables. The X-Y offset error and Z offset error are the RMS error between the known and computed offset; like the shape error, we first scaled the computed offset by the scale factor that minimized the RMS error. Note that the orthographic factorization method supplies no estimation of translation along the camera's optical axis, so the Z offset error cannot be computed for that method.

2.5.3. Discussion of Results

Figure 9 shows the average errors in the solutions computed by the various methods, as a functions of object depth in the first frame. We see that the paraperspective method performs significantly better than the orthographic factorization method regardless of depth, because orthography cannot model the scaling effect, which occurs due to the motion along the camera's optical axis. The figure also shows that the paraperspective method performs substantially better than scaled orthographic method at close range, while the errors from the two methods are nearly the same when the object is distant. This confirms the importance of modeling the position effect when objects are near the camera. Perspective refinement of the paraperspective results only marginally improves the recovered camera motion, while it significantly improves the accuracy of the computed shape, even up to fairly distant ranges. This implies that unmodeled perspective distortion in the images effects primarily the shape portion of the paraperspective factorization method's solution, and that the effects are significant only when the object is within a certain distance of the camera.

The separate refinement method often did not converge to a clear minimum. Instead, after significantly improving the solution, it began reducing the error by increasingly tiny increments, at which point iteration was halted. Unfortunately "refinement" of the true shape and motion values, also shown in Figure 9, is not an alternative in real systems. However, starting at the correct solution shows what is essentially the best we can hope for using the current least squares error formulation. Its closeness to the results of perspectively refining the paraperspective solution shows that the latter method approaches the best results that can be expected without taking additional knowledge into account. The fact that the two lines of the graph are not identical could indicate that the error surface contains local minima, or it could simply be the result of our halting refinement when the error reduction slows drastically.

In other experiments, whose graphs are not shown, translation in the image plane was eliminated and the object was kept centered in the image. In these experiments, the paraperspective method and the scaled orthographic method performed equally well, as we would expect since such image sequences contain no position effects. In still other experiments, the depth translation was eliminated as well, so that the object remained at a fixed distance from the camera. The orthographic factorization method performed very well in these experi-

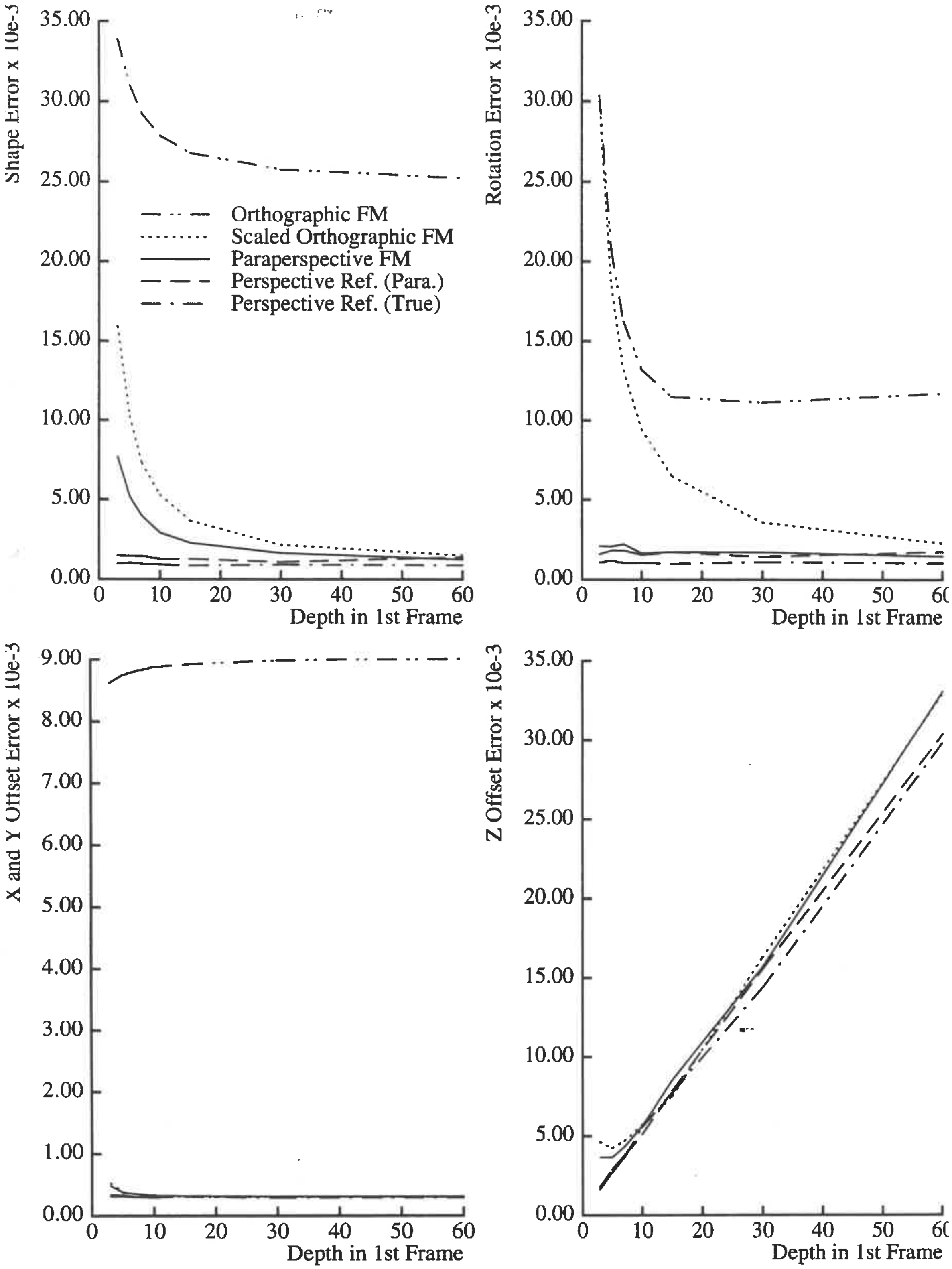


Figure 9. Methods compared for a typical case

noise standard deviation = 2 pixels

ments, and the paraperspective factorization method provided no significant improvement since such sequences contain neither scaling effects nor position effects.

When the object was placed extremely close to the camera, i.e. when the depth is one or two times the object size, normalization failure occasionally became a problem. Even when the paraperspective normalization succeeded, separate refinement was unable to remove the distortions in the object.

The methods were all implemented in C using double floating point precision versions of the *Numerical Recipes in C* routines for most of the numerical processing. The factorization methods each required about 4 seconds to solve the systems of 60 frames and 60 points on a Sparc 5/85 workstation, with most of the time spent computing the SVD of the measurement matrix. The separate refinement method required about 85 seconds to solve the same system.

2.6. Analysis of Paraperspective Method using Synthetic Data

Now that we have shown the advantages of the paraperspective factorization method over the previous orthographic factorization method, we further analyze the performance of the paraperspective method to determine its behavior at various depths and its robustness with respect to noise. The synthetic sequences used in these experiments were created in the same manner as in the previous section, except that the standard deviation of the noise was varied from 0 to 4.0 pixels.

In Figure 10, we see that at high depth values, the error in the solution is roughly proportional to the level of noise in the input, while at low depths the error is inversely related to the depth. This occurs because at low depths, perspective distortion of the object's shape is the primary source of error in the computed results. At higher depths, perspective distortion of the object's shape is negligible, and noise becomes the dominant cause of error in the results. For example, at a noise level of 1 pixel, the rotation and XY-offset errors are nearly invariant to the depth once the object is farther from the camera than 10 times the object size. The shape results, however, appear sensitive to perspective distortion even at depths of 30 or 60 times the object size.

2.7. Paraperspective Factorization Applied to Laboratory Image Sequence

A hotel model was imaged by a camera mounted on a computer-controlled movable platform. The camera motion included substantial translation away from the camera and across the field of view, as shown in Figure 11. The feature tracker (described in [42], essentially a non-hierarchical version of the tracker described in Chapter 3) automatically identified and tracked 197 points throughout the sequence of 181 images.

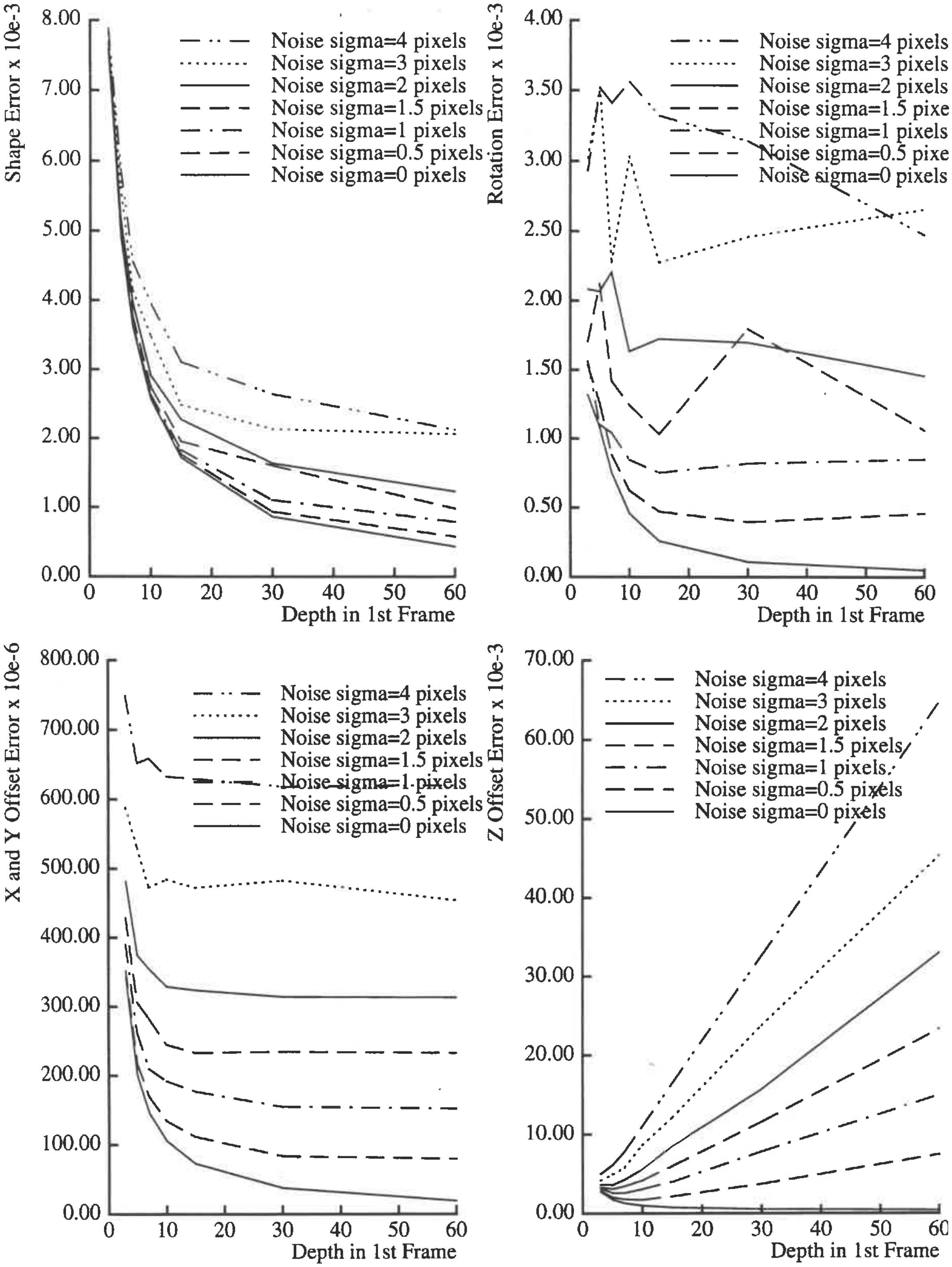
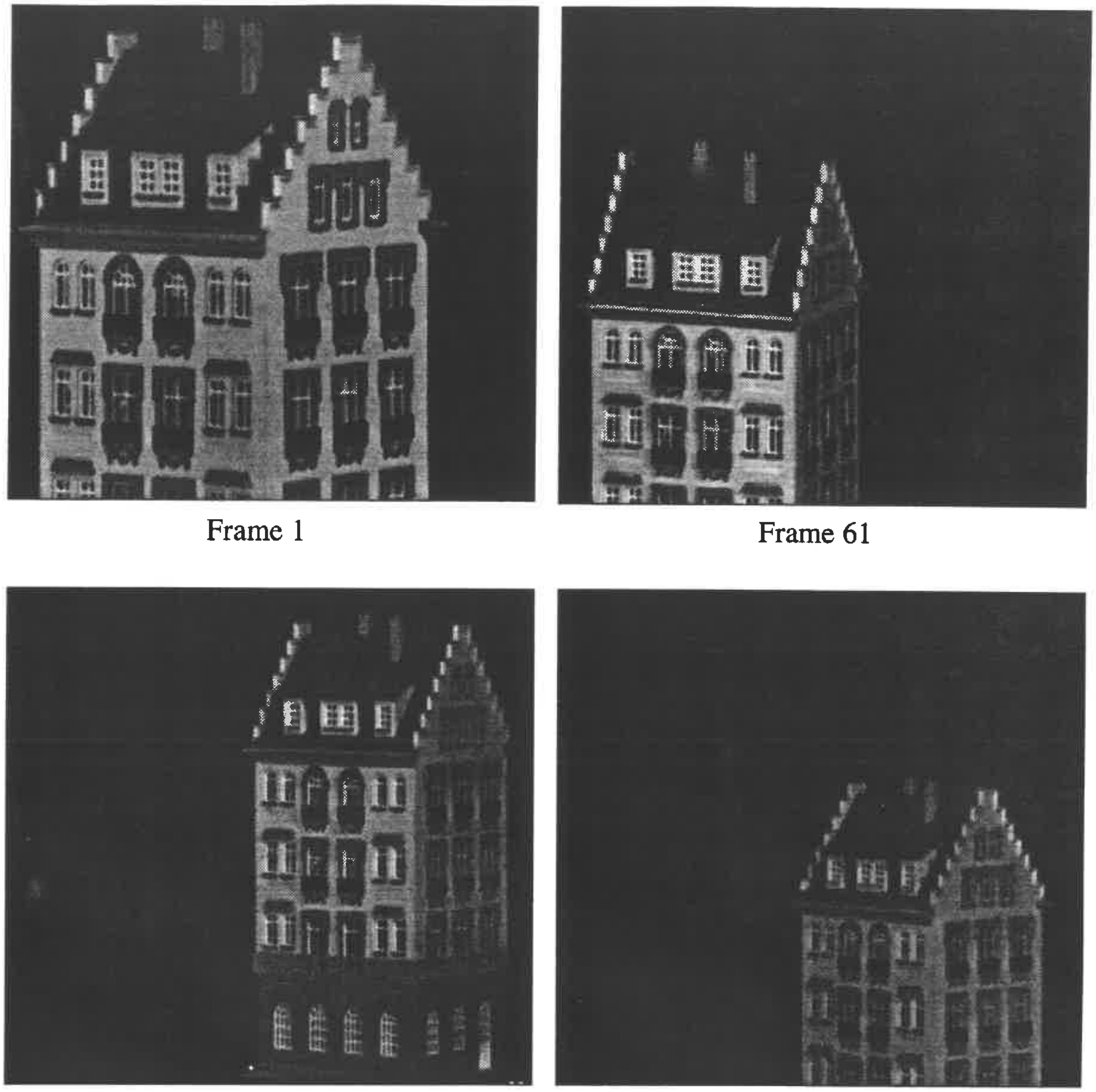


Figure 10. Paraperspective shape and motion recovery by noise level



Frame 1

Frame 61

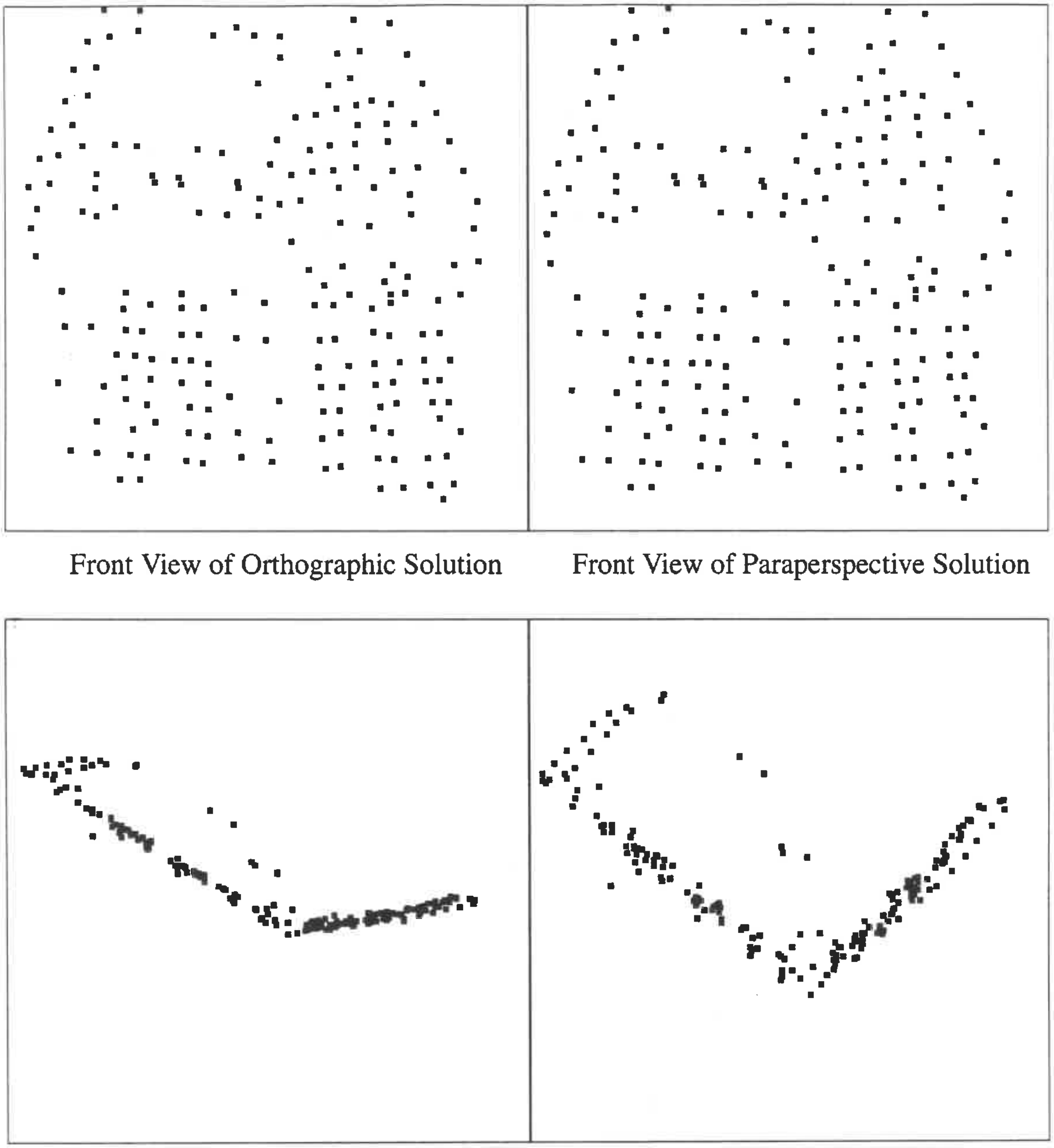
Frame 121

Frame 151

Figure 11. Hotel Model Image Sequence

Both the paraperspective factorization method and the orthographic factorization method were tested with this sequence. The shape recovered by the orthographic factorization method was rather deformed (see Figure 12) and the recovered motion incorrect, because the method could not account for the scaling and position effects which are prominent in the sequence. The paraperspective factorization method, however, models these effects of perspective projection, and therefore was able to determine the correct Euclidean shape and motion.

Several features in the sequence were poorly tracked, and as a result their recovered 3D positions were incorrect. While they did not disrupt the overall solution greatly, we found



Front View of Orthographic Solution

Front View of Paraperspective Solution

Top View of Orthographic Solution

Top View of Paraperspective Solution

Figure 12. Comparison of Orthographic and Paraperspective Shape Results

that we could achieve improved results by automatically removing these features in the following manner. Using the recovered shape and motion, we computed the reconstructed measurement matrix W^{recon}, and then eliminated from W those features for which the average error between the elements of W and W^{recon} was more than twice the average such error. We then ran the shape and motion recovery again, using only the remaining 179 features. Eliminating the poorly tracked features decreased errors in the recovered rotation about the camera's x-axis in each frame by an average of 0.5 degrees, while the errors in the other rotation parameters were also slightly improved. The final rotation values are shown in Figure 13, along with the values we measured using the camera platform. The computed rotation about the camera x-axis, y-axis, and z-axis was always within 0.29 degrees, 1.78 degrees, and 0.45 degrees of the measured rotation, respectively.

The system of 181 frames and 197 points required 42 seconds to solve on a Sparc 5/85. This is in addition to the feature tracking, which required 16 seconds to automatically detect the features and 6 seconds per image to perform the tracking, for a total of 100 seconds or over 18 minutes.

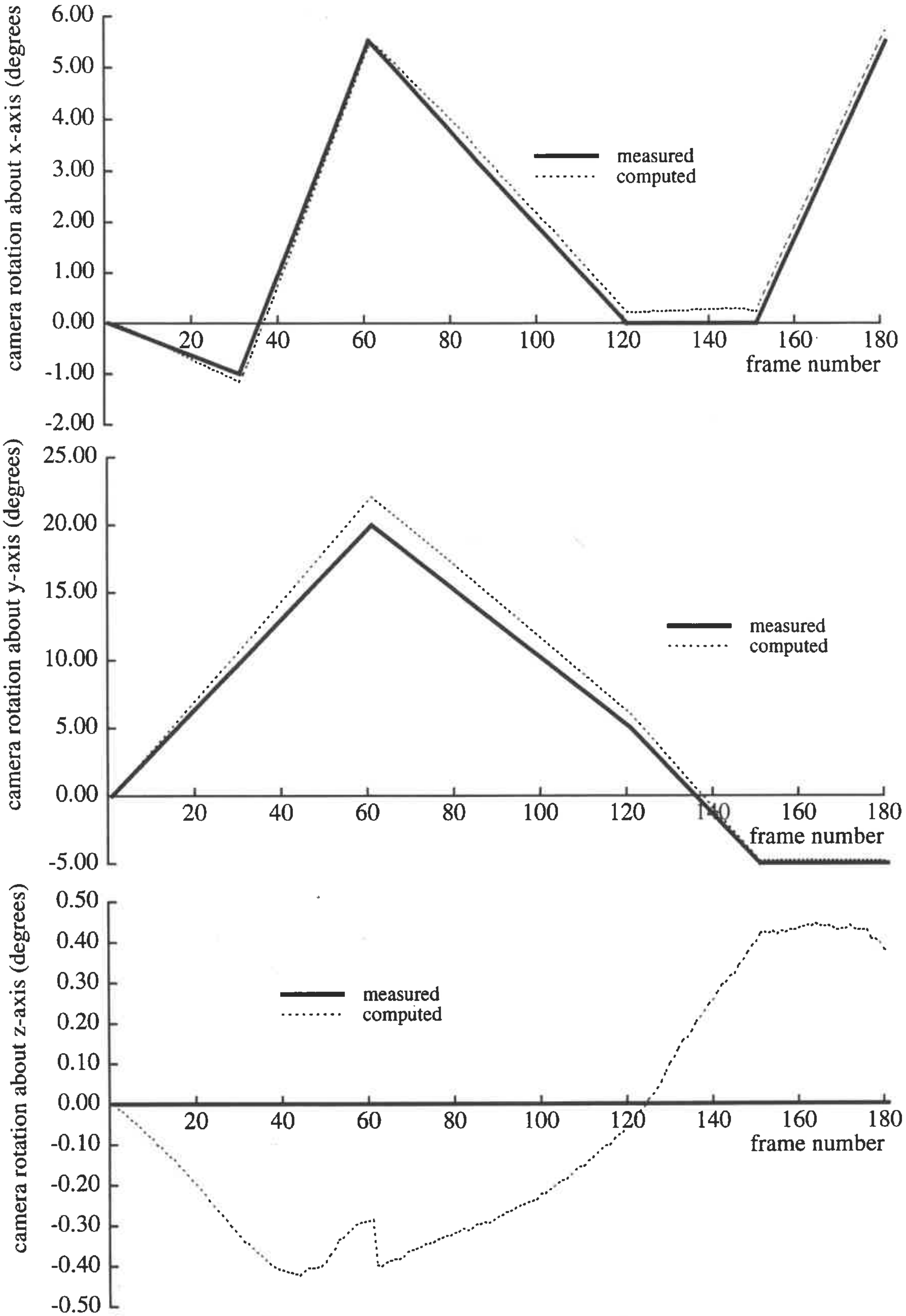


Figure 13. Hotel Model Rotation Results

3. Confidence-Weighted Tracking and Shape Recovery

This chapter shows how the addition of confidence information to the factorization method can be used to improve both the tracking and the shape recovery aspects of the factorization method. We first explain our basic tracking method, and show how to estimate the "trackability" of a region of the image to determine whether it would be a good feature. This trackability estimate is combined with the tracking residue to provide confidence estimates for our tracking measures. We then show how these confidence values are used to provide computationally inexpensive inter-level smoothing in our multi-resolution tracking approach. Finally, we generalize the decomposition step of the factorization method to incorporate confidence measures for each feature. This extension allows the method to be applied to sequences in which features become occluded or leave the field of view, and improves the accuracy of the shape recovery by weighting the measurements of precisely tracked features more heavily than those for poorly tracked features.

3.1. Image Motion Estimation

We use the tracking method developed by Lucas and Kanade [17]. This method assumes a translational model of image feature motion, so that the intensity values of any given region of an image do not change, but merely shift from one position to another. Given an intensity feature template F defined over some region R, and an intensity image I, we wish to find the translation $\mathbf{h}$ which minimizes the tracking residue E defined as

$$E = \sum_{\mathbf{x} \in R} [I(\mathbf{x}+\mathbf{h}) - F(\mathbf{x})]^2 \tag{75}$$

The minimum error occurs when the derivative of E with respect to $\mathbf{h}$ is zero, or

$$\frac{\partial}{\partial \mathbf{h}} E = \sum_{\mathbf{x} \in R} 2 [I(\mathbf{x}+\mathbf{h}) - F(\mathbf{x})] \frac{\partial}{\partial \mathbf{h}} I(\mathbf{x}+\mathbf{h}) = 0 \tag{76}$$

Making the linear approximation that $I(\mathbf{x}+\mathbf{h}) \approx I(\mathbf{x}) + \mathbf{h}\frac{\partial}{\partial \mathbf{x}} I(\mathbf{x})$, we can solve for the translation $\mathbf{h}$ as

$$\mathbf{h} = \left[\sum_{\mathbf{x} \in R} \left(\frac{\partial I}{\partial \mathbf{x}}\right)^T [F(\mathbf{x}) - I(\mathbf{x})]\right] \left[\sum_{\mathbf{x} \in R} \left(\frac{\partial I}{\partial \mathbf{x}}\right)\left(\frac{\partial I}{\partial \mathbf{x}}\right)^T\right]^{-1} \tag{77}$$

Because a linear approximation for $I(\mathbf{x}+\mathbf{h})$ was used, the above formula provides only an approximate solution for $\mathbf{h}$. However, by iteratively applying this step, the method quickly converges to the $\mathbf{h}$ which yields the minimum error E, using the following formulation.

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \left[\sum_{\mathbf{x} \in R} \left(\frac{\partial I}{\partial \mathbf{x}}\right)^T\Bigg|_{\mathbf{x}+\mathbf{h}_n} [F(\mathbf{x}) - I(\mathbf{x}+\mathbf{h}_n)]\right] \left[\sum_{\mathbf{x} \in R} \left(\frac{\partial I}{\partial \mathbf{x}}\right)\left(\frac{\partial I}{\partial \mathbf{x}}\right)^T\Bigg|_{\mathbf{x}+\mathbf{h}_n}\right]^{-1} \tag{78}$$

Here $\mathbf{h}_0$, the initial estimate, can be taken as zero if only small displacements are involved.

This method is computationally simple. At each iteration, five quantities are computed by summation over the image region.

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}^T = \begin{bmatrix} \sum_{\mathbf{x} \in R} \left.\dfrac{\partial I}{\partial x_1}\right|_{\mathbf{x}+\mathbf{h}_n} [F(\mathbf{x}) - I(\mathbf{x}+\mathbf{h}_n)] \\ \sum_{\mathbf{x} \in R} \left.\dfrac{\partial I}{\partial x_2}\right|_{\mathbf{x}+\mathbf{h}_n} [F(\mathbf{x}) - I(\mathbf{x}+\mathbf{h}_n)] \end{bmatrix}^T \tag{79}$$

$$G = \begin{bmatrix} G_{11} & G_{12} \\ G_{12} & G_{22} \end{bmatrix} = \begin{bmatrix} \sum_{\mathbf{x} \in R} \left.\left(\dfrac{\partial I}{\partial x_1}\right)^2\right|_{\mathbf{x}+\mathbf{h}_n} & \sum_{\mathbf{x} \in R} \left.\left(\dfrac{\partial I}{\partial x_1}\right)\left(\dfrac{\partial I}{\partial x_2}\right)\right|_{\mathbf{x}+\mathbf{h}_n} \\ \sum_{\mathbf{x} \in R} \left.\left(\dfrac{\partial I}{\partial x_1}\right)\left(\dfrac{\partial I}{\partial x_2}\right)\right|_{\mathbf{x}+\mathbf{h}_n} & \sum_{\mathbf{x} \in R} \left.\left(\dfrac{\partial I}{\partial x_2}\right)^2\right|_{\mathbf{x}+\mathbf{h}_n} \end{bmatrix} \tag{80}$$

Since G is a symmetric 2×2 matrix, we compute its inverse simply as

$$G^{-1} = \frac{1}{G_{11}G_{22} - G_{12}^2} \begin{bmatrix} G_{22} & -G_{12} \\ -G_{12} & G_{11} \end{bmatrix} \tag{81}$$

and the image motion is then given by

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mathbf{e}G^{-1} \tag{82}$$

The image intensity data $I(\mathbf{x})$ is generally defined only for integer values of $\mathbf{x}$, yet the above equations require calculation of $I(\mathbf{x}+\mathbf{h}_n)$, where $\mathbf{h}_n$ is not generally an integer. We use simple bilinear interpolation to estimate the image intensities at non-integral values.

$$i_1 = \lfloor x_1 \rfloor$$
$$i_2 = \lfloor x_2 \rfloor$$

$$\begin{aligned} I(\mathbf{x}) = {} & (x_1 - i_1)\,(x_2 - i_2)\,I(\begin{bmatrix} i_1 \\ i_2 \end{bmatrix}) + (i_1 + 1 - x_1)\,(x_2 - i_2)\,I(\begin{bmatrix} i_1 + 1 \\ i_2 \end{bmatrix}) + \\ & (x_1 - i_1)\,(i_2 + 1 - x_2)\,I(\begin{bmatrix} i_1 \\ i_2 + 1 \end{bmatrix}) + (i_1 + 1 - x_1)\,(i_2 + 1 - x_2)\,I(\begin{bmatrix} i_1 + 1 \\ i_2 + 1 \end{bmatrix}) \end{aligned} \tag{83}$$

Here the function $\lfloor r \rfloor$ represents the greatest integer of r. The same scheme is used to interpolate derivatives $\partial I / \partial x|_{\mathbf{x}+\mathbf{h}_n}$ at non-integer positions.

3.2. Feature Selection and Image Motion Confidence Estimation

Tomasi and Kanade pointed out in [43] that tile 2×2 matrix

$$G = \sum_{\mathbf{x} \in R} \left(\frac{\partial I}{\partial \mathbf{x}}\right)\left(\frac{\partial I}{\partial \mathbf{x}}\right)^T \tag{84}$$

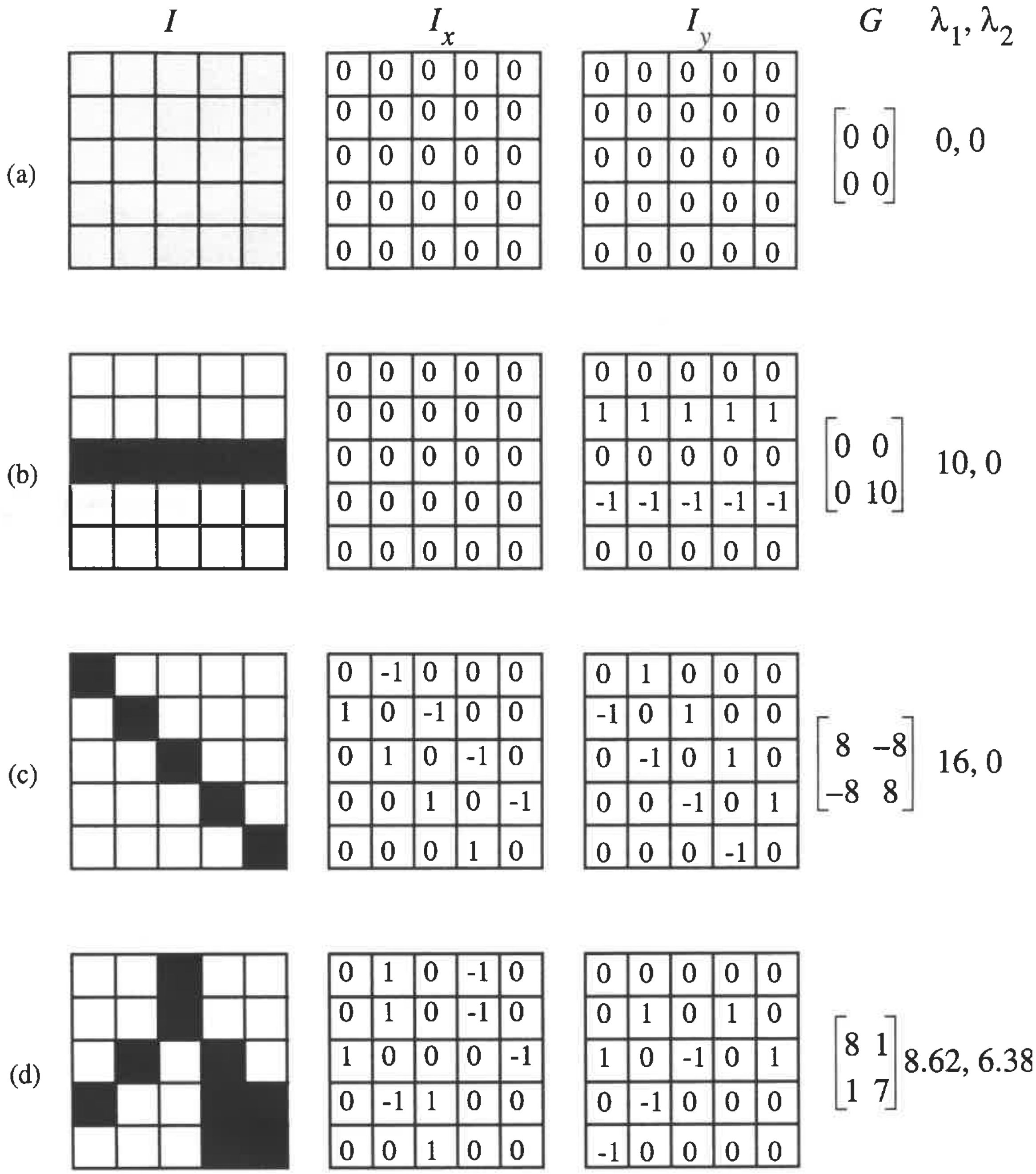
can be used as a measure of a region's "trackability". To determine whether a region of the image constitutes a trackable feature point, they examine the two eigenvalues of G. Figure 14 shows the eigenvalues of G for four sample image regions. Relatively textureless regions, such as the region shown in Figure 14(a), are difficult to track and have very low image derivatives. Therefore their corresponding G matrices have much smaller magnitudes than G matrices calculated over equal-sized region with greater amounts of texture and brightness variation. However, simply examining the magnitudes of the G matrices is not sufficient for determining trackability. Regions containing high spatial derivatives can be line or edge features, which can only be localized in one direction. If the x- and y- gradients are highly correlated, as is the case of oriented features, then the G matrix will have one large eigenvalue and one small eigenvalue, as illustrated in Figure 14(b), (c). When both eigenvalues are large, the region has high spatial gradient in two orthogonal directions, and therefore can be localized in the image.

The criteria of choosing features whose G matrices have two large eigenvalues can also be seen from a purely mathematical point of view. If G has one or more small eigenvalues, then it is nearly singular, so computing its inverse, as required for equation (81), becomes an ill-conditioned problem. When both eigenvalues are large, the matrix is invertible, so equation (81) is well-conditioned.

While the basic feature detection an tracking approach is valid for any type of region R, in our implementation we have restricted R to square windows of a fixed size. In order to choose features, we compute the 2×2 matrix G for all square regions of the given size in the image. We then choose as features those regions with the largest values of the smaller eigenvalue of G, λ_{min}. This generally results in many overlapping features containing largely the same regions, so among overlapping regions we choose only the single region with the highest λ_{min}.

The feature window size must be large enough to include sufficient texture, while small enough that the feature planarity assumption is not drastically violated. Clearly larger windows will require more computation to perform feature detection and tracking. On the images we have typically tested, 11×11, 13×13, and 15×15 feature windows generally work well for both feature detection and tracking.

Once features have been selected, the value λ_{min} can still be used to provide an estimate of the relative confidence with which a particular feature can be tracked. Features with high λ_{min} contain high-frequency texture patterns and can be localized very precisely even in the



	I_x					I_y					G	λ_1, λ_2
(a)	0	0	0	0	0	0	0	0	0	0	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	0, 0
	0	0	0	0	0	0	0	0	0	0		
	0	0	0	0	0	0	0	0	0	0		
	0	0	0	0	0	0	0	0	0	0		
	0	0	0	0	0	0	0	0	0	0		
(b)	0	0	0	0	0	0	0	0	0	0	$\begin{bmatrix} 0 & 0 \\ 0 & 10 \end{bmatrix}$	10, 0
	0	0	0	0	0	1	1	1	1	1		
	0	0	0	0	0	0	0	0	0	0		
	0	0	0	0	0	-1	-1	-1	-1	-1		
	0	0	0	0	0	0	0	0	0	0		
(c)	0	-1	0	0	0	0	1	0	0	0	$\begin{bmatrix} 8 & -8 \\ -8 & 8 \end{bmatrix}$	16, 0
	1	0	-1	0	0	-1	0	1	0	0		
	0	1	0	-1	0	0	-1	0	1	0		
	0	0	1	0	-1	0	0	-1	0	1		
	0	0	0	1	0	0	0	0	-1	0		
(d)	0	1	0	-1	0	0	0	0	0	0	$\begin{bmatrix} 8 & 1 \\ 1 & 7 \end{bmatrix}$	8.62, 6.38
	0	1	0	-1	0	0	1	0	1	0		
	1	0	0	0	-1	1	0	-1	0	1		
	0	-1	1	0	0	0	-1	0	0	0		
	0	0	1	0	0	-1	0	0	0	0		

Figure 14. Feature trackability of example image regions

These simple binary images are used to illustrate the trackability of various image regions. (a) contains no image texture so it would make a poor feature. (b) and (c) have high gradients, but because the x- and y- gradients are highly correlated with each other, they also are not trackable. Only feature (d) can be used as a trackable feature.

presence of image noise. Features with lower values of λ_{min} contain smoother texture, so image noise is more likely to cause shifts in the computed position of the points. We can estimate the confidence in the tracking results of a feature as

$$\gamma_R = \frac{\lambda_{min}}{\log n} \tag{85}$$

where n is the noise intensity in the given region of the image. We estimate the amount of noise using the tracking residue E, which is an indicator of the total level of noise and other deviation from the "translational patch" assumption. Therefore our confidence measure is

$$\gamma_R = \frac{\lambda_{min}}{\log E} \tag{86}$$

Clearly the scaling factor of these confidence values is arbitrary, as they have units of $(intensity)^2/\log(intensity)$. Therefore they should only be used to estimate the confidence of feature measurements relative to each other.

3.3. Tracking Despite Large Image Motion

Typical image sequences may contain large inter-frame image motions due to unsteady or rapid camera motion. Traditional correlation-based tracking techniques using exhaustive search over a large region of the image are computationally expensive, and gradient-based techniques cannot follow large image motions due to local minima in the search space. Hierarchical methods have been studied to address these problems in the optical flow domain [1][5]. A "pyramid" of reduced resolution or spatially smoothed copies of each original image is produced, and flow computation proceeds from the lowest resolution or lowest frequency levels of the pyramid to the highest. Eliminating the image's high-frequency spatial components enables gradient descent methods to avoid local minima in the search for the optimal feature displacement values, and allows them to search over a larger region of the image.

We apply these techniques to the feature tracking domain. Our approach is to first compute a sparse optical flow map using a novel hierarchical method that incorporates image-based confidence measures. We then interpolate this map to determine an initial estimate for the feature tracking stage. The final feature tracking is then performed using the gradient descent technique described in section 3.1.

3.3.1. Hierarchical Confidence-Weighted Sparse Optical Flow Estimation

The iterative tracking method outlined above produces feature translation results accurate to within sub-pixel resolution, provided that the initial flow estimate $\mathbf{h}_0$ transforms the feature to within a few pixels of the correct minimum. (The actual range in which the method will converge to the correct solution depends on the spatial frequency of the image in the tracking region.) However, image sequences taken from unsteady platforms, or sequences with low temporal sampling frequency often contain motions of 50 pixels or more.

Multi-resolution techniques could be applied to this problem by tracking each feature throughout the levels of the image pyramid, using windows centered on our initial features, using the results of tracking this larger version of the feature in one level of the pyramid as a starting value for tracking in the next higher resolution level of the pyramid. However, there is no certainty that features identified based on their trackability in the highest resolution images will still be trackable in low-resolution versions of the same images. This technique would encounter difficulty when feature points lie near the image borders, and would result in redundant computation at the highest levels of the pyramid, where all features would largely overlap.

We use a multi-resolution method to compute a sparse optical flow map between pairs of consecutive images, which we then interpolate to initially estimate each feature's translation. Unfortunately, a major drawback of optical flow is that many regions of typical images do not contain sufficient texture to produce reliable optical flow estimates. Optical flow researchers have developed a variety of methods for post-smoothing flow results. However, all of these methods are computationally expensive, some require very dense optical flow computation, and some smooth optical flow results even in regions where sufficient texture information to compute the flow is available, thereby "smoothing over" already accurate flow results [20][21][27][34]. Extremely accurate flow results are not required for our purposes, since our flow field will simply be interpolated to determine initial values for the feature tracking, so instead we propose an inexpensive method similar to the method put forth by Nagel [27].

Rather than post-smoothing the optical flow estimates, we incorporate confidence-weighted smoothing between the levels of the pyramid into the optical flow computation itself. We perform the basic optical flow computation using the Lucas-Kanade method described in section 3.1., expressing our confidence in each flow measurement using the γ measure described in section 3.2. However, at each iteration of the flow computation, the effective flow estimate is computed by combining the current flow estimate and the flow estimate from the previous level, weighting each by their respective confidences. The resulting flow will take advantage of the texture in the current image whenever possible, and inherit the flow of the larger region from the previous level of the pyramid when insufficient texture exists in the current image. Since there is no expensive post-smoothing step, and no reliance on dense flow estimates, this method requires very little additional computation.

To compute the flow at some position (x, y) in pyramid level l, we interpolate from the flow estimates computed at pyramid level $l-1$. The optical flow estimates are sparse, and have only been computed at certain image positions. Let (x_a, y_a), (x_a, y_b), (x_b, y_a), and (x_b, y_b), be the four points surrounding (x, y) at which flow estimates have been computed. These four points form a rectangle which (x, y) lies within, as shown in Figure 15. Let the flow estimates for these four points be $\mathbf{u}_{aa}$, $\mathbf{u}_{ab}$, $\mathbf{u}_{ba}$, and $\mathbf{u}_{bb}$, and their associated confi-

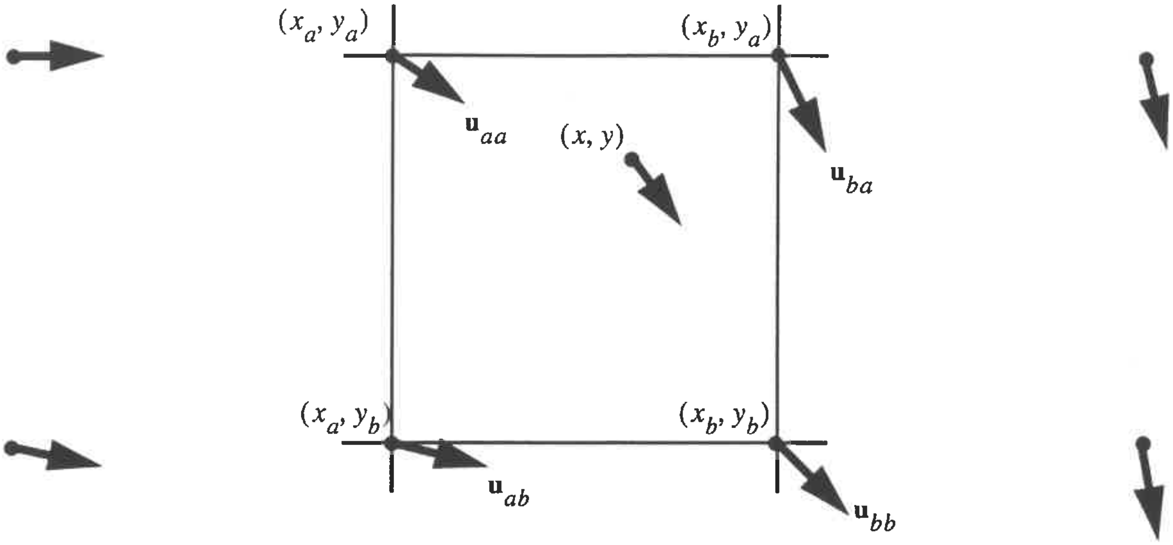


Figure 15. Interpolation of Sparse Optical Flow

The flow at the point (x, y) is interpolated from the computed flow values at the four nearest points where flow has been computed, (x_a, y_a), (x_a, y_b), (x_b, y_a), and (x_b, y_b).

dences be γ_{aa}, γ_{ab}, γ_{ba}, and γ_{bb}, respectively. We interpolate between these four measurements to estimate the flow and confidence for (x, y) from level $l-1$. We first adjust the confidences to account for the interpolation, by defining

$$\begin{aligned} \gamma'_{aa} &= (x_b - x)\,(y_b - y)\,\gamma_{aa} \\ \gamma'_{ab} &= (x_b - x)\,(y - y_a)\,\gamma_{ab} \\ \gamma'_{ba} &= (x - x_a)\,(y_b - y)\,\gamma_{ba} \\ \gamma'_{bb} &= (x - x_a)\,(y - y_a)\,\gamma_{bb} \end{aligned} \tag{87}$$

These four measurements with associated confidences are combined in the manner suggested by the information fusion theorem [1] to compute the optimal flow estimate and its associated confidence value.

$$\begin{aligned} \gamma_{l-1}(x, y) &= [\gamma'_{aa} + \gamma'_{ab} + \gamma'_{ba} + \gamma'_{bb}] \\ \mathbf{u}_{l-1}(x, y) &= [\gamma_{l-1}(x, y)]^{-1}\,[\gamma'_{aa}\mathbf{u}_{aa} + \gamma'_{ab}\mathbf{u}_{ab} + \gamma'_{ba}\mathbf{u}_{ba} + \gamma'_{bb}\mathbf{u}_{bb}] \end{aligned} \tag{88}$$

We now modify the standard image motion computation described in section 3.1 to use $\mathbf{u}_{l-1}(x, y)$ and $\gamma_{l-1}(x, y)$ in the computation of the image motion in level l. We use $\mathbf{u}_{l-1}(x, y)$ as the initial value $\mathbf{h}_0$ for iteratively computing the flow at level l. At each iteration, we compute a flow estimate $\mathbf{h}_n$ using equation (82) and compute the corresponding confidence γ_R of the estimate using equation (86). We weight this estimate with the estimate from the previous level of the pyramid to obtain the actual flow estimate for iteration n.

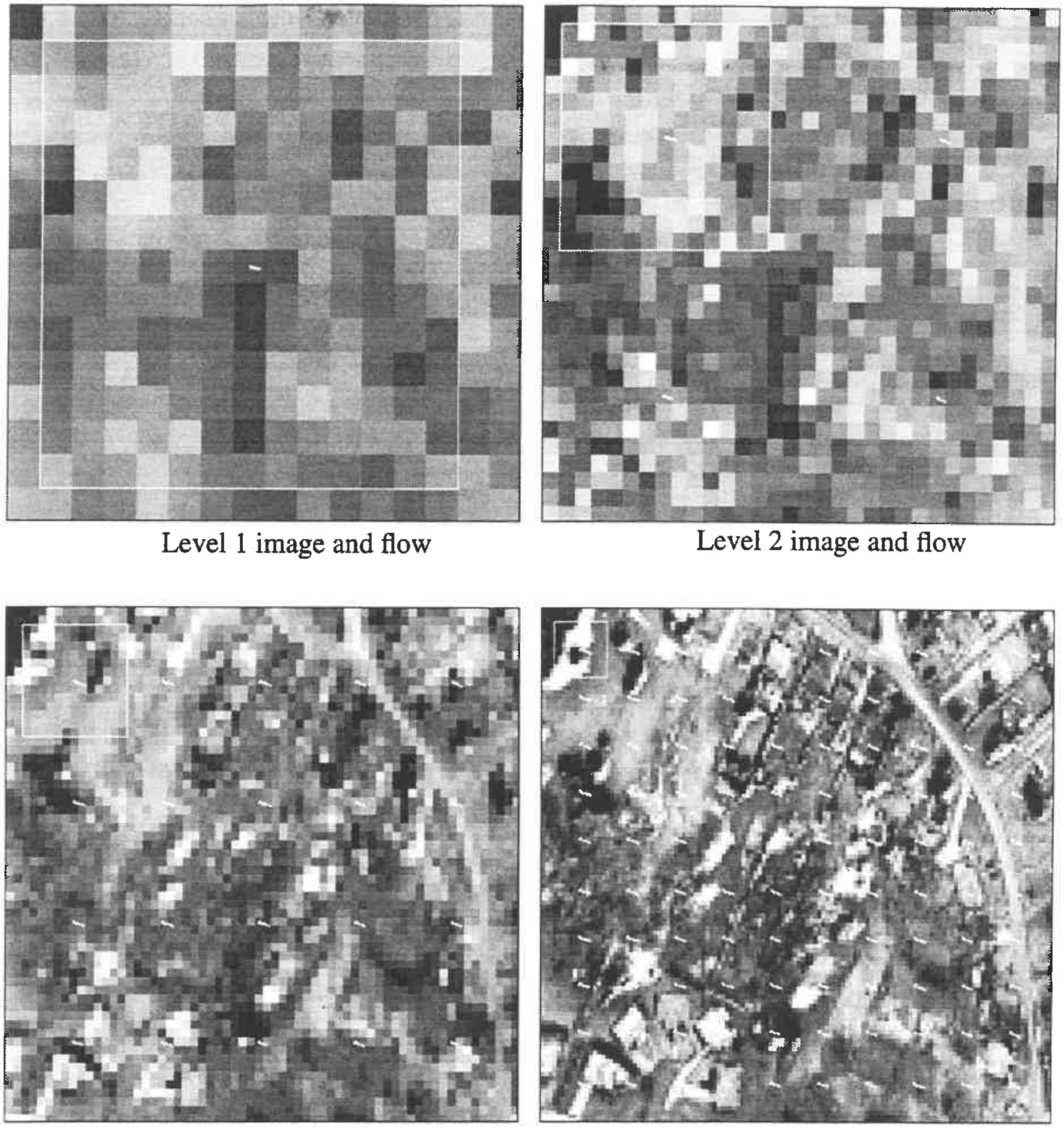
$$\begin{aligned} \gamma_{l_n}(x, y) &= (1-k)\gamma + k\gamma_{l-1}(x, y) \\ \mathbf{u}_{l_n}(x, y) &= \gamma_{l_n}^{-1}[(1-k)(\gamma_R \mathbf{h}_n) + k\gamma_{l-1}(x, y) h_{l-1}(x, y)] \end{aligned} \quad (89)$$

Here k is a constant weighting factor between 0 and 1 that determines how much confidence to attach to estimates from lower levels of the pyramid. A weighting factor $k = 1$ will cause all flow estimates to be strictly inherited from the previous level without regard to the image or flow estimates computed at the current level. Using $k = 0$ causes each level's computation to use the previous level's flow estimate only as an initial value, but it is does not influence the flow computation beyond the initialization.

3.3.2. Sparse Optical Flow Results

Our initial hierarchical implementation did not incorporate any confidence-based weighting from one level to the next. Rather, each level of the pyramid simply interpolated the flow estimates from the previous level and used that flow estimate as its initial motion estimate $\mathbf{h}_0$, equivalent to the $k = 0$ case in the above formulation. This system successfully tracked feature motions over pixel motions as large as 100 pixels, and worked well for images containing sufficient texture information at all points in the image, such as the aerial image sequence shown in Figure 16. However, in image sequences containing large, homogeneous, textureless regions, the flow results were very poor. Therefore the initial feature position estimates of any features near those regions were incorrect, and the feature tracker was unable to find the correct minimum. The flow computation of this method on a pair of images from one such sequence is shown in Figure 17. Note that the flow estimates in the highly textured parts of the image, such as the stovetop, the sink, the calendar, and the upper portion of the refrigerator, are correct.

Figure 18 shows the results on the same pair of images using an inter-level weighting constant of $k = 0.5$, though we observed roughly equivalent behavior with k ranging from 0.3 to 0.7. The flow results were substantially improved in portions of the image which lacked significant image texture, enabling the correct tracking results shown in Figure 19. Notice that in many cases the correct position of a feature in the second image did not overlap the feature's position in the first image at all, so clearly a non-hierarchical method which uses initial position estimates based on the previous frame would fail. Figure 20 shows the tracking results for the aerial sequences, demonstrating that the tracker successfully tracked feature motions of over 30 pixels.



Level 1 image and flow

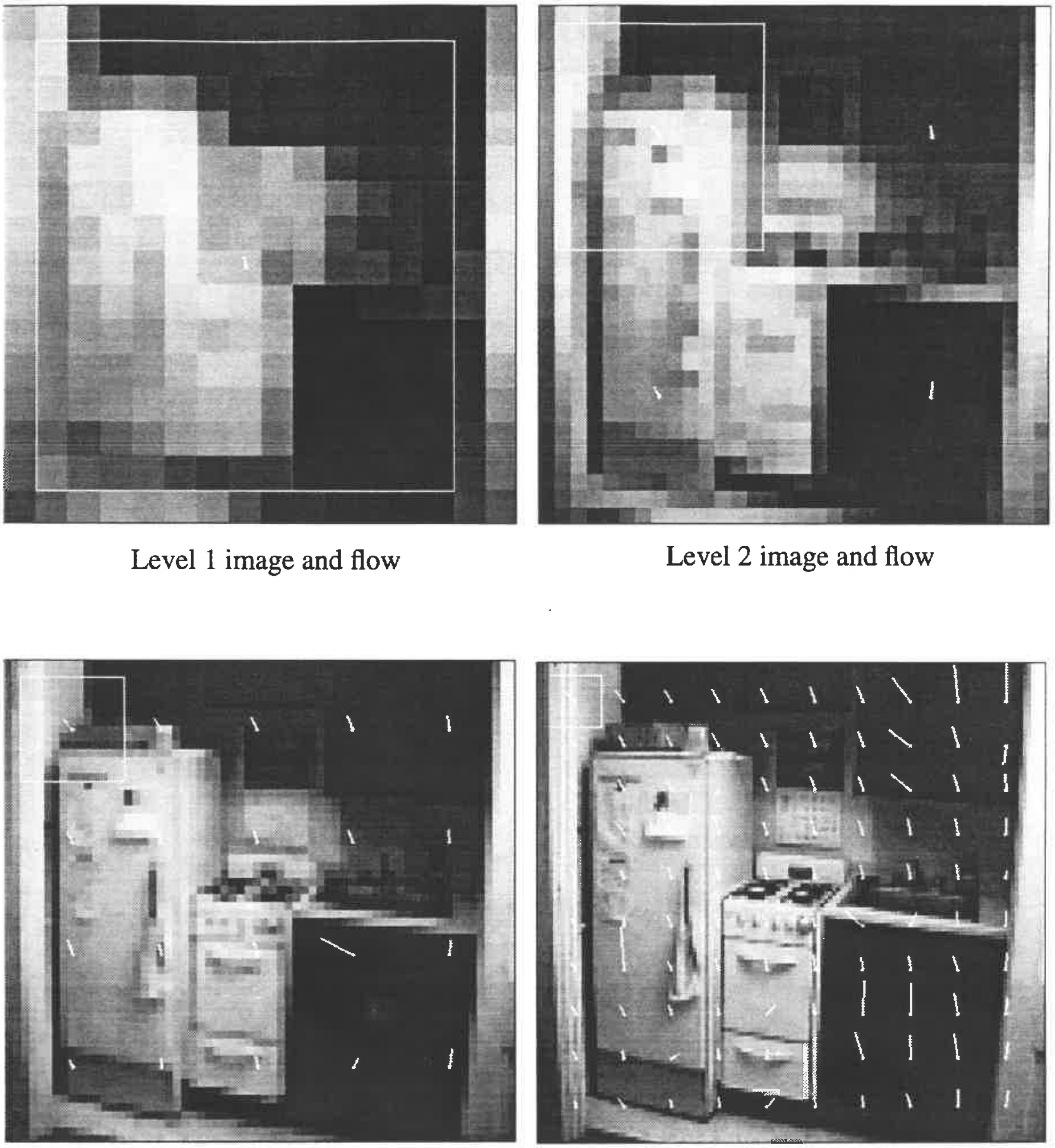
Level 2 image and flow

Level 3 image and flow

Level 4 image and flow

Figure 16. Unsmoothed optical flow for Aerial Sequence

In this sequence, there was sufficient texture throughout the image, so even without the addition of confidence-based smoothing, the sparse optical flow map could be computed reliably. The box in the upper-left corner of each image indicates the size of the regions used for flow computation. Each point at which a flow value has been computed is indicated with a small white circle, while the white line extending from each point indicates the direction and magnitude of the optical flow at that point.



Level 1 image and flow

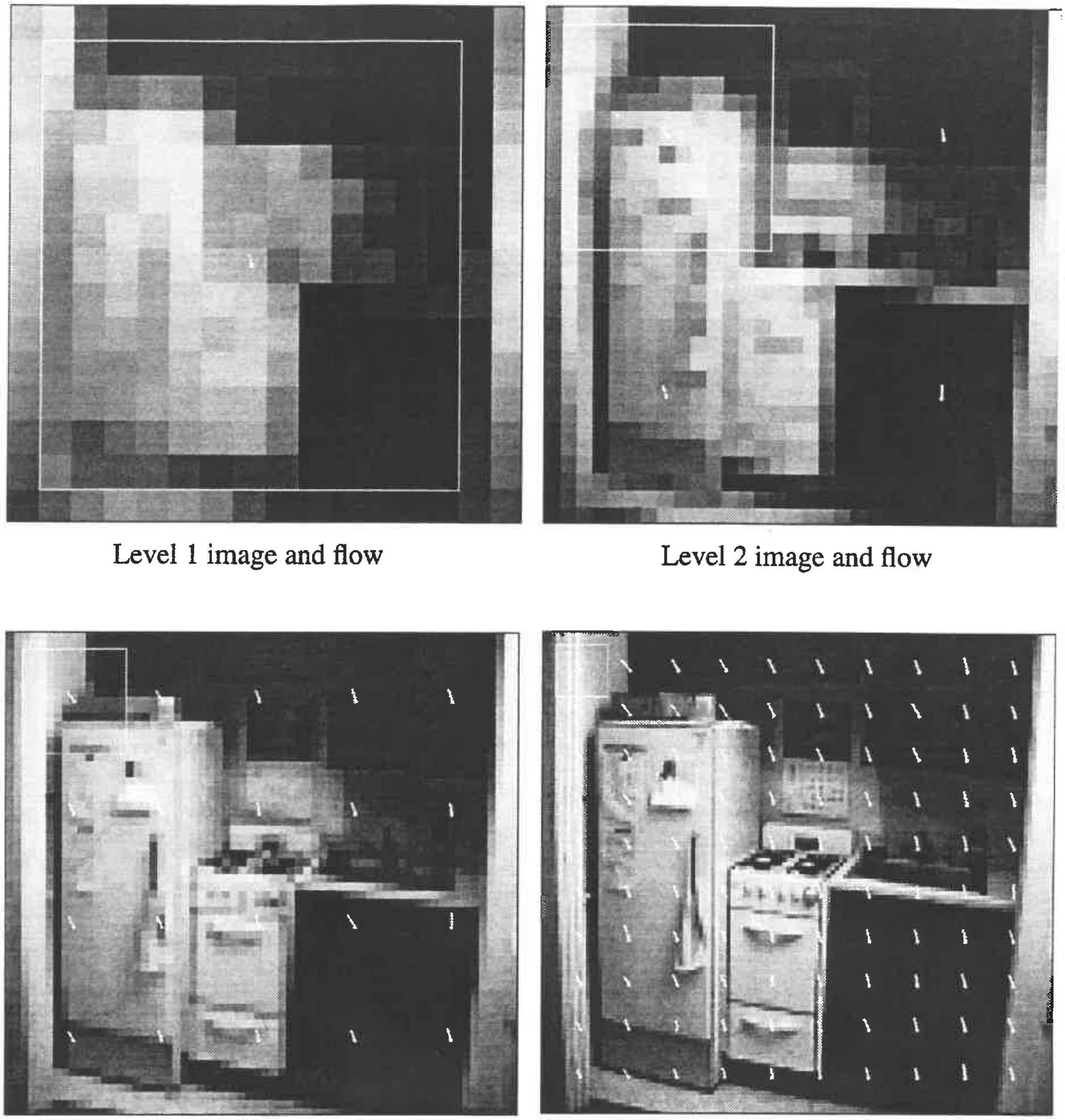
Level 2 image and flow

Level 3 image and flow

Level 4 image and flow

Figure 17. Unsmoothed optical flow - Kitchen Sequence

While the flow results are correct in highly textured regions of the image, the flow results are very poor in regions containing insufficient texture. The box in the upper-left corner of each image indicates the size of the region used for flow computation.



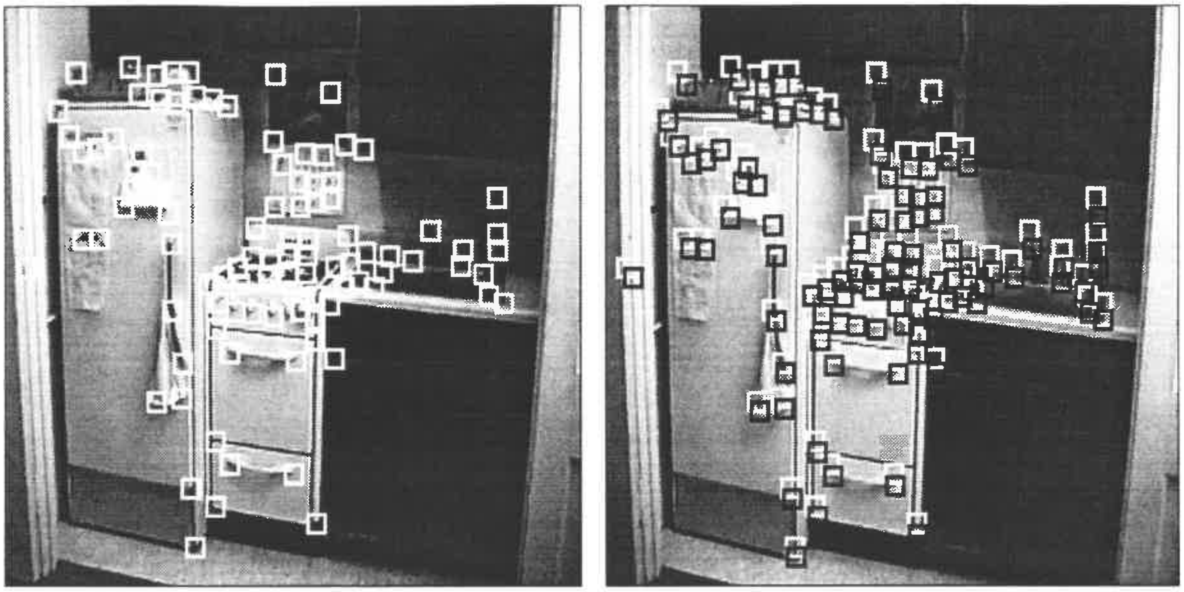
Level 1 image and flow

Level 2 image and flow

Level 3 image and flow

Level 4 image and flow

Figure 18. Optical flow using inter-level confidence-based smoothing
In regions containing little or no image texture, the flow at the high-resolution level of the image pyramid is primarily inherited from the flow in the previous level of the pyramid, yielding better flow estimates in those regions.

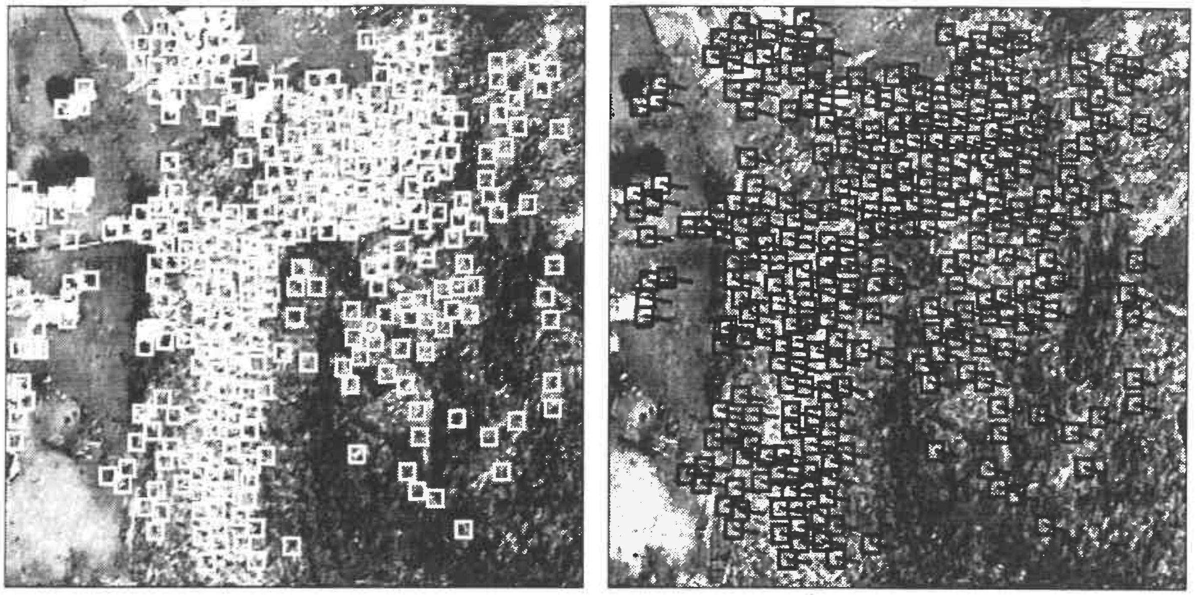


(a) Image 1

(b) Image 2

Figure 19. Tracking Results for Kitchen Sequence

(a) The features for this sequence were automatically chosen from the first image according to their λ_{min} values. (b) The tracked position of each feature in the second image is shown in black, while white squares mark the same coordinates at which each feature was detected in the first frame.



frame 72

frame 73

Figure 20. Tracking Results for Aerial Image Sequence

This pair of images was chosen for the large inter-frame displacement. The right image shows the feature positions in the 73rd image of the sequence, with lines pointing to the positions in the previous image. Many of the features are displaced by 30 or more pixels.

3.4. Factorization Despite Occluded and Uncertain Tracking Data

So far, our formulation of the factorization method has assumed that all of the entries of the measurement matrix are known and are equally reliable. In real image sequences, this is not generally the case. Some feature points are not tracked throughout the entire sequence because they leave the field of view, become occluded, or change their appearance significantly enough that the feature tracker can no longer track them. As the object moves, new feature points can be detected on parts of the object which were not visible in the initial image. Thus the measurement matrix W is not entirely filled; there are some pairs (f, p) for which (u_{fp}, v_{fp}) was not observed.

Not all observed position measurements are known with the same confidence. Some feature windows contain sharp, trackable features, enabling exact localization, while others may have significantly less texture information. Some matchings are very exact, while others are less exact due to unmodeled change in the appearance of a feature. Previously, using some arbitrary criteria, each measurement was either accepted or rejected as too unreliable, and then all accepted observations were treated equally throughout the method.

We address both of these issues by assigning a confidence value to each measurement using equation (86), and modify the shape and motion recovery to weight each measurement by its corresponding confidence value. If a feature is not observed in some frames, the confidence values for the measurements corresponding to those frames are set to zero.

3.4.1. Confidence-Weighted Formulation

We can view the decomposition step of Section 2.3.4 as a way, given the measurement matrix W, to compute an $\hat{M}$, $\hat{S}$, and T that satisfy the equation

$$W = \hat{M}\hat{S} + T\begin{bmatrix}1 & \dots & 1\end{bmatrix} \tag{90}$$

There, our first step was to compute T directly from W. We then used the SVD to factor the matrix $W^* = W - T\begin{bmatrix}1 & \dots & 1\end{bmatrix}$ into the product of $\hat{M}$ and $\hat{S}$. In fact, using the SVD to perform this factorization produced an $\hat{M}$ and $\hat{S}$ that, given W and the T just computed from W, minimized the error

$$\varepsilon_0 = \sum_{r=1}^{2F} \sum_{p=1}^{P} \left(W_{rp} - \left(\hat{M}_{r1}\hat{S}_{1p} + \hat{M}_{r2}\hat{S}_{2p} + \hat{M}_{r3}\hat{S}_{3p} + T_r \right) \right)^2 \tag{91}$$

In our new confidence-weighted formulation, we associate each element of the measurement matrix W_{rp} with a confidence value γ_{rp}. We incorporate these confidences into the factorization method by reformulating the decomposition step as a standard weighted least squares problem where each measurement W_{rp} has a standard deviation inversely proportional to γ_{rp}. We seek the $\hat{M}$, $\hat{S}$, and T which minimize the error

$$\varepsilon = \sum_{r=1}^{2F} \sum_{p=1}^{P} \left(\frac{W_{rp} - \left(\hat{M}_{r1}\hat{S}_{1p} + \hat{M}_{r2}\hat{S}_{2p} + \hat{M}_{r3}\hat{S}_{3p} + T_r \right)}{1/\gamma_{rp}} \right)^2 \qquad (92)$$

Note that the scale of the γ_{rp} is arbitrary, as is the constant of proportionality relating these confidence values to the standard deviations of the feature measurements. However, this effects ε only by a constant factor, and will not effect the values of the final $\hat{M}$, $\hat{S}$, and T chosen to minimize ε. Once we have solved this minimization problem for $\hat{M}$, $\hat{S}$, and T, we will proceed with the normalization step and the rest of the shape and motion recovery process precisely as before.

3.4.2. Iterative Solution Method

There is sufficient mathematical constraint to compute $\hat{M}$, $\hat{S}$, and T, provided the number of known elements of W, *i.e.* elements of W for which $\gamma_{fp} > 0$, exceeds the total number of variables $(8F + 3P)$. The minimization of equation (91) is a nonlinear least squares problem because each term contains products of the variables $\hat{M}_{rj}$ and $\hat{S}_{jp}$. However, it is separable; the set of variables can be partitioned into two subsets, the motion variables and the shape variables, so that the problem becomes a simple linear least squares problem with respect to one subset when the other is known. Unfortunately there does not appear to be any extension of the SVD method to address the problem of weighted decomposition. Instead, we use a variant of an algorithm suggested by Ruhe and Wedin [32] for solving such problems - one that is equivalent to alternately refining the two sets of variables. In other words, we hold $\hat{S}$ fixed at some value and solve for the $\hat{M}$ and T which minimize ε. Then holding $\hat{M}$ and T fixed, we solve for the $\hat{S}$ which minimizes ε. Each step of the iteration is simple and fast since, as we will show shortly, it is a series of linear least squares problems. We repeat the process until a step of the iteration produces no significant reduction in the error ε.

To compute $\hat{M}$ and T for a given $\hat{S}$, we first rewrite the minimization of equation (91) as

$$\varepsilon = \sum_{r=1}^{2F} \varepsilon_r, \quad \text{where} \quad \varepsilon_r = \sum_{p=1}^{P} \gamma^2_{rp}\left(W_{rp} - \left(\hat{M}_{r1}\hat{S}_{1p} + \hat{M}_{r2}\hat{S}_{2p} + \hat{M}_{r3}\hat{S}_{3p} + T_r \right)\right)^2. \qquad (93)$$

For a fixed $\hat{S}$, the total error ε can be minimized by independently minimizing each error ε_r, since no motion variable appears in more than one ε_r equation. Each ε_r describes a weighted linear least squares problem in the variables $\hat{M}_{r1}$, $\hat{M}_{r2}$, $\hat{M}_{r3}$, and T_r. For every row r, the four variables are computed by finding the least squares solution to the overconstrained linear system of 4 variables and P equations

$$\begin{bmatrix} \hat{S}_{11}\gamma_{r1} & \hat{S}_{21}\gamma_{r1} & \hat{S}_{31}\gamma_{r1} & \gamma_{r1} \\ \dots & \dots & \dots & \dots \\ \hat{S}_{1P}\gamma_{rP} & \hat{S}_{2P}\gamma_{rP} & \hat{S}_{3P}\gamma_{rP} & \gamma_{rP} \end{bmatrix} \begin{bmatrix} \hat{M}_{r1} \\ \hat{M}_{r2} \\ \hat{M}_{r3} \\ T_r \end{bmatrix} = \begin{bmatrix} W_{r1}\gamma_{r1} \\ \dots \\ W_{rP}\gamma_{rP} \end{bmatrix} \qquad (94)$$

Many of the γ_{rp} may be zero, so we only include those rows of the above equation which are not zero.

Similarly, for a fixed $\hat{M}$ and T, each p^{th} column of $\hat{S}$ can be computed independently of the other columns, by finding the least squares solution to the linear system of 3 variables and $2F$ equations

$$\begin{bmatrix} \hat{M}_{11}\gamma_{1p} & \hat{M}_{12}\gamma_{1p} & \hat{M}_{13}\gamma_{1p} \\ \dots & \dots & \dots \\ \hat{M}_{2F,1}\gamma_{2F,p} & \hat{M}_{2F,2}\gamma_{2F,p} & \hat{M}_{2F,3}\gamma_{2F,p} \end{bmatrix} \begin{bmatrix} \hat{S}_{1p} \\ \hat{S}_{2p} \\ \hat{S}_{3p} \end{bmatrix} = \begin{bmatrix} (W_{1p} - T_1)\gamma_{1p} \\ \dots \\ (W_{2F,p} - T_{2F})\gamma_{2F,p} \end{bmatrix} \qquad (95)$$

As in any iterative method, an initial value is needed to begin the process. Our initial experiments showed, however, that when the confidence matrix contained few zero values, the method consistently converged to the correct solution in a small number of iterations, even beginning with a random initial value. For example, in the special case of $\gamma_{rp} = 1.0$ (which corresponds to the case in which all features are tracked throughout the entire sequence and are known with equal confidence), the method always converged in 5 or fewer iterations, requiring even less time than computing the singular value decomposition of W; when the γ_{rp} were randomly given values ranging from 1 to 10, the method generally converged in 10 or fewer iterations; and with a γ whose fill fraction (fraction of non-zero entries) was 0.8, the method converged in 20 or fewer iterations. However, when the fill fraction decreased to 0.6, the method occasionally failed to converge even after 100 iterations.

In order to apply the method to sequences with lower fill fractions, it is critical that we obtain a reasonable initial value before proceeding with the iteration. We developed an approach analogous to the propagation method described in [43]. We first find some subset of the rows and some subset of the columns of W for which all of the confidences of measurements belonging to both subsets are non-zero. A simple choice would be to select all of the features which are visible in the first frame, and all of the frames from the first frame to the first frame in which one of those features disappears. These subsets could be enlarged by a simple greedy algorithm in which points with short feature tracks are discarded, enabling more features to be included in the subset, until the total number of measurements included in the subset reaches a maximum. The method we actually use improves over this approach slightly. We first choosing the point which was seen in the largest number of frames and place it in the subset. We then consider other points to add in descending order of the total number of frames in which they were observed, and use the simple greedy heuristic described above. In practice, it seems to work well for realistic fill patterns.

Once we have chosen our initial subset, we solve for the corresponding rows of $\hat{M}$ and T and columns of $\hat{S}$ for the subset by running the iterative method starting with a random initial value. As indicated above, since this subset has a fill fraction of 1, this converges quickly, producing estimated values for this subset of $\hat{M}$, $\hat{S}$, and T. We can solve for one additional row of $\hat{M}$ and T by solving the linear least squares problem of equation (94) using only the known columns of $\hat{S}$. We can solve for one additional column of $\hat{S}$ by solving the linear least squares problem of equation (95) using only the known rows of $\hat{M}$ and T. We continue solving for additional rows of $\hat{M}$ and T or columns of $\hat{S}$ until $\hat{M}$, $\hat{S}$, and T are completely known. Using this as the initial value allows the iterative method to converge in far fewer iterations.

3.4.3. Analysis of Weighted Factorization using Synthetic Data

We tested the confidence-weighted paraperspective factorization method with artificially generated measurement matrices and confidence matrices whose fill fraction (fraction of non-zero entries) was varied from 1.0 down to 0.2. Creating a confidence matrix that has a given fill fraction and a fairly realistic fill pattern (the arrangement of zero and non-zero confidence values) is a non-trivial task. First, a surface was manually created to connect the three dimensional points of the synthetic object. This allowed us to automatically determine at what point in a motion sequence any given point would become occluded, and when it would become visible again, producing a synthetic fill pattern. These feature tracks are then extended or shortened until the desired fill fraction is achieved. In the case of lengthening the feature tracks, this is not a strictly realistic fill pattern since it is assuming that a point remains visible for a few frames after some occluding surface has passed in front of it. However, it was an effective way to test the shape reconstruction using the same object for a variety of fill fractions.

Figure 21 shows how the performance degrades as the noise level increases and fill fraction decreases. The synthetic sequences were of a single object consisting of 99 points, initially located 60 times the object size from the camera. Each data point represents the average solution error over 5 runs, using a different seed for the random noise. The motion, method of generating the sequences, and error measures are as described in Section 2.5.

Errors in the recovered motion only increase slightly as the fill fraction decreases from 1.0 to 0.5. At a very low noise level, such as 0.1 pixels, this behavior continues down to a fill fraction of 0.3. When the fill fraction is decreased below this range, however, the error in the recovered motion increases very sharply. The shape results appear more sensitive to decreased fill fractions; as the fill fraction drops to 0.8 or lower, the shape error increases sharply, and then increases dramatically when the fill fraction reaches 0.6 or 0.5. While the system of equations defined by equation (91) is still overconstrained at these lower fill fractions, apparently there is insufficient redundancy to overcome the effects of the noise in the data.

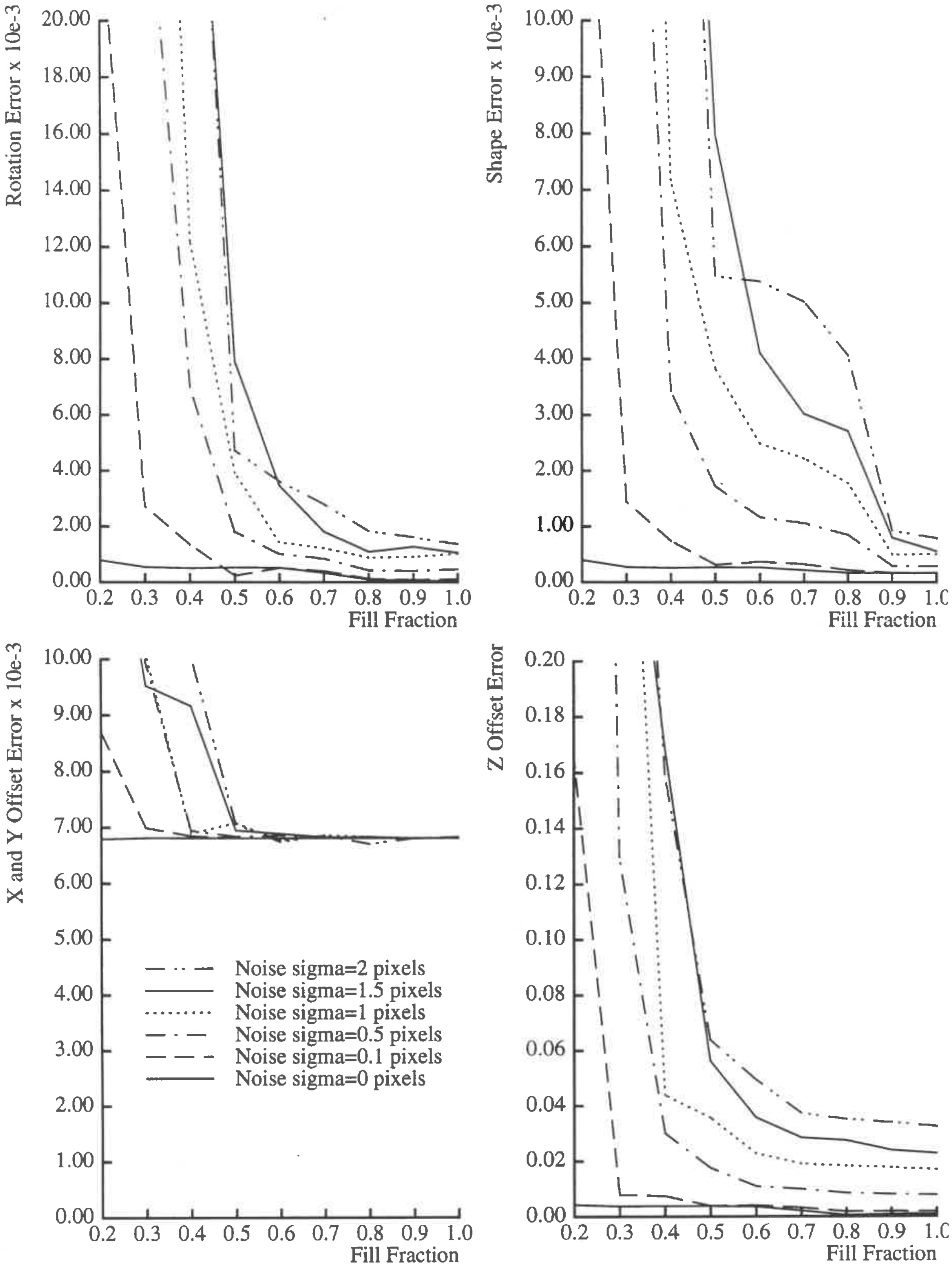


Figure 21. Confidence-Weighted Shape and Motion Recovery by Noise Level, Fill Fraction

The methods were implemented in C using the *Numerical Recipes in C* routines for matrix operations and numerical methods. The time required to solve the system of 60 frames and 99 points depends on the number of iterations required for convergence, which seemed to vary significantly even for scenarios with the same parameters. Typical numbers of iterations and running times on a Sparc 5/85 workstation are summarized in the following table.

Table 3: Running Time of Confidence-Weighted Factorization

Fill Fraction	Number of Iterations	Total Solution Time
0.7-1.0	4-7	5-7.5 seconds
0.6	5-11	6-12
0.5	10-25	9-14
0.4	20-50	16-24
0.3	50-100	38-45
0.2	100+	38-40[a]

[a] For the case of a 0.2 fill fraction, each iteration requires examining on average 33% fewer measurements than for the 0.3 fill fraction case, so its typical solution time is the same or lower even when it executes more iterations.

3.4.4. Discussion

The weighted factorization method is able to handle moderate amounts of occlusion quite effectively and account for varying feature confidences at nearly the same computational cost as the unweighted version of the factorization method. It's behavior degrades gracefully up to a certain point at which it tends to fail catastrophically, even though theoretically in those situations the problem is still overconstrained. For problems with low fill fractions, the initial value is determined by first solving a very small system and then extending the solution one row or column at a time while keeping estimates for all previous rows or columns fixed. Small errors at any step of the process can have large effects on the values subsequently. Additionally the refinement method is limited to refining the affine shape and motion separately. In some cases, the error could be reduced significantly by rotating a portion of the shape and motion by some rotation matrix so that it better aligns with other parts of the solution. However, the shape and motion cannot be updated simultaneously, and rotating only the shape or only the motion is likely to cause large increases in the error rather than reductions. Therefore the method must proceed in tiny steps, rotating the motion by a small amount, subsequently rotating the shape by that amount, and so on. This explains the observed behavior of the method at low fill fractions; rather than clearly converge, reduction of the error slows drastically, improving by tiny amounts until the limit of 100 iterations is reached. The limitation of this method is nearly the same as that of the separate refinement method - inability to refine shape and motion simultaneously.

While we have detailed our experimental results as functions of the fill fraction, the errors in fact seem to depend on more than simply the fill fraction. Smaller sized problems than the ones used here tend to experience failure at higher fill fractions, while for larger sized problems the methods seem to work even at lower fill fractions (see next section). This may be attributable to the fact that a given fill fraction indicates less redundancy in the information for a smaller problem than for a larger problem. The performance may be more closely related to the *number* of non-zero confidences per row or column, or to some measure which also accounts for the overall length of the sequence or total number of points in addition to the fill fraction. Even for problems of the same size and fill fraction, the performance of the method seems to vary according depending on the particular fill pattern of observed and unobserved measurements.

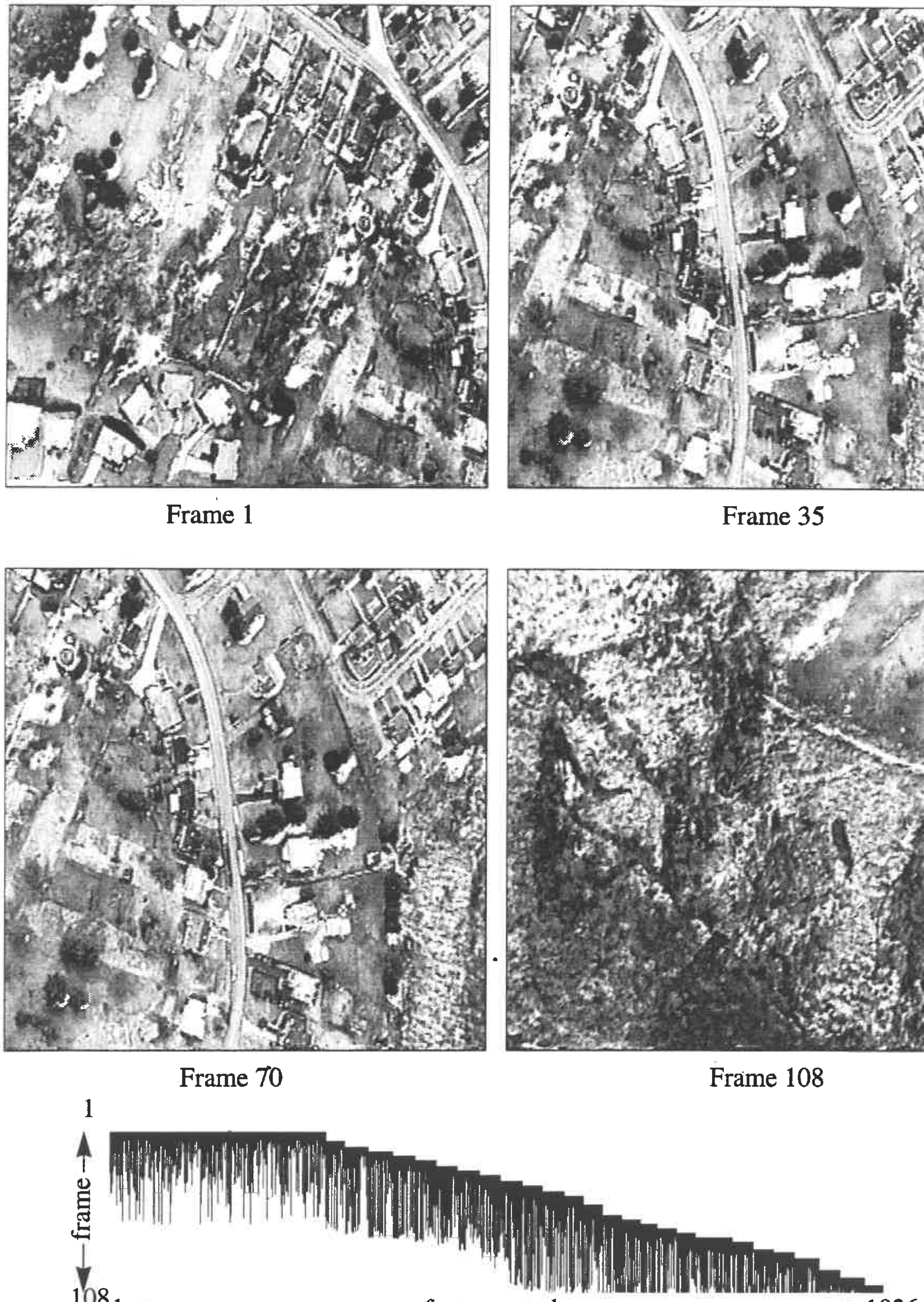
3.5. Application to Aerial Image Sequence

An aerial image sequence was taken from a small airplane overflying a suburban Pittsburgh residential area adjacent to a steep, snowy valley, using a small hand-held video camera. The plane altered its altitude during the sequence and also varied its roll, pitch, and yaw slightly. Several images from the sequence are shown in Figure 22.

Due to the bumpy motion of the plane and the instability of the hand-held camera, features often moved by as much as 30 pixels from one image to the next, but the multi-resolution feature tracker was able to accurately track these motions accurately. The sequence covered a long sweep of terrain, so none of the features were visible throughout the entire sequence. As some features left the field of view, new features were automatically detected and added to the set of features being tracked. A vertical bar in the fill pattern (shown in Figure 22) indicates the range of frames through which a feature was successfully tracked. Each observed data measurement was assigned a confidence value based on the gradient of the feature and the tracking residue. A total of 1026 points were tracked in the 108 image sequence with each point being visible for an average of 30 frames of the sequence, for an overall fill fraction of 28%.

Feature detection required 22 seconds to detect the initial 300 features, and tracking between frames required about 7 seconds per frame on a Sparc 5/85 workstation. New features were detected 26 times during the course of the sequence, whenever the number of visible features dropped below 250. These feature detections took only 12-16 seconds, since much of the image was "blocked out" by the existing features. The total time spent in the feature detection and tracking stage was 1068 seconds, or just under 18 minutes.

The confidence-weighted paraperspective factorization method was used to recover the shape of the terrain and the motion of the airplane. An initial block was chosen in which 36 points were seen in the same 57 frames. Solving this 144×36 system converged in only 7 iterations. The solution was extended one row or column at a time and then all variables were iteratively refined. The entire solution required an additional 48 iterations to converge



Frame 1

Frame 35

Frame 70

Frame 108

Fill pattern indicating points visible in each frame

Figure 22. Aerial Image Sequence

to a solution. Total computation to compute the shape and motion of the 1026 point object over 108 frames was 460 seconds, or 7 minutes and 40 seconds. Two views of the reconstructed terrain map are shown in Figure 23. While no ground-truth was available for the shape or the motion, we observed that the terrain was qualitatively correct, capturing the flat residential area and the steep hillside as well, and that the recovered positions of features on buildings were elevated from the surrounding terrain.

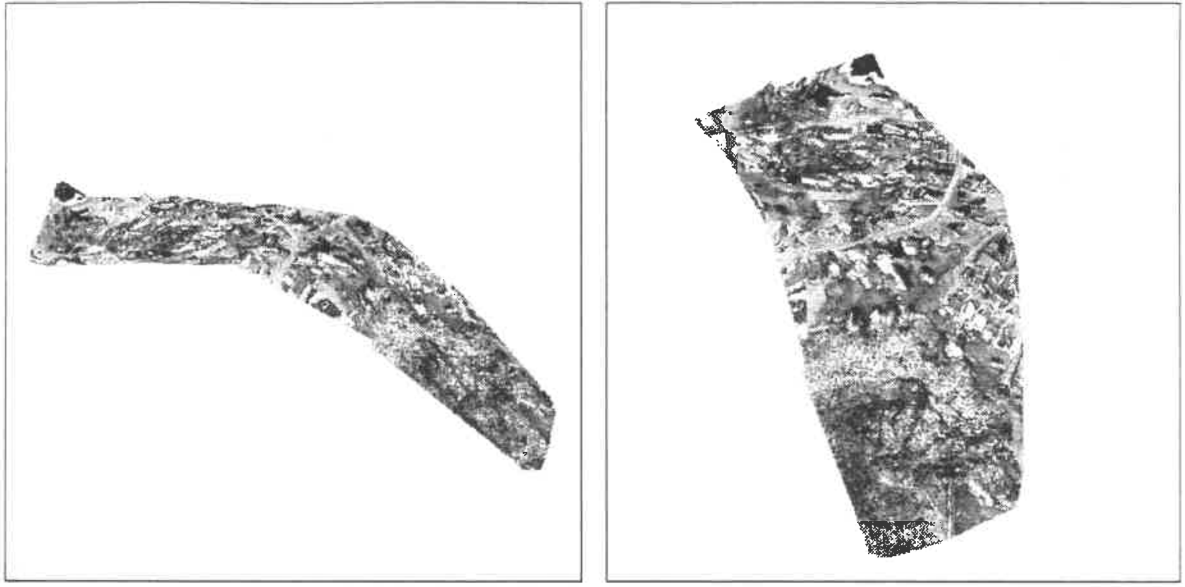


Figure 23. Reconstructed Terrain from Two Viewpoints

4. Projective Factorization

The results of the paraperspective factorization method have been shown to be quite good up to fairly close distances, and can be further refined to account for perspective foreshortening effects with a separate iterative refinement step. However, for images containing severe foreshortening, such as a rigid scene with both near and distant points, the separate iterative refinement step proceeds very slowly. Worse yet, if noise levels are high and foreshortening effects severe, the normalization step of the paraperspective factorization method may fail, providing no initial value for perspective refinement. Clearly a method is needed which models perspective distortion more directly.

This chapter details the projective factorization method, which applies the basic approach of the factorization method to a perspective camera model. In the decomposition step, image measurements are decomposed into a projective shape and a projective motion, rather than an affine shape and motion as in the previous factorization methods. The non-linearity of the perspective projection step makes the use of SVD or bilinear methods impossible for this decomposition step, so general non-linear least-squares techniques are used. Following decomposition, normalization constraints are subsequently applied to the projective motion to convert the projective solution to a Euclidean one. The method is able to compute the correct shape and motion by refining shape and motion parameters simultaneously, unlike the separate perspective refinement method of section 2.4.

The projective decomposition approach is quite similar to the methods Ponce, Marimont, and Cass [30], Szeliski and Kang [38], and Hartley [18] use for recovering projective shape and motion. Our normalization constraints which convert the projective solution into a Euclidean one is novel, and requires only linear operations. Faugeras, Luong, and Maybank [15] and Hartley [18] have put forth methods for converting epipolar geometries or projective motions into Euclidean motions, but their approaches require continuation or other complex methods.

We also describe a technique for recovering Euclidean shape and motion without first performing projective decomposition, similar to the method of Szeliski and Kang [38]. By constraining the shape and motion to follow a Euclidean form, the method can potentially achieve greater accuracy than projective factorization. However, we show that when foreshortening effects are significant, this method fails to converge to the correct minimum when a simplistic initial value is used. Its usefulness therefore is limited to relatively distant objects, or to refining the results produced by other methods.

Finally we show that the non-linear minimization framework developed for projective factorization and Euclidean optimization can be extended to incorporate image data directly rather than relying solely on tracked features. The shape from motion problem is reformulated in terms of minimizing differences in image intensity rather than minimizing errors between predicted and tracked feature point positions. While the method could in theory obviate the need for feature tracking altogether, in practice it is primarily useful for refining results in which some of the initially tracked features were poorly tracked, thus allowing a more dense shape recovery.

4.1. Projective Factorization

The orthographic, scaled orthographic, and paraperspective projection models all can be modeled by bilinear equations. This enables them to be solved by first separating the shape and motion components using Singular Value Decomposition or, in the case of occluded or missing data, using an iterative bilinear solution technique. Perspective projection, however, involves division, which cannot be modeled by these bilinear equations. In order to extend shape and motion recovery to sequences containing foreshortening effects, we turn to the realm of general non-linear equation solution techniques.

In this section, we first show how to recover the projective shape and projective motion from an image sequence. A projective shape is a shape which is known only up to an arbitrary rotation, scaling or skewing of the axes, and scaling of each point on the object by the distance to any given point, as illustrated in Figure 24. There exists a projective transformation which will transform a projective shape into the actual Euclidean shape of the object (i.e. with the axes correctly scaled unskewed and the projective deformation removed), though the correct transformation is not always known. The problem of recovering projective shape and motion from an image sequence has been addressed by several researchers in the past few years [4][7][25][30][38]. Many produce only the projective shape and motion as their answer. Others compute the transformation which converts the projective solution into a Euclidean solution by requiring the direct measurement of the Euclidean world positions of five of the points, or by assuming coplanarity of certain points. We present a method for converting a projective shape and motion into a Euclidean one by applying normalization constraints to the motion, in a manner analogous to the way an affine shape and motion were converted to a Euclidean one in the factorization method.

4.1.1. Least Squares Formulation

Recall from equation (64) that the perspective projection of a point $\mathbf{s}_p$ onto the image plane in frame f is

$$P_p(f, p) = \begin{bmatrix} P_{x_p}(f, p) \\ P_{y_p}(f, p) \end{bmatrix} = \begin{bmatrix} l\dfrac{\mathbf{i}_f \cdot \mathbf{s}_p + x_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f} + o_x \\ la\dfrac{\mathbf{j}_f \cdot \mathbf{s}_p + y_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f} + o_y \end{bmatrix} \qquad (96)$$

This can be rewritten in the form

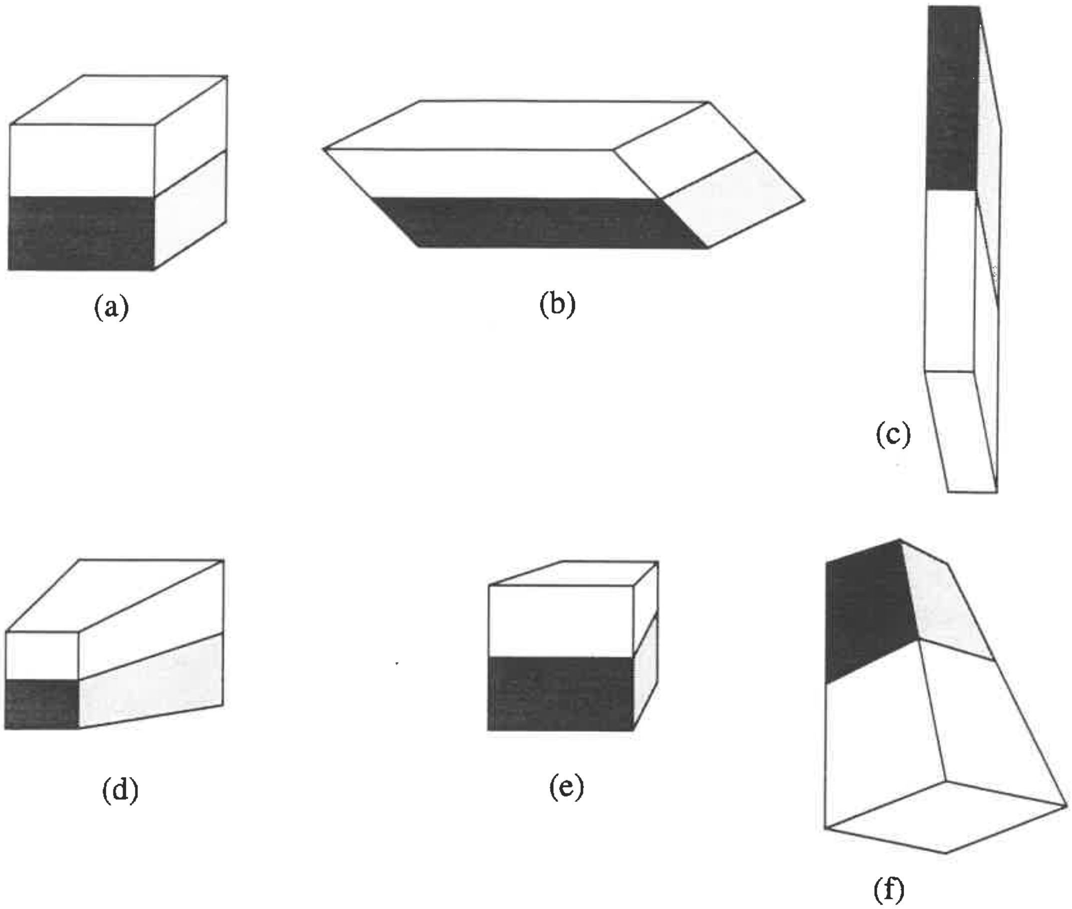


Figure 24. Affine Equivalence and Projective Equivalence
The objects (a), (b), and (c) are all affinely equivalent, since each is the result of arbitrarily scaling the axes or changing the angles between the axes of another. (a), (b), (c), (d), (e), and (f) are projectively equivalent, since each is the result of scaling the positions of the points of another based on the distance to some arbitrary point.

$$P_p(f, p) = \begin{bmatrix} \dfrac{l(\mathbf{i}_f + o_x\mathbf{k}_f) \cdot \mathbf{s}_p + lx_f + o_x z_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f} \\ \dfrac{la(\mathbf{j}_f + o_y\mathbf{k}_f) \cdot \mathbf{s}_p + lay_f + o_y z_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f} \end{bmatrix} \tag{97}$$

The projective formulation of these equations simply collects all of the motion and camera variables into a three vectors $\mathbf{m}_{f1}$, $\mathbf{m}_{f2}$, and $\mathbf{m}_{f3}$, similar to the way the motion parameters were collected into $\mathbf{m}_f$ and $\mathbf{n}_f$ in the previous methods. Using this notation, the perspective projection equations that predict the x- and y- positions of point p in frame f become

$$P_p(f,p) = \begin{bmatrix} P_{x_p}(f,p) \\ P_{y_p}(f,p) \end{bmatrix} = \begin{bmatrix} \dfrac{\mathbf{m}_{f1} \cdot \mathbf{s}_p}{\mathbf{m}_{f3} \cdot \mathbf{s}_p} \\ \dfrac{\mathbf{m}_{f2} \cdot \mathbf{s}_p}{\mathbf{m}_{f3} \cdot \mathbf{s}_p} \end{bmatrix} \tag{98}$$

Here the $\mathbf{s}_p$ vectors are written in homogeneous coordinates, where each $\mathbf{s}_p$ is augmented to be a 4-vector whose fourth element is set to 1. The $4 \times P$ matrix whose columns are the $\mathbf{s}_p$ vectors is called the *projective shape matrix*. The 3×4 *projective motion matrix* M_f for each frame is defined by collecting the three $\mathbf{m}_{fi}$ 4-vectors, whose values in terms of the camera motion and camera parameters can be determined from equation (97) as

$$M_f = \begin{bmatrix} \mathbf{m}_{f1} \\ \mathbf{m}_{f2} \\ \mathbf{m}_{f3} \end{bmatrix} = \begin{bmatrix} m_{f11} & m_{f12} & m_{f13} & m_{f14} \\ m_{f21} & m_{f22} & m_{f23} & m_{f24} \\ m_{f31} & m_{f32} & m_{f33} & m_{f34} \end{bmatrix} = \left[\begin{array}{c|c} l\,(\mathbf{i}_f + o_x \mathbf{k}_f) & l x_f + o_x z_f \\ la\,(\mathbf{j}_f + o_y \mathbf{k}_f) & la y_f + o_y z_f \\ \mathbf{k}_f & z_f \end{array} \right] \tag{99}$$

The projective shape and motion recovery problem is posed as, given a set of observed points $\mathbf{u}_{fp}$, compute the projective shape and motion which minimize the error

$$\begin{aligned} \varepsilon &= \sum_{f=1}^{F} \sum_{p=1}^{P} (P_p(f,p) - \mathbf{u}_{fp})^T (P_p(f,p) - \mathbf{u}_{fp})\, \gamma_{fp} \\ &= \sum_{f=1}^{F} \sum_{p=1}^{P} \left[\left(P_{x_p}(f,p) - u_{fp} \right)^2 + \left(P_{y_p}(f,p) - v_{fp} \right)^2 \right] \gamma_{fp} \end{aligned} \tag{100}$$

Here γ_{fp} are the confidence values assigned to each observation.

We will use the Levenberg-Marquardt optimization technique described in the next section to find the projective shape and motion which minimize the error ε. Note that due to the division, both the numerator and the denominator of the projection function can be multiplied by any arbitrary non-zero constant, and the resulting predicted image positions would be unchanged. Therefore each projective shape vector $\mathbf{s}_p$ can be scaled by an arbitrary non-zero constant, and each projective shape matrix M_f can similarly be multiplied by a non-zero constant, without changing resulting the error measure ε. To eliminate redundant degrees of freedom in the model, we fix the 4^{th} element of each $\mathbf{s}_p$ at 1 and set $m_{f34} = 1$.

4.1.2. Levenberg-Marquardt Solution Method

A review of the Levenberg-Marquardt technique for non-linear optimization was given in section 2.4.2. In the projective shape and motion recovery problem, the variable vector $\mathbf{a}$ consists of all of the projective motion and projective shape variables m_{fjk} and s_{pj} (not including those whose values have been fixed to 1.) The observed y_i values are simply the measured feature point positions u_{fp} and v_{fp}, while the predicted Y_i values are the functions

$P_x(f, p)$ and $P_y(f, p)$ given by equation (100) for the case of perspective projection using a projective shape and motion formulation, and $1/\sigma_i^2 = \gamma_{fp}$. Equations (68) and (70) for the projective shape and motion recovery problem can be written as

$$\alpha_{kl} = \sum_{f=1}^{F}\sum_{p=1}^{P} \gamma_{fp}\left[\left(\frac{\partial}{\partial a_k}P_x(f, p)\right)\left(\frac{\partial}{\partial a_l}P_x(f, p)\right) + \left(\frac{\partial}{\partial a_k}P_y(f, p)\right)\left(\frac{\partial}{\partial a_l}P_y(f, p)\right)\right] \tag{101}$$

$$\beta_k = \sum_{f=1}^{F}\sum_{p=1}^{P} \gamma_{fp}\left[(u_{fp} - P_x(f, p))\frac{\partial}{\partial a_k}P_x(f, p) + (v_{fp} - P_y(f, p))\frac{\partial}{\partial a_k}P_y(f, p)\right] \tag{102}$$

Notice that each $P_x(f, p)$ or $P_y(f, p)$ is a function only of the shape variables for point p and the motion variables for frame f. Therefore many of the partial derivatives are zero, which in turn causes many of the terms of the summation to be zero. Rather than requiring the summation of $2FP$ terms, each element of the vector β requires the summation of $2P$ or $2F$ terms.

$$\beta_k = \begin{cases} \displaystyle\sum_{p=1}^{P} \gamma_{fp}\left[(u_{fp} - P_x(f, p))\frac{\partial}{\partial a_k}P_x(f, p) + (v_{fp} - P_y(f, p))\frac{\partial}{\partial a_k}P_y(f, p)\right] \\ \qquad\qquad\qquad\qquad \text{if } a_k \text{ is a motion variable for frame } f \\ \displaystyle\sum_{f=1}^{F} \gamma_{fp}\left[(u_{fp} - P_x(f, p))\frac{\partial}{\partial a_k}P_x(f, p) + (v_{fp} - P_y(f, p))\frac{\partial}{\partial a_k}P_y(f, p)\right] \\ \qquad\qquad\qquad\qquad \text{if } a_k \text{ is a shape variable for point } p \end{cases} \tag{103}$$

Similarly many terms of the α matrix have zero values, while others require the summation of a smaller number of terms.

$$\alpha_{kl} = \begin{cases} 0 & \text{if } a_k \text{ and } a_l \text{ are shape variables from different points} \\ 0 & \text{if } a_k \text{ and } a_l \text{ are motion variables from different frames} \\ 0 & \text{if } a_k \text{ is a shape variable for point } p\text{, } a_l \text{ is a motion variable for frame } f\text{,} \\ & \text{and } \gamma_{fp} = 0 \\ \gamma_{fp}\left[\left(\frac{\partial}{\partial a_k}P_x(f,p)\right)\left(\frac{\partial}{\partial a_l}P_x(f,p)\right) + \left(\frac{\partial}{\partial a_k}P_y(f,p)\right)\left(\frac{\partial}{\partial a_l}P_y(f,p)\right)\right] & \\ & \text{if } a_k \text{ shape variable for point } p\text{, } a_l \text{ motion variable for frame } f\text{, } \gamma_{fp} \neq 0 \\ \sum\limits_{2F \text{ terms}} \gamma_{fp}\left(\left[\left(\frac{\partial}{\partial a_k}P_x(f,p)\right)\left(\frac{\partial}{\partial a_l}P_x(f,p)\right) + \left(\frac{\partial}{\partial a_k}P_y(f,p)\right)\left(\frac{\partial}{\partial a_l}P_y(f,p)\right)\right]\right) & \\ & \text{if } a_k \text{ and } a_l \text{ shape variables from same point} \\ \sum\limits_{P \text{ terms}} \gamma_{fp}\left(\left[\left(\frac{\partial}{\partial a_k}P_x(f,p)\right)\left(\frac{\partial}{\partial a_l}P_x(f,p)\right) + \left(\frac{\partial}{\partial a_k}P_y(f,p)\right)\left(\frac{\partial}{\partial a_l}P_y(f,p)\right)\right]\right) & \\ & \text{if } a_k \text{ and } a_l \text{ motion variables from same frame} \end{cases} \tag{104}$$

We order the variables in $\mathbf{a}$ to take advantage of the sparseness of α. The 11 motion variables for the first frame comprise the first 11 elements of $\mathbf{a}$, the 11 motion variables for the second frame are the next 11 elements of $\mathbf{a}$, and so on, so that altogether the motion variables comprise the first $11F$ elements of $\mathbf{a}$. The shape variables are ordered similarly, three variables per feature point, so that $\mathbf{a}$ has a total of $11F + 3P$ elements. This places the elements with generally large magnitudes along a block diagonal portion of the Hessian matrix α as shown in Figure 25.

The block-diagonal dominance of the matrix allows us to use the preconditioned conjugate gradient method described in [31] to efficiently solve equation (71), which is required at every step of the minimization. The preconditioned conjugate gradient method is itself an iterative technique which is well-suited to solving sparse linear systems. To solve the linear system $\alpha\delta\mathbf{a} = \beta$ for $\delta\mathbf{a}$, we are not required to supply the matrix α in any specific representation. Rather we supply the conjugate gradient routine with three functions. The first computes the product of α with an arbitrary vector, and the second computes the product of α^T with an arbitrary vector. Since α is symmetric in our problem, the same function performs both of these operations. The third function must provide a solution to linear systems of the form $\alpha^*\mathbf{x} = \mathbf{b}$, where α^* is a preconditioner matrix which approximates α and which we can invert easily. We use the block diagonal portion of α as the preconditioner α^*. Since the elements of α with the largest magnitudes lie along the block diagonal portion of α, α^* is a good approximation to α. Furthermore we can solve systems of the form $\alpha^*\mathbf{x} = \mathbf{b}$ efficiently by inverting each of the small matrices lying along the diagonal individually.

Since we do not need to provide the matrix α but only three functions, the conjugate gradient method frees us to use any efficient form of storage for α. Many elements of this

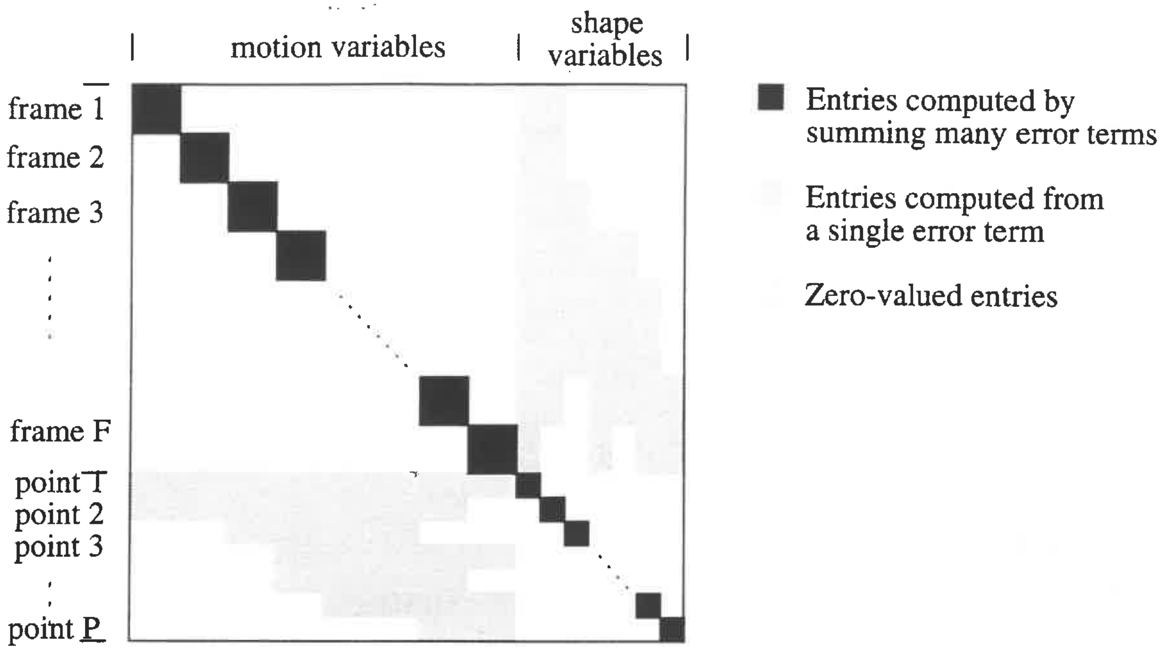


Figure 25. Hessian matrix for projective shape from motion recovery
The diagonal blocks are 11x11 for the motion variables and 3x3 for the shape variables. The actual shape of the shaded area is determined by the fill pattern of feature observability for the particular sequence.

$(11F + 3P) \times (11F + 3P)$ matrix are zero, so we reduce storage requirements by using a sparse matrix representation that stores only potentially non-zero elements of the matrix. Our representation stores each column independently as a sequence of non-zero strips of elements. This representation enables the product of α^T with a vector to be computed very efficiently. The product of α with a vector is computed using the same method, since α is symmetric. We could have further halved the storage requirements for α by taking advantage of its symmetry, but that would have increased the complexity and computation required to multiply the matrix by a vector.

Using the conjugate gradient method to solve the system $\alpha\delta\mathbf{a} = \beta$ has an additional benefit, which is that a solution is provided even when α is not of full rank. This frees us from having to arbitrarily fix certain parameters to bring the system to a minimal parameterization, such as fixing the first frame's motion or choosing four points as a projective basis whose positions will remain fixed.

The expressions for computing α and β in equations (103) and (104) still require the computation of the derivatives of the predicted position $P(f, p)$ with respect to each variable a_k, where each a_k corresponds to either a single shape variable s_{pi} or a motion variable m_{fij}. First, we define

$$x_{fp} = \mathbf{m}_{f1} \cdot \mathbf{s}_p \qquad y_{fp} = \mathbf{m}_{f2} \cdot \mathbf{s}_p \qquad z_{fp} = \mathbf{m}_{f3} \cdot \mathbf{s}_p \tag{105}$$

The derivatives with respect to each of the shape and motion variables are then given by

$$\frac{\partial}{\partial s_{pi}} P_p(f,p) = \begin{bmatrix} \dfrac{m_{f1i} z_{fp} - m_{f3i} x_{fp}}{z_{fp}^2} \\ \dfrac{m_{f2i} z_{fp} - m_{f3i} y_{fp}}{z_{fp}^2} \end{bmatrix} \tag{106}$$

$$\frac{\partial}{\partial m_{f1i}} P_p(f,p) = \begin{bmatrix} \dfrac{s_{pi}}{z_{fp}} \\ 0 \end{bmatrix} \qquad \frac{\partial}{\partial m_{f2i}} P_p(f,p) = \begin{bmatrix} 0 \\ \dfrac{s_{pi}}{z_{fp}} \end{bmatrix} \qquad \frac{\partial}{\partial m_{f3i}} P_p(f,p) = \begin{bmatrix} \dfrac{s_{pi} x_{fp}}{z_{fp}^2} \\ \dfrac{s_{pi} y_{fp}}{z_{fp}^2} \end{bmatrix} \tag{107}$$

Our solution should be restricted so that $z_{fp} > 0$ for all observed f and p. This merely requires than all 3D points lie in front of the camera when they are imaged. Hopefully, as long as the z_{fp} computed using the initial value are all positive, we should never encounter a negative z_{fp}, since as we approach this region of the search space z_{fp} approaches zero, which causes the error to go to infinity. However, occasionally the solution takes a large "step" in which is crosses over a $z_{fp} = 0$ boundary and reaches a part of the search space where a z_{fp} is negative. These steps should be rejected regardless of whether the total error e at those points is larger or smaller than the previous position, because they clearly indicate that too large a step was taken, that the current region of the search space is not well modeled by a quadratic, and that we should try smaller steps. Rejecting them causes λ to increase so that the method will take smaller steps towards the minimum.

4.1.3. Depth-shape Formulation

The above formulation represents each feature point by three variables, the feature point's coordinates in the world coordinate system. However, in a sense a feature is defined by the position at which it was detected in the first image. Therefore a feature's position in the first frame is not corrupted by noise, because its position in the first frame defines the feature. Therefore we can reduce the number of unknowns per feature point by one-third by defining each feature by a single unknown, the depth of the feature in the first frame, like the standard formulation of the stereo problem. Changing this distance variable will cause the point's predicted position in subsequent images to change accordingly, but its position in the first image will always remain fixed to the position at which it was originally detected. This formulation not only ensures that the features do not drift from their originally defined positions, but significantly reduces the size of the matrix α, thereby speeding the computation. This constraint is illustrated in Figure 26.

As long as all features are visible in the first frame of the sequence, this change requires only

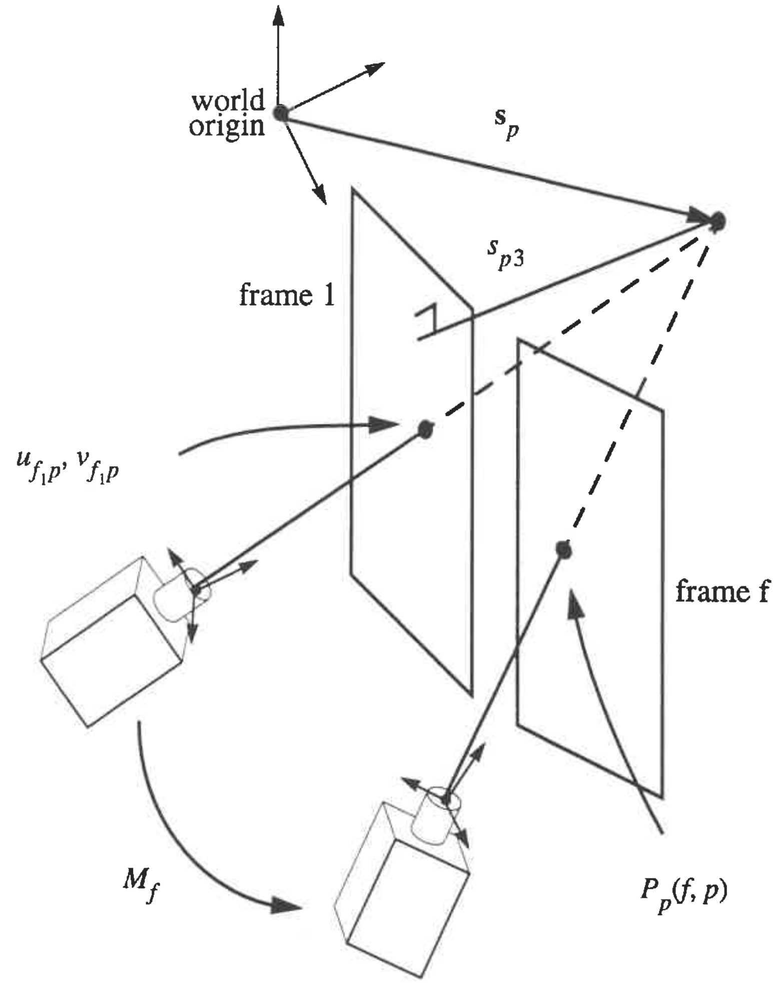


Figure 26. Shape-depth formulation

Given a point's position in the first frame, it is constrained to lie along the ray connecting the camera focus in the first frame to the position of the point on the image plane in the first frame and extending to the 3D point itself. This reduces it to a single unknown, th e distance along that ray at which the actual 3D point is located.

a minor change to the previously described method. The first and second elements of each shape vector $\mathbf{s}_p$ are no longer variables, but are replaced with functions of the third element, s_{p3}, and the first frame's motion variables. These functions are determined by solving the "inverse projection" problem -- projecting the known position in the first image away from the camera by the given depth. We solve the system

$$\begin{bmatrix} u_{f_1p} \\ v_{f_1p} \end{bmatrix} = \begin{bmatrix} \dfrac{m_{f_111}s_{p1} + m_{f_112}s_{p2} + m_{f_113}s_{p3} + m_{f_114}}{m_{f_131}s_{p1} + m_{f_132}s_{p2} + m_{f_133}s_{p3} + m_{f_134}} \\ \dfrac{m_{f_121}s_{p1} + m_{f_122}s_{p2} + m_{f_123}s_{p3} + m_{f_124}}{m_{f_131}s_{p1} + m_{f_132}s_{p2} + m_{f_133}s_{p3} + m_{f_134}} \end{bmatrix} \qquad (108)$$

for s_1 and s_2 as functions of s_3. Multiplying through by the denominator turns this into linear system of 2 equations in the two unknowns s_{p1} and s_{p2}.

$$\begin{bmatrix} (m_{f_131}u_{1p} - m_{f_111}) & (m_{f_132}u_{1p} - m_{f_112}) \\ (m_{f_131}v_{1p} - m_{f_121}) & (m_{f_132}v_{1p} - m_{f_122}) \end{bmatrix} \begin{bmatrix} s_{p1} \\ s_{p2} \end{bmatrix} = -\begin{bmatrix} (m_{f_133}u_{1p} - m_{f_113})s_{p3} + (m_{f_134}u_{1p} - m_{f_114}) \\ (m_{f_133}v_{1p} - m_{f_123})s_{p3} + (m_{f_134}v_{1p} - m_{f_124}) \end{bmatrix} \qquad (109)$$

This is easily solved to find s_{p1}, s_{p2}, and their derivatives $\partial s_{p1}/\partial s_{p3}$ and $\partial s_{p2}/\partial s_{p3}$. This gives the derivative of the predicted feature position with respect to the single shape variable per point, s_{p3}, as

$$\frac{\partial}{\partial s_{p3}} P_p(f, p) = \begin{bmatrix} \dfrac{\left(m_{f11}\dfrac{\partial s_{p1}}{\partial s_{p3}} + m_{f11}\dfrac{\partial s^2{}_{p2}}{\partial s^3_{p3}} + m_{f13}\right)z_{fp} - \left(m_{f31}\dfrac{\partial s_{p1}}{\partial s_{p3}} + m_{f32}\dfrac{\partial s_{p2}}{\partial s_{p3}} + m_{f33}\right)x_{fp}}{z^2_{fp}} \\ \dfrac{\left(m_{f21}\dfrac{\partial s_{p1}}{\partial s_{p3}} + m_{f22}\dfrac{\partial s_{p2}}{\partial s_{p3}} + m_{f23}\right)z_{fp} - \left(m_{f31}\dfrac{\partial s_{p1}}{\partial s_{p3}} + m_{f32}\dfrac{\partial s_{p2}}{\partial s_{p3}} + m_{f33}\right)y_{fp}}{z^2_{fp}} \end{bmatrix} \qquad (110)$$

Since s_{p1} and s_{p2} are now varied such that the feature's position in the first frame is fixed, changing the first frame's camera parameters will no longer have any effect on the total error. Therefore we leave them fixed at their initial value rather than solving for them as unknowns.

This single variable "depth-shape" formulation can only be used when all of the feature points are visible in the first image of the sequence. Since s_{p1} and s_{p2} are in fact functions not only of s_{p3}, but also of the motion parameters in the first frame in which the point was seen, if that frame's motion parameters were variables and not constants, then we would need to update the Hessian matrix to account for this. Error terms for the observation for point p and f would not only depend on the shape variables for point p and the motion variables for frame f, but would also depend on the motion variables for some frame f', the frame in which point p was first seen. Although this could be easily accommodated within the Levenberg-Marquardt minimization framework, it would alter the pattern of the Hessian matrix. The upper-left block of the matrix would no longer be block diagonal, but would contain smaller blocks scattered throughout the currently unfilled areas. This could destroy its block-diagonal dominance and significantly increase the number of iterations of the gradient descent method required to solve the large system. However, this increase might be offset by the reduction of the number of variables from $6F+3P$ to $6F+P$. We have not implemented a system capable of applying the depth-shape formulation to sequences in which some of the points are not seen in the first image, so currently such sequences must be solved using three variables per point.

4.1.4. Projective Normalization

It is well-known that a projective motion matrix M_f defined by equation (99) can be written as the product of two matrices: a matrix of intrinsic parameters representing the properties of the camera which remain constant for all frames, and a matrix of motion parameters which varies with each frame.

$$M_f = \begin{bmatrix} l & 0 & o_x \\ 0 & la & o_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} \mathbf{i}_f & x_f \\ \mathbf{j}_f & y_f \\ \mathbf{k}_f & z_f \end{array}\right] = K\left[R_f \mid \mathbf{t}_f\right] \tag{111}$$

Here K is the intrinsic camera parameter matrix, R_f collects $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ into a 3×3 rotation matrix, and $\mathbf{t}_f$ collects the x_f, y_f, and z_f terms. The intrinsic camera parameter matrix includes the focal length l, the aspect ratio a, and the center of projection (o_x, o_y).

Unfortunately, the recovered projective shape and motion are unknown up to an arbitrary 4×4 matrix. That is, we have solved for the $\hat{M}_f$ and $\hat{\mathbf{s}}_p$ which minimize the error between the observed and predicted image positions, but in fact we could multiply all of the motion matrices by an arbitrary 4×4 matrix A and multiply the projective shape vectors by A^{-1} and the resulting predicted feature points and total error ε would be exactly the same. Furthermore we can scale each of the projective motion matrices and each of the shape vectors independently by arbitrary scaling factors without changing the resulting point positions or total error. Because of this unknown 4×4 matrix and unknown scaling factors, the recovered projective motion matrices $\hat{M}_f$ are not likely to be expressible in the form shown in equation (111), which requires that the first three columns of all of the projective motion matrices be simple rotations of a single, constant matrix K. In order to find the actual Euclidean shape and motion, we must first find the 4×4 matrix A which rotates each $\hat{M}_f$ into a matrix of that form.

$$\begin{gathered} M_f \cong \hat{M}_f A \\ \mathbf{s}_p \cong A^{-1}\hat{\mathbf{s}}_p \end{gathered} \tag{112}$$

where $\cong$ indicates equivalence up to a scaling factor. Just as with the previous factorization methods, we find A by applying constraints to the form of the motion matrices based on the orthonormality of the camera axes $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ in each frame.

Introducing an unknown scale factor d_f for each frame, we can rewrite the $\cong$ relation using equality.

$$M_f = K\left[R_f \;\middle|\mathbf{t}_f\right] = \left[KR_f\middle|\;K\mathbf{t}_f\right] = d_f\hat{M}_fA \tag{113}$$

Since M_f contains twelve elements, equation (113) gives twelve equations for each frame. The first three columns of M_f can be constrained to be the product of an intrinsic camera matrix K, constant for all frames, and a rotation matrix. However, the translation component of the motion for each frame is arbitrary, so equating the fourth columns provides no constraints on A. Therefore we use only the first three columns to provide constraints.

$$KR_f = \left(d_f\hat{M}_fA\right)\Big|_{\text{columns 1-3}} = d_f\hat{M}_f\hat{A} \tag{114}$$

where

$$\hat{A} = A\big|_{\text{columns 1-3}} \tag{115}$$

Multiplying each side of equation (114) by its transpose, and taking advantage of the fact that the transpose of a rotation matrix is also its inverse, we derive

$$\begin{aligned} R_f &= d_fK^{-1}\hat{M}_f\hat{A} \\ R_fR_f^T &= d_fK^{-1}\hat{M}_f\hat{A}\left(d_fK^{-1}\hat{M}_f\hat{A}\right)^T \\ I &= d_f^2K^{-1}\hat{M}_f\hat{A}\hat{A}^T(K^{-1}\hat{M}_f)^T \end{aligned} \tag{116}$$

Here we assume that the camera calibration matrix K is known, and we introduce the calibrated projective motion matrix $\hat{M}'_f = K^{-1}\hat{M}_f$. Introducing $Q = \hat{A}\hat{A}^T$, we see that

$$\begin{bmatrix} \frac{1}{d_f^2} & 0 & 0 \\ 0 & \frac{1}{d_f^2} & 0 \\ 0 & 0 & \frac{1}{d_f^2} \end{bmatrix} = \hat{M}'_fQ\hat{M}'^T_f \tag{117}$$

Since we do not know the scale factors d_f, the diagonal elements can only be constrained to have equal magnitudes. Furthermore the product $\hat{M}'_fQ\hat{M}'^T_f$ is guaranteed to be symmetric, so the lower triangular elements provide redundant constraints. Therefore there are six constraints on Q per frame.

$$
\begin{aligned}
(\hat{M}_f Q \hat{M}_f^T)_{11} &= (\hat{M}_f Q \hat{M}_f^T)_{22} \\
(\hat{M}_f Q \hat{M}_f^{\prime T})_{22} &= (\hat{M}_f Q \hat{M}_f^T)_{33} \\
(\hat{M}_f Q \hat{M}_f^T)_{33} &= (\hat{M}_f Q \hat{M}_f^T)_{11} \\
(\hat{M}_f Q \hat{M}_f^T)_{12} &= 0 \\
(\hat{M}_f Q \hat{M}_f^T)_{13} &= 0 \\
(\hat{M}_f Q \hat{M}_f^T)_{23} &= 0
\end{aligned}
\qquad (118)
$$

Of course, the third constraint is actually redundant since it follows by transitivity of the first two. However, choosing only the first two constraints would make the solution of the system more sensitive to errors in $(\hat{M}_f Q \hat{M}_f^{\prime T})_{22}$ than to the other terms. So in the interests of robustness and not favoring any one term over another, we retain the redundant constraint. These constraints are linear in the 10 unique elements of the 4×4 symmetric matrix Q, so they can be solved efficiently. They are homogeneous constraints, to we add the arbitrary constraint that

$$
(\hat{M}_f Q \hat{M}_f^T)_{11} = 1 \qquad (119)
$$

to avoid the solution $Q = 0$.

The Q matrix is decomposed to determine A in a manner very similar to that of section 2.2.3. Since Q is the product of a 3×4 matrix $\hat{A}$ and its transpose, we expect that Q will have one zero eigenvalue. Indeed, when we use Jacobi Transformation to compute the eigenvalue decomposition $Q = L \Lambda L^T$ we generally find that the last eigenvalue in the diagonal matrix Λ is much smaller than the others. The first three columns of $L \Lambda^{1/2}$ become our answer for $\hat{A}$, the first three columns of A.

There are no constraints with which to determine the fourth column of A, which effects the translational component of the motion result in each frame. This portion of the transformation is indeed arbitrary, because up to this point we have not done anything to fix the world origin. We choose the fourth column to be $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$, and later, after the conversion to a Euclidean solution is complete, we will transform the camera translation and object shape results to affix the origin to the object's mass center, for consistency with previous formulations.

Once the 4×4 matrix A has been computed, the camera motion for each frame and position of each point are determined by equation (112). After multiplication by A^{-1}, the fourth element of s_p may no longer equal one. However, we know that each shape vector can be multiplied by a scalar multiple without changing the results, so we scale each shape vector so that the fourth element is 1, to achieve a shape vector consistent with the Euclidean formulation of equation (96). The first three elements of each resulting vector are the Euclidean point positions.

Since each motion matrix M_f is still defined only up to a scaling factor, each is scaled so that $|\mathbf{i}_f| = |\mathbf{j}_f| = |\mathbf{k}_f| = 1$. The first three columns of the resulting matrices contain the Euclidean orientation of the camera in each frame, while the last column contains the position of the camera in every frame. As with the previous factorization methods, the resulting $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ are not guaranteed to be orthonormal, so we compute the orthonormal vectors closest to those given values.

4.1.5. Projective Normalization for Distant Objects

The above-described method works quite well for objects which are near the camera. However, in image sequences in which the object is very distant from the camera and foreshortening effects are small, it performs poorly. Due to the large distance z_f, the magnitudes of the actual Euclidean m_{f31}, m_{f32}, and m_{f33} are very small compared to the magnitude of m_{f34}. In computing the projective shape and motion, large errors in these quantities have only a minimal impact on the error sum ε. In short, the relative errors in these quantities are much larger than the errors in the distance m_{f34} or in the other rows of M_f. Yet in the projective metric constraints (118), the constraints using these elements are weighed equally with other constraints. We therefore introduce a weighting factor σ which specifies the weights to use for the constraints in (118) involving the third row of M_f. The version of equation (118) for distant objects becomes

$$\begin{aligned}
(\hat{M}_f Q \hat{M}_f^T)_{11} &= (\hat{M}_f Q \hat{M}_f^T)_{22} \\
(\hat{M}_f Q \hat{M}_f^T)_{22}\sigma &= (\hat{M}_f Q \hat{M}_f^T)_{33}\sigma \\
(\hat{M}_f Q \hat{M}_f^T)_{33}\sigma &= (\hat{M}_f Q \hat{M}_f^T)_{11}\sigma \\
(\hat{M}_f Q \hat{M}_f^T)_{12} &= 0 \\
(\hat{M}_f Q \hat{M}_f^T)_{13}\sigma &= 0 \\
(\hat{M}_f Q \hat{M}_f^T)_{23}\sigma &= 0
\end{aligned} \tag{120}$$

The weighting factor σ needs to be adjusted depending on the distance of the object from the camera for the particular problem. If the actual shape and motion were known rather than just a projective transformation of the shape and motion, we could estimate σ based on the amount of foreshortening present in the images. The term in denominator of the projection function $P_p(f, p)$ is the sum $m_{f31}s_{p1} + m_{f32}s_{p2} + m_{f33}s_{p3} + m_{f34}$. The sum of the first three terms of this summation varies with each point, while the third term is constant for each frame. If the sum of the first three terms is small relative to the fourth term m_{f34}, then there is very little foreshortening in the scene. In fact, the projection may be accurately modeled by scaled orthographic projection, and we have little confidence in m_{f31}, m_{f32}, and m_{f33} and should use a small value of σ. If the sum of the first three terms is large relative to m_{f34}, then foreshortening effects are significant and we should use a large value of σ. In fact, the average ratio of $|m_{f31}s_{p1} + m_{f32}s_{p2} + m_{f33}s_{p3}|$ to m_{f34} should provide an excellent estimate of σ. We compute σ as

$$\sigma = \frac{\sum_{f=1}^{F}\sum_{p=1}^{P}\left|m_{f31}s_{p1}+m_{f32}s_{p2}+m_{f33}s_{p3}\right|}{\sum_{f=1}^{F}\sum_{p=1}^{P}m_{f34}} \qquad (121)$$

The problem is that the above reasoning assumes a Euclidean interpretation of the motion matrix, that is, that m_{f34} represents "depth" and s_p represents the Euclidean object shape. If these are projective values, then the shape contains an arbitrary projective deformation, so the ratio will describe a total amount of foreshortening necessary to remove the object's arbitrary foreshortening deformation before applying additional foreshortening to project the shape into the particular frame. Therefore we need to convert the projective solution to a Euclidean one in order to estimate σ, yet without σ we cannot convert the projective solution into a Euclidean one!

The method of the previous section performs too poorly for distant objects to consider using it to estimate σ, so another way is necessary. The key to our procedure is to realize that when the object is distant, its Euclidean motion matrix will be well-approximated by a scaled orthographic projection matrix. A scaled orthographic matrix will have zero-valued m_{f31}, m_{f32}, and m_{f33}, since that will cause the denominator of the perspective projection function $P_p(f, p)$ to be constant for each frame f, independent of the point p. Such a matrix still has the same form as shown in equation (113), except that the lowest row of the rotation matrix is replaced with zeros.

We first find a transformation Q_{so} which best transforms the projective motion matrices into motion matrices with a scaled orthographic form. Repeating the derivation of the previous section with the new definition of R produces the following constraints.

$$\begin{aligned}\left(\hat{M}'_f Q_{so}\hat{M}'^T_f\right)_{11} &= \left(\hat{M}'_f Q_{so}\hat{M}'^T_f\right)_{22}\\ \left(\hat{M}'_f Q_{so}\hat{M}'^T_f\right)_{12} &= 0\\ \left(\hat{M}'_f Q_{so}\hat{M}'^T_f\right)_{13} &= 0\\ \left(\hat{M}'_f Q_{so}\hat{M}'^T_f\right)_{23} &= 0\\ \left(\hat{M}'_f Q_{so}\hat{M}'^T_f\right)_{33} &= 0\end{aligned} \qquad (122)$$

We use these constraints to compute Q_{so} and then compute A_{so} from Q_{so} in the same manner as described in the previous section. This matrix is the matrix which best transforms the projective motion into a projective motion matrix having scaled orthographic form. This transformation will only be satisfied perfectly if the object is indeed very distant and foreshortening effects are negligible, yet in any case, the transformation provides us with an estimate of the actual shape and motion by multiplying $\hat{M}_f$ by A_{so} and A_{so}^{-1} by $\hat{S}_p$. We use this shape and motion solution to estimate σ from equation (121).

Once σ has been computed, we find Q using equation (120), and compute A from Q exactly as before. To avoid reliance on possibly noisy entries in the denominator of M_f, we scale the resulting motion matrices so that $|\mathbf{i}_f| = |\mathbf{j}_f| = 1$, and compute $\mathbf{k}_f = \mathbf{i}_f \times \mathbf{j}_f$.

Finally note that when the depth is so extreme that foreshortening effects are completely overcome by noise, using Q_{so} may provide a better solution than using the Q computed using equation (120). However, in such cases using the scaled orthographic or paraperspective factorization method can be expected to produce better results. Using Q_{so} will enforce the scaled orthographic nature of the projection to remove affine or projective deformations from the projective shape, but small errors in Q_{so} will still cause small affine or projective deformations to remain. The previous factorization methods do not allow projective deformations to enter the solution during decomposition, so no projective deformations will exist in the result.

4.2. Non-Linear Euclidean Optimization for Shape and Motion Recovery

The projective shape and motion recovery step of the projective factorization actually uses a higher number of variables than there are degrees of freedom in the original problem. Eleven motion parameters per frame are used, when in fact each camera position has only three rotational and three translational degrees of freedom. In the presence of noise, there is no guarantee that the computed projective motion can be transformed by any 4×4 matrix into a motion of the correct Euclidean form, in which the camera axes are orthonormal and the camera calibration parameters constant from frame to frame. In essence, we merely cross our fingers and hope that we can transform the recovered motion into something close to Euclidean form. If we can transform it into something "close" to the right form, we hope that the computed camera orientations and translations will be "close" to the correct answer. When the feature measurements are very noisy, this may not be true. Even when feature measurements are fairly accurate, more accurate shape and motion results can be expected if we impose all of the constraints the problem has to offer.

In this section we introduce a method which ensures that the solution is of a valid Euclidean form with orthonormal camera axes and time-invariant calibration parameters throughout the minimization rather than imposing these constraints as a post-processing step. Applying these constraints should ensure more accurate shape and motion reconstruction. Furthermore, this method allows us to treat the camera parameters as additional variables to solve for, reducing the burden of camera calibration.

The problem is actually not so different from the projective factorization step. We still use Levenberg-Marquardt optimization to find the shape and motion which minimize the error sum defined in equation (100), and still use the perspective projection equations defined in (97). However, we now minimize the errors in the perspective projection equations using the Euclidean shape, motion, and camera parameters directly as the variables to be refined, rather than using the projective motion variables m_{fij}. We write the rotation in terms of the three rotation angles θ_f, φ_f, and ω_f introduced in equations (74), so that each frame's $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ vectors are guaranteed to be orthonormal.

$$R_f = \begin{bmatrix} \mathbf{i}_f \\ \mathbf{j}_f \\ \mathbf{k}_f \end{bmatrix} = \begin{bmatrix} \cos\theta_f\cos\varphi_f & \sin\theta_f\cos\varphi_f & -\sin\varphi_f \\ (\cos\theta_f\sin\varphi_f\sin\omega_f - \sin\theta_f\cos\omega_f) & (\sin\theta_f\sin\varphi_f\sin\omega_f + \cos\theta_f\cos\omega_f) & \cos\varphi_f\sin\omega_f \\ (\cos\theta_f\sin\varphi_f\cos\omega_f + \sin\theta_f\sin\omega_f) & (\sin\theta_f\sin\varphi_f\cos\omega_f - \cos\theta_f\sin\omega_f) & \cos\varphi_f\cos\omega_f \end{bmatrix} \tag{123}$$

We order our variables in a manner similar to that of the projective factorization. The three rotational variables for each frame (θ_f, φ_f, and ω_f) are followed by the three translational variables (x_f, y_f, and z_f), for a total of six variables per frame. Following that are the three shape variables per point, representing the Euclidean three-dimensional position of each point. Finally, we add the four camera variables; the focal length, the aspect ratio, and the image center x- and y- coordinates. These variables are solved for simultaneously with the recovery of shape and motion. The reasoning that produced equations (103) and (104) in the projective case still holds in the Euclidean case, giving the Hessian matrix the form shown in Figure 27.

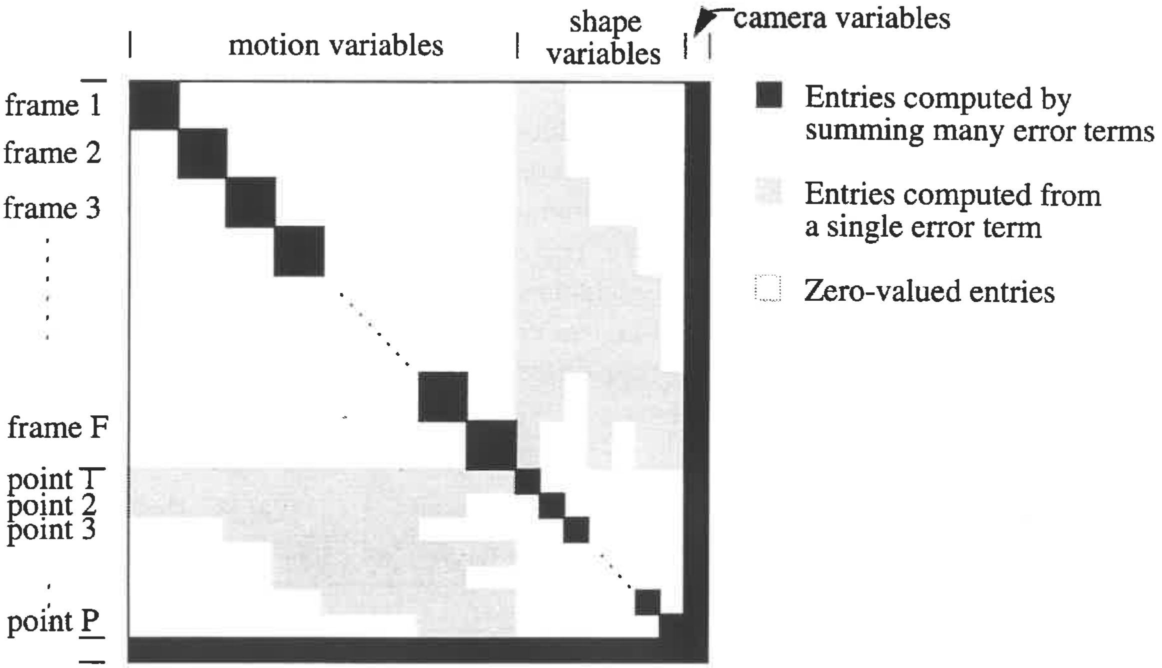


Figure 27. Hessian matrix for Euclidean shape from motion recovery. The diagonal blocks are 6x6 for the motion variables and 3x3 for the shape variables, while there are 4 camera variables.

Our experience shows that the focal length l trades off very easily with the depth to the object z_f. which makes their precise recovery difficult and slows the rate of the system's convergence. Since the scaling factor of M_f is arbitrary, we divide it by l and introduce a perspective distortion factor $\mu = 1/l$ and a relative depth factor $d_f = z_f/l$. Solving for these variables is more numerically stable than solving for l and z_f. In cases in which the object is far from the camera and foreshortening effects are overcome by noise, the best solution may be one with infinite focal length, i.e. a scaled orthographic solution. Using the μ and d_f formulation allows a graceful change to scaled orthography by letting μ approach zero, while a method explicitly representing the focal length would need the focal length to approach infinity in order to model scaled orthography. The motion matrix is defined in terms of these parameters as

$$M_f = \begin{bmatrix} i_{f1} + o_x\mu k_{f1} & i_{f2} + o_x\mu k_{f2} & i_{f3} + o_x\mu k_{f3} & x_f + o_x d_f \\ aj_{f1} + o_y\mu k_{f1} & aj_{f2} + o_y\mu k_{f2} & aj_{f3} + o_y\mu k_{f3} & ay_f + o_y d_f \\ \mu k_{f1} & \mu k_{f2} & \mu k_{f3} & d_f \end{bmatrix} \qquad (124)$$

Even with this formulation, experience shows that precise recovery of camera parameters is difficult unless the initial values are accurate. When reliable initial estimates of the camera calibration parameters are unavailable, it may be preferable to use the projective shape recovery method to recover the projective shape and motion, and then use other information such as the known positions of a few "beacon" points or the knowledge that several points lie on a plane, to convert the solution into a Euclidean one.

We proceed with the Levenberg-Marquardt solution in the same manner as in the previous chapter, using equations (103) and (104) to compute α and β at each step of the minimization. The derivative of the projection function with respect to the shape variables is the same as for the projective method, given in (106). The equations for the derivatives of the projection function with respect to the variables, however, are defined differently, because M_f is defined as a function of the rotational and translational motion variables as well as the intrinsic camera parameters. Still using the expressions x_{fp}, y_{fp}, and z_{fp} defined for simplicity in equation (105), these derivatives are

$$\text{for } \upsilon = \theta_f, \varphi_f \text{ or } \omega_f \qquad \frac{\partial}{\partial\upsilon}P_p(f,p) = \begin{bmatrix} \dfrac{z_{fp}\left(\dfrac{\partial \mathbf{i}_f}{\partial\upsilon}\cdot\mathbf{s}_p\right) - x_{fp}\left(\dfrac{\partial \mathbf{k}_f}{\partial\upsilon}\cdot\mathbf{s}_p\right)}{z_{fp}^2} \\ \dfrac{z_{fp}\left(\dfrac{\partial \mathbf{j}_f}{\partial\upsilon}\cdot\mathbf{s}_p\right) - y_{fp}\left(\dfrac{\partial \mathbf{k}_f}{\partial\upsilon}\cdot\mathbf{s}_p\right)}{z_{fp}^2} \end{bmatrix} \qquad (125)$$

$$\frac{\partial}{\partial x_f}P_p(f,p) = \begin{bmatrix} \frac{1}{z_{fp}} \\ 0 \end{bmatrix} \qquad \frac{\partial}{\partial y_f}P_p(f,p) = \begin{bmatrix} 0 \\ \frac{a}{z_{fp}} \end{bmatrix} \qquad \frac{\partial}{\partial d_f}P_p(f,p) = \begin{bmatrix} \frac{-x_{fp}}{z_{fp}^2} \\ \frac{-y_{fp}}{z_{fp}^2} \end{bmatrix} \tag{126}$$

Analytic expressions for the derivatives $\frac{\partial R_f}{\partial \theta_f}$, $\frac{\partial R_f}{\partial \varphi_f}$, and $\frac{\partial R_f}{\partial \omega_f}$ are computed as functions of θ_f, φ_f, and ω_f from equation (123).

The derivatives of the projection function with respect to the camera calibration variables are

$$\frac{\partial}{\partial \beta}P_p(f,p) = \begin{bmatrix} \frac{-x_{fp}(\mathbf{r}_{f3}\cdot\mathbf{s}_p)}{z_{fp}^2} \\ \frac{-y_{fp}(\mathbf{r}_{f3}\cdot\mathbf{s}_p)}{z_{fp}^2} \end{bmatrix} \qquad \frac{\partial}{\partial a}P_p(f,p) = \begin{bmatrix} 0 \\ \frac{\mathbf{r}_{f3}\cdot\mathbf{s}_p + y_f}{z_{fp}} \end{bmatrix} \tag{127}$$

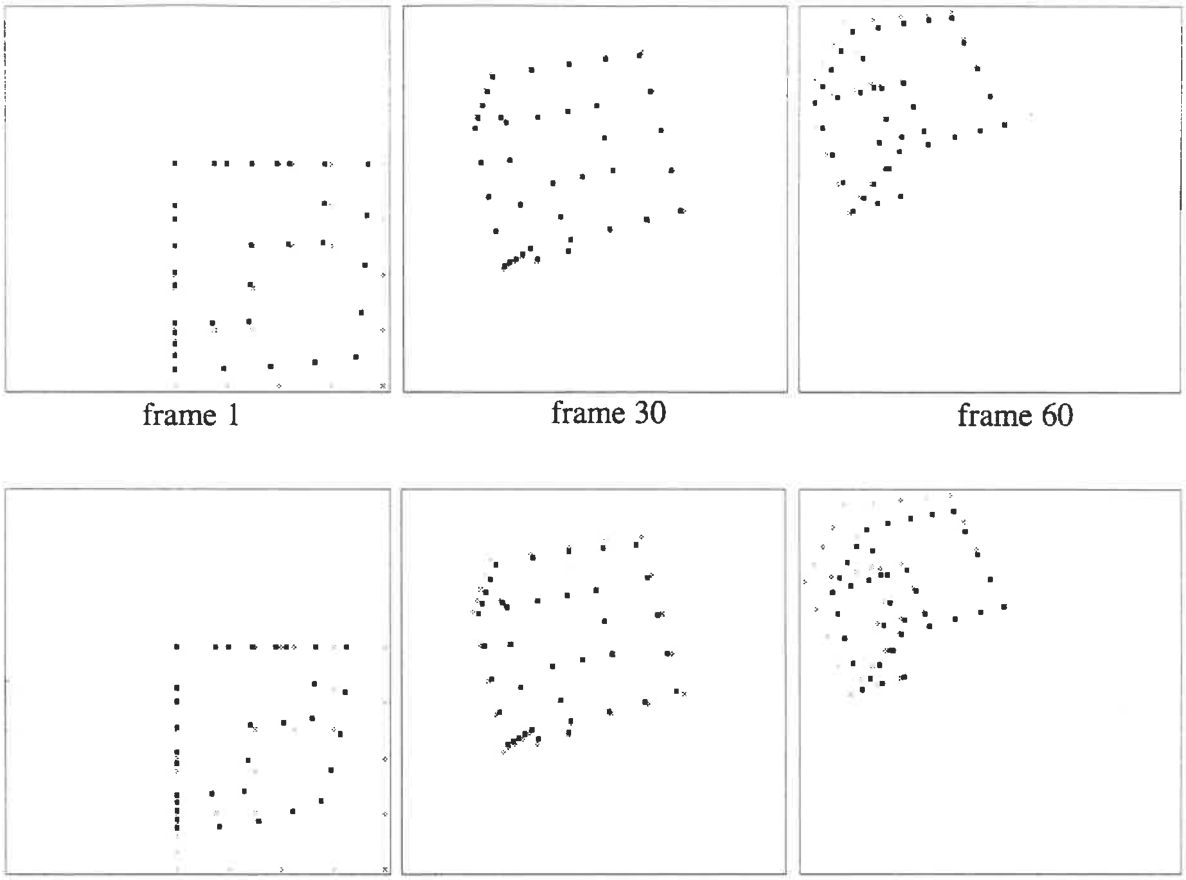
$$\frac{\partial}{\partial o_x}P_p(f,p) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \frac{\partial}{\partial o_y}P_p(f,p) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

In the Euclidean case, no normalization or post-processing is necessary once convergence is achieved. The final rotation, translation, and shape variables values are the answer, with the exception of scaling the normalized depth values d_f by the final focal length μ to compute that actual depth translation z_f.

4.3. Shape and Motion Recovery under Radial Projection

Telephoto lenses used for viewing distant scenes are generally very well modeled by the pinhole camera model used in the standard perspective projection formulation. However, when viewing scenes much closer to the camera, generally lenses with much shorter focal lengths and wider fields of view are used. These lenses generally display lens distortion effects not modeled by the pinhole camera model. The most visible of these unmodeled effects is radial distortion. Radial distortion is caused by the variance in the thickness of the camera lens at the center and the thickness near its edge, which causes points far from the image center to be magnified less than points near the image center. This difference in magnification causes points to be moved towards the center, along the direction of a radius of a circle centered at the image center. Straight lines in the world can be transformed into curves in the image. These effects are illustrated in Figure 28. They are also evidenced in the real images shown in Figure 19, by the curves in the edges of the doorway and refrigerator near the outer portions of the kitchen sequence.

In this section we introduce a model of radial distortion which differs slightly from the pro-



frame 1 frame 30 frame 60

frame 1 frame 30 frame 60

Figure 28. Illustration of Radial Distortion

This figure shows two noise-free versions of the first synthetic sequence shown in Figure 8. The point positions computed using simple perspective projection are shown in gray, while the point positions after radial distortion are shown in black. The upper sequence uses a moderate radial distortion parameter of $\kappa = 0.3$, while the lower sequence uses $\kappa = 1$, simulating the magnitude of radial distortion that might be expected with a very wide-angle, fish-eye lens. Notice the curves added to straight lines, and that the line connecting any undistorted point to its distorted point points to the image center.

jection model used by photogrammetrists and camera calibration researchers, and first show that our model and the standard model are nearly equivalent. We then show how our shape and motion reconstruction formulation can be extended to account for radial distortion by adding a fifth camera parameter which determines the magnitude of the radial distortion in the image.

4.3.1. Standard "2D" Radial Distortion Model

Many camera calibration techniques are based on a radial camera model [41][44][46]. This camera model relates the perspectively projected but non-radially-distorted position of a point (X_u, Y_u) to the distorted position (X_d, Y_d) by the equations [41]

$$\left.\begin{aligned} X_u &= X_d\left(1+\kappa_1\rho^2+\kappa_2\rho^4+\ldots\right) \\ Y_u &= Y_d\left(1+\kappa_1\rho^2+\kappa_2\rho^4+\ldots\right) \end{aligned}\right\} \qquad (128)$$

Here the radial distance $\rho^2 = X_d^2 + Y_d^2$ and the κ_i are the radial distortion coefficients. Tsai reported that using a single-term radial projection model is generally sufficient for machine vision applications [44]. Note that the radial distance ρ, which determines the amount that a particular point is radially distorted, depends on the final distorted position in the image. In calibration applications, the position of a point in the image is generally known, and the model is used to determine the undistorted position of the point. However, in order to determine a projection function $P_{r2D}(f, p)$ which can predict the final distorted position of a point from the motion data for frame f and the shape data for frame p, we must solve the two third order polynomials

$$\left.\begin{aligned} X_u &= X_d\left(1+\kappa_1 X_d^2+\kappa_1 Y_d^2\right) \\ Y_u &= Y_d\left(1+\kappa_1 X_d^2+\kappa_1 Y_d^2\right) \end{aligned}\right\} \qquad (129)$$

for X_d and Y_d as functions of the undistorted position determined by perspective projection (X_u, Y_u), where $X_u = P_{p_x}(f, p)$, $Y_u = P_{p_y}(f, p)$, and unit camera calibration parameters are used in the perspective projection. These equations have exactly one real solution, given by equation (130), which is derived by reducing (129) to a single third degree polynomial in Y_d or X_d.

$$e = \begin{cases} \left(\frac{d}{2}\right)^{1/3}\left(1+\sqrt{1+\frac{4d}{27}}\right)^{1/3} - \frac{\sqrt[3]{2}}{3}d^{2/3}\left(1+\sqrt{1+\frac{4d}{27}}\right)^{-1/3} & d \text{ defined} \\ 1 & d \text{ undefined} \end{cases}$$

$$\begin{aligned} \text{where} \qquad d &= \frac{1}{\kappa_1\left(X_u^2+Y_u^2\right)} \\ Y_d &= Y_u e \\ X_d &= X_u e \end{aligned} \qquad (130)$$

Accounting for other camera and digitization parameters, the projection function becomes

$$p_{r2D}(f, p) = \begin{bmatrix} P_{r2D_x}(f, p) \\ P_{r2D_y}(f, p) \end{bmatrix} = \begin{bmatrix} lX_d + ox \\ laY_d + oy \end{bmatrix} \qquad (131)$$

Although the model has proven quite adequate for photogrammetry and camera calibration problems, we can see that the above formulation suffers from several significant drawbacks when used in 3D vision scenarios where the forward projection must be computed. The expressions are ill-formed when either the radial distortion parameter κ_1 or the distance from the image center $X_u^2 + Y_u^2$ are zero. In these cases e approaches unity, but in practical terms it must be computed as the difference of two large numbers, making it sensitive to truncation and roundoff errors. We expect that in many cases the radial distortion parameter will be very small, since many image sequences contain little or no radial distortion, and some feature points are likely to be located near the center of the image, so this will cause significant numerical problems. Additionally, computation of the distorted position is very complicated, and the expressions for the derivative of the projection function with respect to the motion and shape variables, required for Levenberg-Marquardt minimization, would be hideous.

4.3.2. "3D" Radial Distortion Model

The numerical and computational shortcomings of the standard "2D" model when applied to three-dimensional problems forced us to use a slightly different model, which we will show to be very similar to the standard model but has simpler mathematics for our purposes. Instead of basing the amount of distortion on the distorted image position of a point, we base it on its undistorted image position, which is far more easily calculated from the shape and motion variables.

$$X_u = X_d\left(1 + \kappa_1 X_u^2 + \kappa_1 Y_u^2\right) \qquad Y_u = Y_d\left(1 + \kappa_1 X_u^2 + \kappa_1 Y_u^2\right) \tag{132}$$

The distorted image position of a point is

$$P_r(f, p) = \begin{bmatrix} P_{r_x}(f, p) \\ P_{r_y}(f, p) \end{bmatrix} = \begin{bmatrix} \dfrac{lP_{p_x}(f, p)}{\left(1 + \kappa_1 P_{p_x}(f, p)^2 + \kappa_1 P_{p_y}(f, p)^2\right)} + o_x \\ \dfrac{laP_{p_x}(f, p)}{\left(1 + \kappa_1 P_{p_x}(f, p)^2 + \kappa_1 P_{p_y}(f, p)^2\right)} + o_y \end{bmatrix} \tag{133}$$

Since the camera calibration parameters are applied after the radial distortion, unit camera calibration parameters should be used in the perspective projection. However, to prevent the focal length and depth from trading off against each other, we use $\mu = 1/l$ and $d_f = z_f/l$ as in the perspective case, and the radial distortion parameter κ_1 similarly becomes scaled by μ^2.

Unlike the previous formulation, the forward projection equations for our radial distortion model are perfectly well-behaved when κ_1 approaches zero as well as when the point is located near the image center. Furthermore, the expression is far simpler, and its evaluation requires only multiplication, addition, and division.

We can now solve for shape and motion under a radial distortion model using the same formulation used in the previous sections. The derivative of the projection function with respect to any shape and motion variable υ is computed by applying the chain rule to the derivative of the perspective projection function.

$$\begin{aligned} \frac{\partial}{\partial \upsilon} P_{r_x}(f, p) &= \frac{\left(1 - \kappa_1 X_u^2 + \kappa_1 Y_u^2\right) \frac{\partial X_u}{\partial \upsilon} - 2\kappa_1 Y_u X_u \frac{\partial Y_u}{\partial \upsilon}}{\left(1 + \kappa_1 X_u^2 + \kappa_1 Y_u^2\right)^2} \\ \frac{\partial}{\partial \upsilon} P_{r_y}(f, p) &= a \frac{\left(1 + \kappa_1 X_u^2 - \kappa_1 Y_u^2\right) \frac{\partial Y_u}{\partial \upsilon} - 2\kappa_1 X_u Y_u \frac{\partial X_u}{\partial \upsilon}}{\left(1 + \kappa_1 X_u^2 + \kappa_1 Y_u^2\right)^2} \end{aligned} \tag{134}$$

Here for simplicity we've used $X_u = P_{p_x}(f, p)$ and $Y_u = P_{p_y}(f, p)$, so the $\frac{\partial X_u}{\partial \upsilon}$ and $\frac{\partial Y_u}{\partial \upsilon}$ are the expressions given for perspective projection in equation (98).

The derivatives with respect to the camera parameters are

$$\begin{aligned} &\frac{\partial X_d}{\partial o_x} = 1 \quad \frac{\partial X_d}{\partial o_y} = 0 \quad \frac{\partial X_d}{\partial a} = 0 \quad \frac{\partial X_d}{\partial \kappa_1} = \frac{-\left(2X_u^2 + 2X_u Y_u\right)}{\left(1 + \kappa_1 X_u^2 + \kappa_1 Y_u^2\right)^2} \\ &\frac{\partial Y_d}{\partial o_x} = 0 \quad \frac{\partial Y_d}{\partial o_y} = 1 \quad \frac{\partial Y_d}{\partial a} = \frac{Y_u}{\left(1 + \kappa_1 X_u^2 + \kappa_1 Y_u^2\right)} \quad \frac{\partial Y_d}{\partial \kappa_1} = a \frac{-\left(2Y_u^2 + 2X_u Y_u\right)}{\left(1 + \kappa_1 X_u^2 + \kappa_1 Y_u^2\right)^2} \end{aligned} \tag{135}$$

Note that either the Euclidean or projective formulation can be used for $X_u = P_{p_x}(f, p)$ and $Y_u = P_{p_y}(f, p)$. Even when a projective formulation is used, camera calibration variables must be used to account for the aspect ratio and image center in addition to the radial distortion parameter. These parameters effect the projection in a manner distinct from the projective motion variables, because they are applied after the radial projection part of the projection. When using a projective formulation, the focal length variable is not used because it would trade off directly with the scaling factors of the projective motion matrices and the radial distortion parameter κ_1. When radial distortion effects are small, the aspect ratio and image center will trade off freely with other scaling and translation aspects of the projective motion matrices.

It may seem unusual to use camera calibration parameters in addition to a projective formulation since projective formulations are generally used specifically to avoid modeling camera calibration parameters. However, as will be shown in the experimental section, a formulation in terms of projective shape and motion variables has far better convergence characteristics than a Euclidean formulation. Therefore using a projective formulation of perspective projection followed by a radial distortion step allows our method to model radial distortion and retain the beneficial convergence characteristics of the projective formulation.

Note that the value of κ used in equation (133) alone does not give any indication of the overall magnitude of radial distortion likely to be seen in an image produced by the lens unless the camera's field of view is also known. For example, if a 2 unit wide object is 10 units from the camera and it completely fills the field of view of the lens, then a value of $\kappa = 1$ will cause a point at the center of the left edge of the image to move ~1% closer to the image center. The value of P_{p_x} at the left edge of the image will be 0.1, so the distortion factor $1/\left(1+\kappa\left(P_{p_x}^2+P_{p_y}^2\right)\right)$ will be $1/1.01$ or about 1%. However, if a 2 unit wide object only half fills the field of view, then points at the left edge of the image will have $P_{p_x} = 0.2$, so the point's position will deform by $1/1.04$ or about 4%. The second lens will produce images with substantially more radial distortion, yet both of them have $\kappa = 1$. Therefore whenever we refer to a radial distortion parameter value we will give the value for the *normalized radial distortion parameter*, which is just κ times the square of the field of view in pixels. Using this definition, the amount of radial distortion visible in an image can be characterized by a single number. So for example, for any lens with a normalized radial distortion parameter value $\kappa = 1$, the center point of the left edge will be distorted by $1/(1+1(0.5)^2) = 0.8$ or a reduction of 20%.

4.3.3. Depth-shape Formulation

3D radial projection is modeled by much simpler forward projection equations than 2D radial projection. However, in order to apply the shape-depth formulation to this projection model, the inverse projection model needs to be solved. That is, given a point's image position in the first image, its depth, the estimated motion in the first image, and calibration parameters, we need to be able to compute its three-dimensional position, in order to predict where the point will appear in the other images.

Solving the inverse projection problem for our 3D radial projection model is very similar to solving the forward projection problem for the 2D radial projection model. We solve the radial projection equations for (X_u, Y_u), given (X_d, Y_d) and the camera calibration parameters. We can then solve for s_{p1} and s_{p2}, the x- and y-components of the shape vectors, from (X_u, Y_u) using the Euclidean or projective equations.

$$d = \kappa_1\left((X_d - o_x)^2 + \left(\frac{Y_d - o_y}{a}\right)^2\right) \qquad e\,. = \begin{cases} \dfrac{1-\sqrt{1-4d}}{2d} & d \neq 0 \\ 1 & d = 0 \end{cases}$$
$$Y_u = \left(\frac{Y_d - o_y}{a}\right)e$$
$$X_u = (X_d - o_x)\,e \tag{136}$$

In the case of points near the center of the image or of zero radial distortion, this expression involves the ratio of two numbers whose values go to zero, but this is far more numerically stable than the difference of two numbers approaching infinity. Furthermore these equations are only used for fixed, measured values of X_d and Y_d (the measured positions of points in the first frame), so we never need to compute derivatives of these expressions.

4.3.4. Comparison of 3D and 2D Radial Distortion Models

Figure 29 is a plot of the number of pixels by which a feature point is distorted from its projectively projected position for a typical radial distortion parameter. The distortion parameters were chosen so points at the edge of a 512×512 image would be distorted by approximately 30 pixels, which is a rather high level of radial distortion. At this distortion level, the two radial distortion models differ by only a few pixels. The bottom curve shows that the two models can be brought into even closer alignment by adjusting the radial distortion constant used in the 3D radial distortion model. This shows that the 3D radial distortion model closely approximates the standard model.

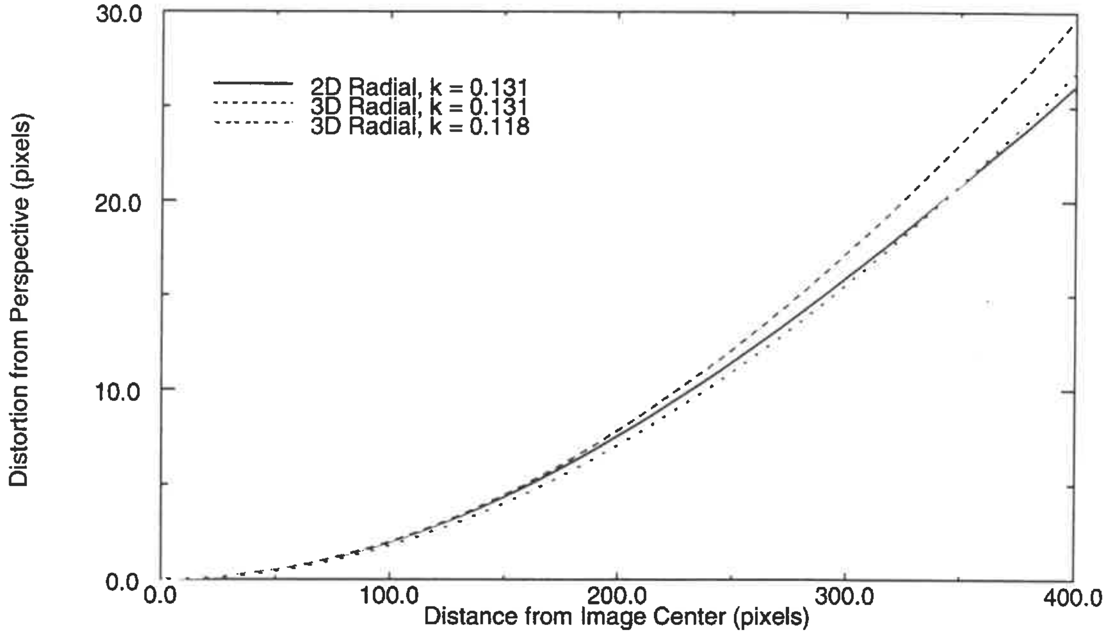


Figure 29. Comparison of distortion models.

4.4. Performance Analysis and Discussion

4.4.1. Analysis of Projective and Euclidean techniques

We performed extensive testing using synthetically-generated data to quantitatively analyze the performance of these non-linear methods as functions of the camera motion and noise level. The methodology of the synthetic data generation and error measurement is the same as that of section 2.5. Using synthetic data, we were able to observe the methods' performances at a variety of distances and noise levels.

The initial motion used for all of these experiments consisted of assigning each frame the same camera orientation and position, i.e. assuming no camera motion at all. The initial object shape consisted of placing each feature point on a plane located far from the camera, with its position on that plane given by the position of the point in the first image.

4.4.1.1. Euclidean Optimization versus Projective Factorization

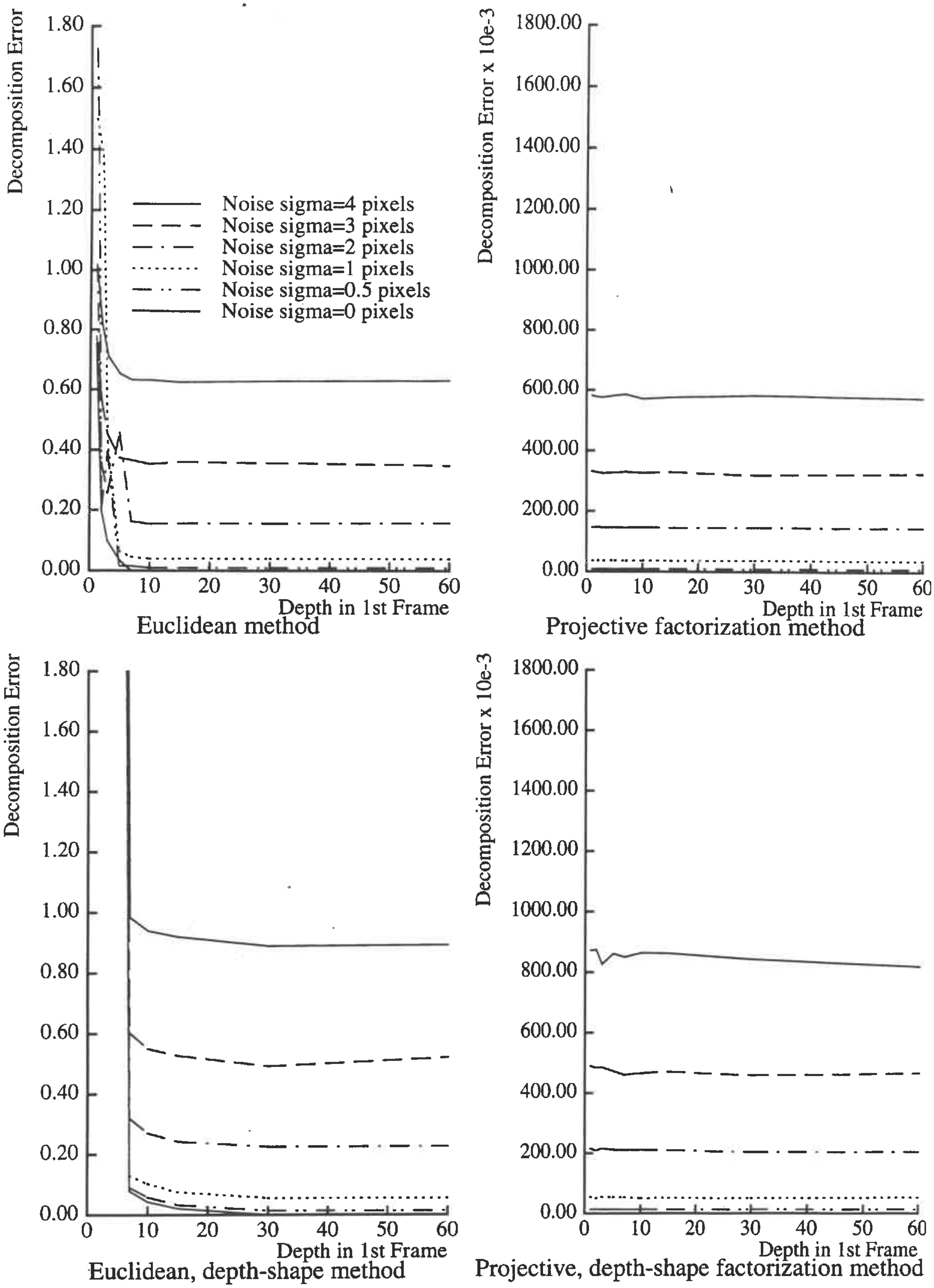
Figure 30 compares the total sum of squares error ε between (u_{fp}, v_{fp}) and $P_p(f, p)$ for the Euclidean optimization method and the projective factorization methods. When the object is far from the camera, both methods are able to minimize the error correctly, without becoming trapped in local minima. However, when the object is near the camera, the Euclidean method often converges to an incorrect result.

The actual shape and motion errors of the Euclidean method are shown in Figure 31. For these experiments, the camera parameters were kept fixed at the actual values used to generate the synthetic data. When the method succeeds, the errors are lower than for the factorization method and virtually invariant to object depth. However, when the object is very close to the camera, the method fails to converge to the proper minimum. Qualitative analysis of the results in these cases demonstrated that the recovered object generally displayed noticeable projective deformations and huge scaling of the shape in the depth axis.

This behavior can be understood by considering the initial value we have chosen. The object is initialized as being very far away, so that perspective effects are minimal. The most rapid way for the method to decrease the total sum of squares error is to deform the object shape by some projective transformation so that it matches the observations more closely. The method hypothesizes that the part of the object that looks larger actually is larger, rather than hypothesizing that that part of the object is closer to the camera. This causes a substantial reduction in the total sum of squares error, but it unfortunately leads the solution towards a highly distorted shape rather than the correct shape. The Euclidean optimization method evidently becomes "trapped" in a local minimum, stuck with an elongated and projectively deformed shape, such that any change to the motion or object shape increases the error rather than decreases it. It cannot recover to reach the true minimum.

The projective factorization method in fact exhibits strikingly similar behavior; the shape is initially improved by producing a shape with projective deformations which is highly elongated in the depth direction. However, this is not such a problem for the projective method, because the final answer is allowed to contain an arbitrary projective deformation, which includes arbitrary scaling of axes. The method does not need to remove the projective deformations and asymmetric scaling factors that were introduced to decrease the error as quickly as possible in the early iterations, it merely has to fine-tune the other parameters to obtain a consistent solution.

This suggests that if the initial value is closer to the correct solution, so that large scaling factors and projective deformations do not have to be introduced into the shape to achieve rapid error reduction, the Euclidean optimization method may perform better. We investigated positioning the "flat" initial shape much closer to the camera so that the initial sce-



Euclidean method

Projective factorization method

Euclidean, depth-shape method

Projective, depth-shape factorization method

Figure 30. Total sum-of-squares error ε

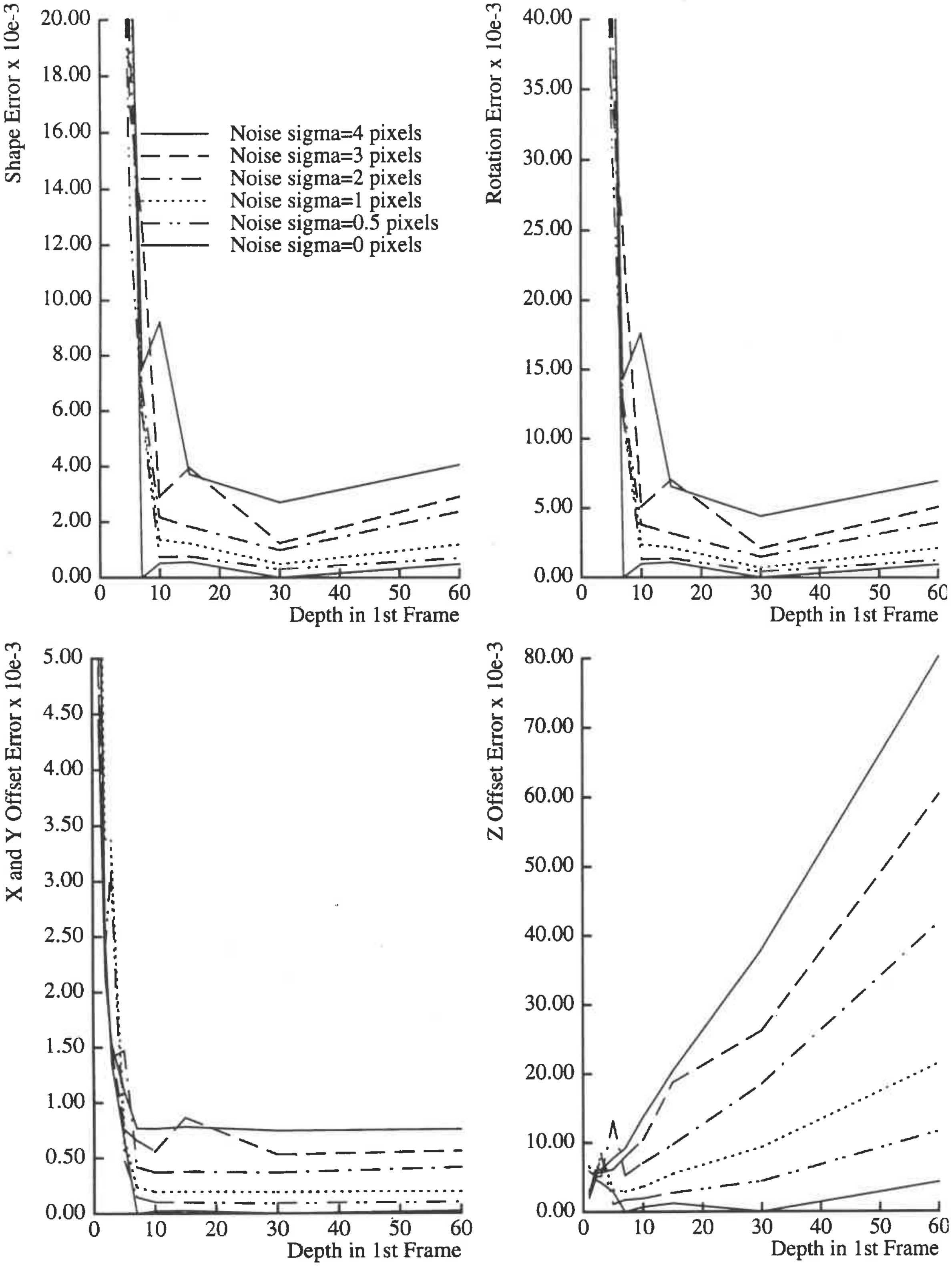


Figure 31. Shape and Motion Recovery Error for Euclidean Method

nario more closely matched the actual scenario. In fact, we supplied the exact distance from the camera to the object in the first frame, and positioned the "flat" object at this correct depth. This approach failed for slightly different reasons. When the object is very close to the camera, relatively small changes to shape or motion parameters can cause a feature point to move behind the camera, which causes the hypothesized shape to be rejected. When this occurs repeatedly, the λ parameter of the Levenberg-Marquardt minimization is increased, and the method is forced to take only tiny steps towards the minimum. In our experiments, using this "closer initialization" approach the minimization failed to converge to the correct answer after 100 iterations. It appears that for close objects, the Euclidean method requires even more accurate initial values than were provided here.

4.4.1.2. Analysis of Projective Normalization for Distant Objects

We show the errors in the shape and motion given by the projective factorization method using three different types of normalization. Figure 32 shows the results using the projective factorization method for $\sigma = 1$, i.e. without using the weighting method described in section 4.1.5. For this method, the results are good for very close objects, in which significant foreshortening is observable. Unfortunately the accuracy decreases very rapidly as the distance to the object is increased. Since the sum of squares error in the decomposition step remains independent of depth, this suggests that the problem is in the normalization step of the method rather than in the projective decomposition.

Using the method described in section 4.1.5. to automatically estimate the weighting value σ to use for the constraints involving $\mathbf{k}_f$ produces the results shown in Figure 33. While the accuracy still degrades as a function of distance, it degrades much more gradually. Particularly, for noise levels as high as 3 pixels it behaves well up to depths of 10 or more. While an ideal method would recover shape and motion accurately in both near and distant scenarios, this performance is satisfactory since above that range, paraperspective factorization or the Euclidean optimization method work well.

4.4.1.3. Analysis of Depth-shape Formulation

Figure 34 shows the results of the projective factorization method using the shape-depth formulation, whereby èach unknown point is represented by a single variable, the depth of the point in object-center coordinates. This method requires substantially less computation due to the reduced number of variables. The performance is slightly worse at low noise levels and low depths, but appears slightly better at high noise levels and high depths. Its worse performance can be explained by the fact that it assumes the point positions in the first frame to be perfectly correct, when in fact our simulations added noise to those values as well.

4.4.1.4. Analysis of Radial Distortion Method

Figure 35 demonstrates the results of projective factorization applied to radially distorted

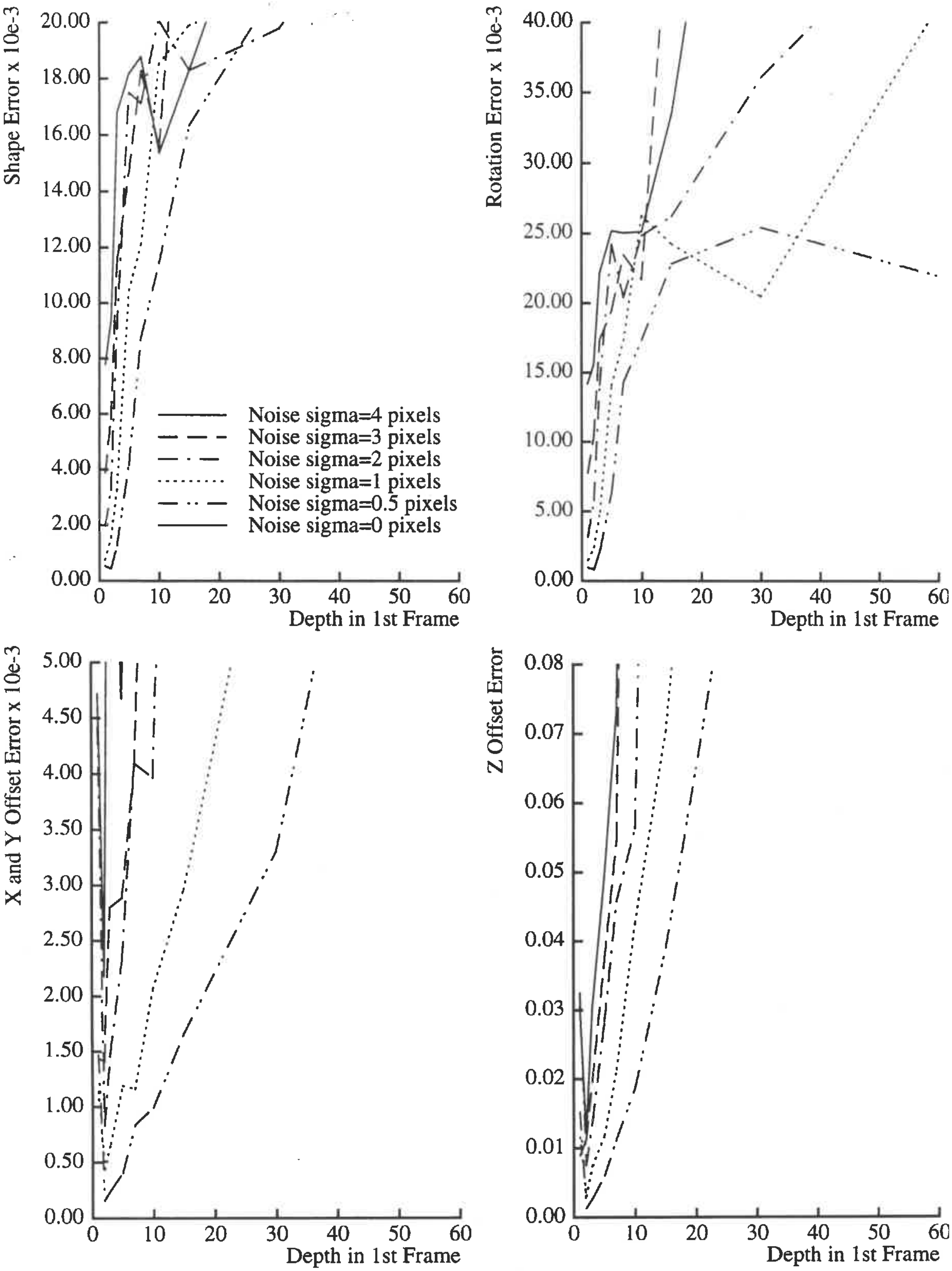


Figure 32. Projective Factorization Shape and Motion Recovery Error,
$\sigma = 1$

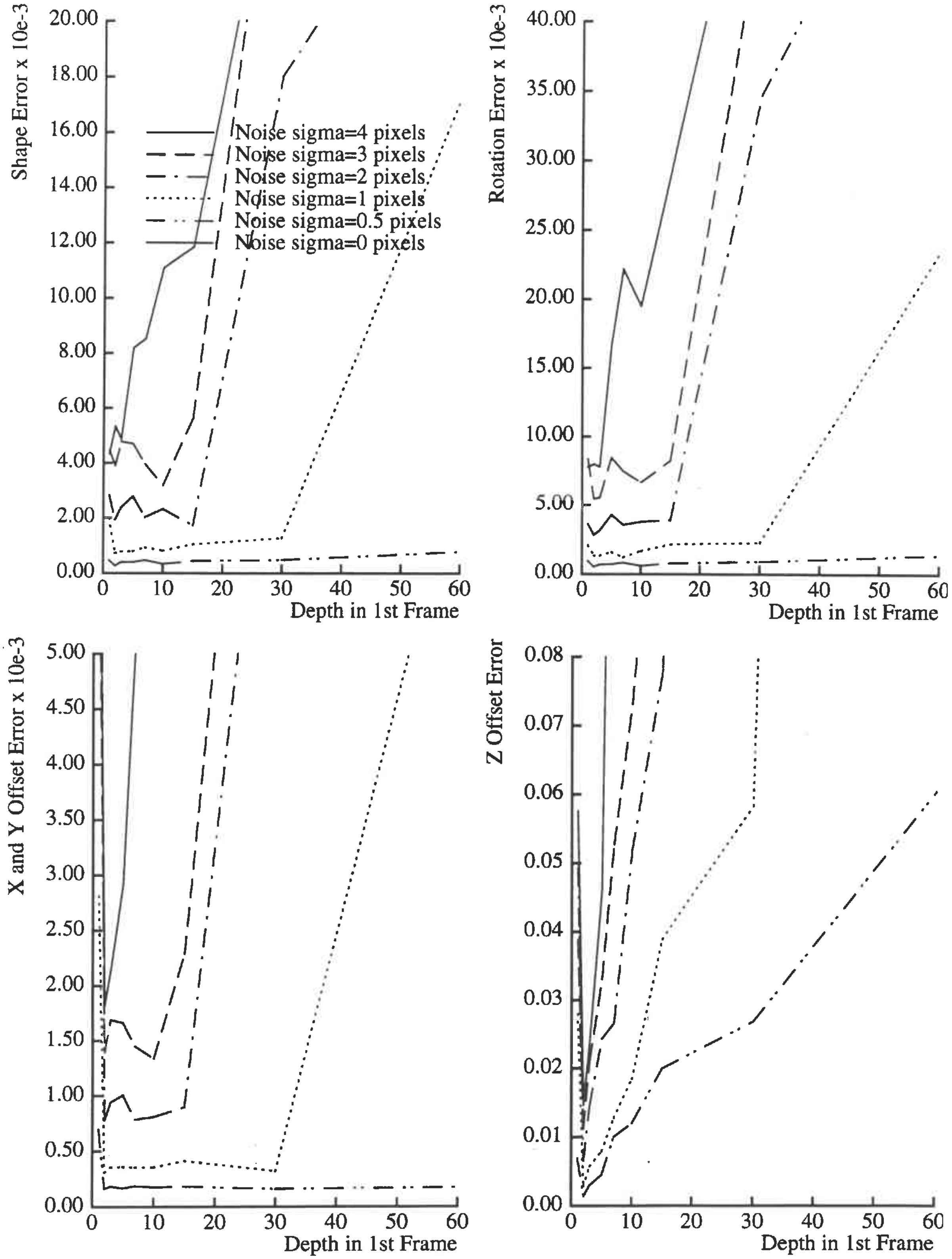


Figure 33. Projective Factorization Shape and Motion Recovery Error, Variable σ

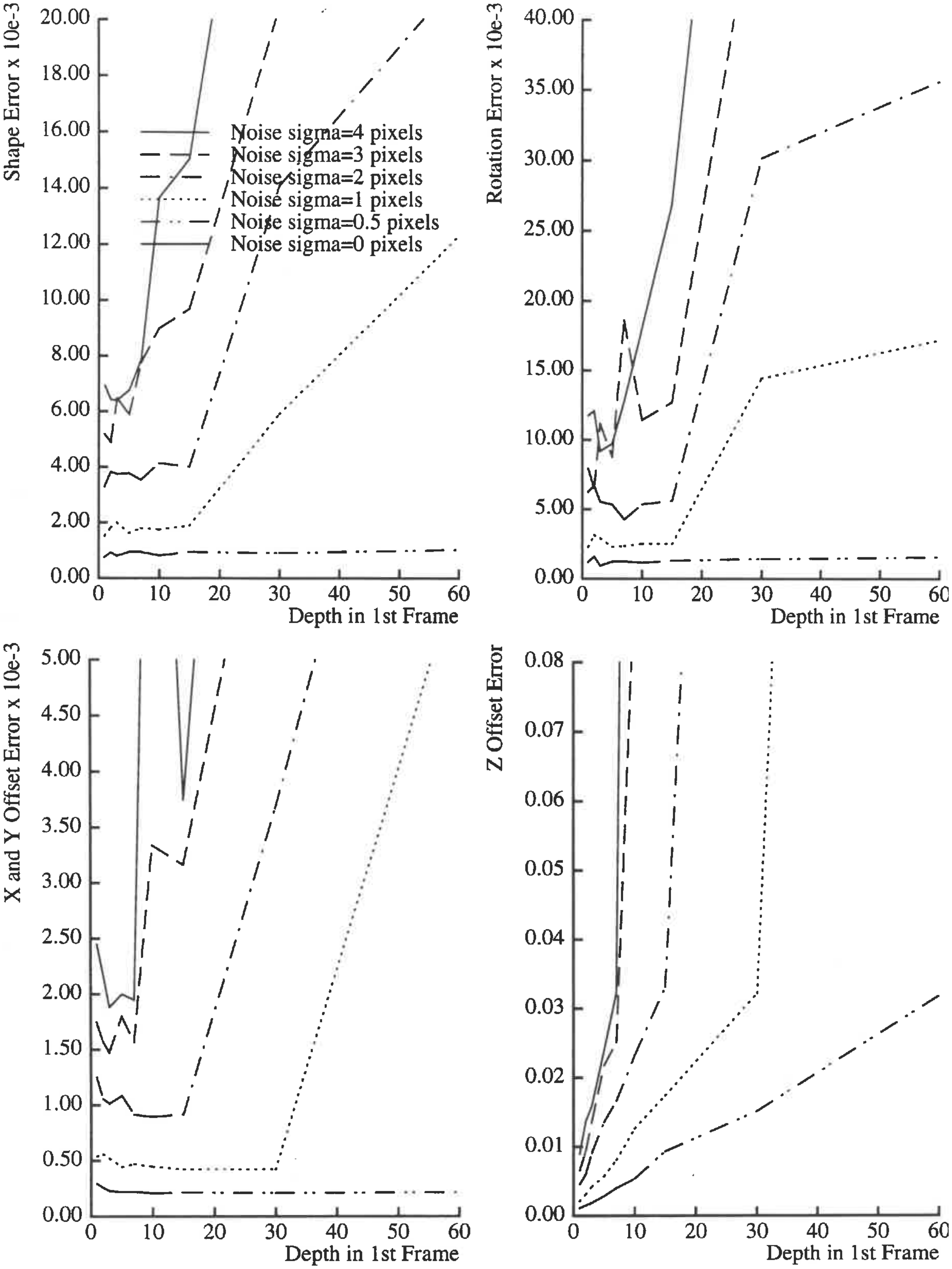


Figure 34. Projective Shape and Motion Recovery Error
Variable σ, Depth-shape Formulation

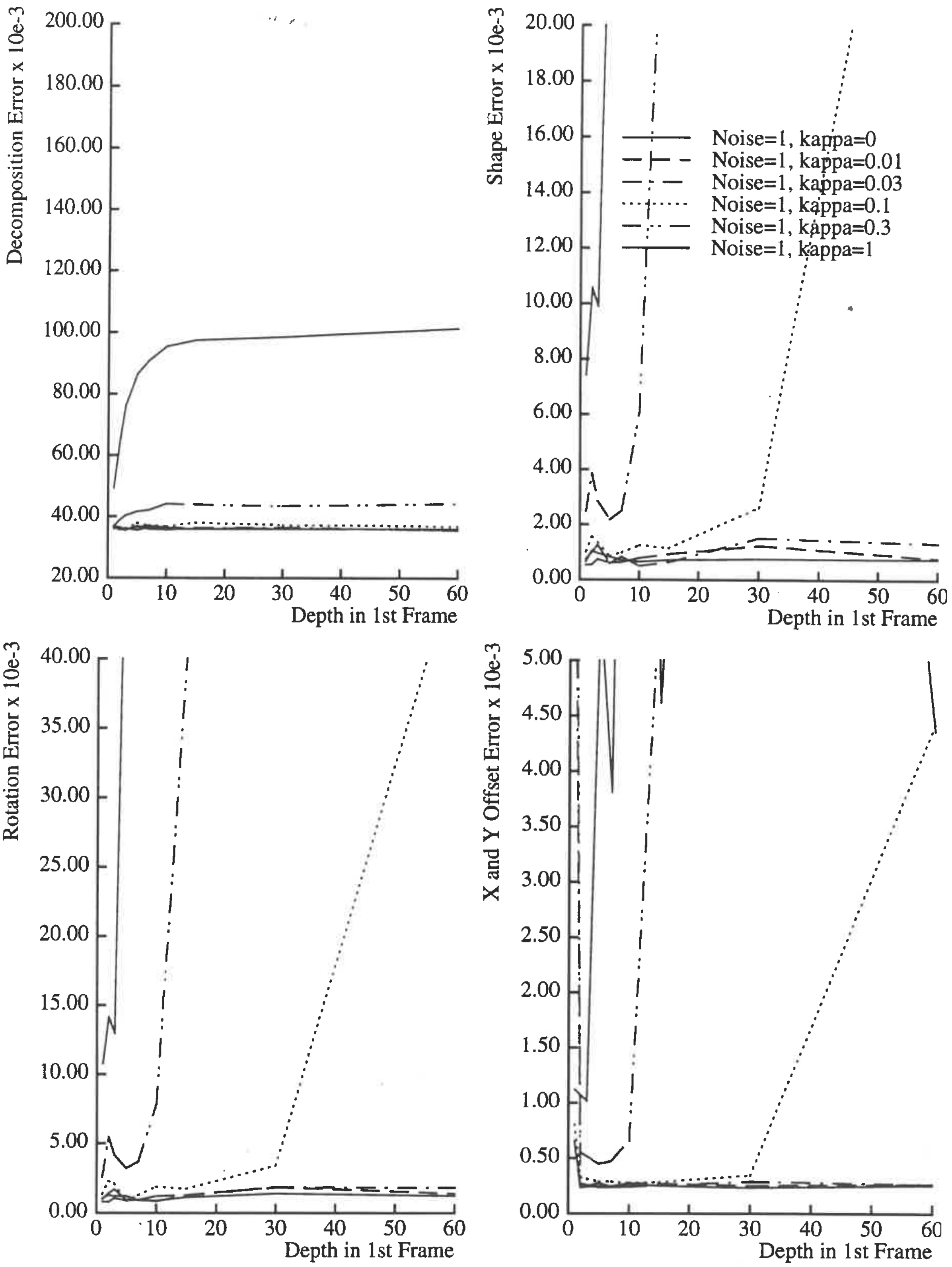


Figure 35. Non-radial Projective Factorization Method Applied to Radially Distorted Images

images when such distortion is not explicitly modeled. Lenses with high radial distortion causes a substantial increase in the errors of the recovered structure and motion. The reason that the decomposition error increases with distance is not an accurate reflection of the behavior of the decomposition method. Rather, it is a reflection of the fact that when an object is very close to the camera, foreshortening effects push a larger portion of its 3D points towards the center of the image, causing a net reduction in the overall magnitude of radial distortion in the images. In real image sequences, however, we expect that an object's closeness to the camera will have no effect on the distribution of feature points in the image and the total decomposition error will remain independent of depth regardless of the amount of radial distortion present in the sequence.

Figure 36 shows the results the projective factorization method when radial distortion is explicitly modeled as described in the previous section. The radial distortion parameter κ is given an initial value of zero, and is allowed to vary simultaneously with the projective shape and motion parameters. The aspect ratio and image center were kept fixed at their known correct values during this process. We first performed several iterations of the Levenberg-Marquardt optimization holding κ fixed at zero, and only when this optimization seemed to converge was κ allowed to vary. When all were allowed to vary from the outset, often κ would move to some large or negative value to try to achieve rapid reduction in the error, but eventually led to a local minimum. Figure 36 show that the error in the shape and motion recovery is virtually independent of κ, indicating that we have successfully modeled the radial distortion.

We performed these experiments for several noise values, although only the results for 1 pixel noise case are shown. When no noise was added to the images, the method dependably produced the correct answer with zero error. With a noise level of 2 pixels, the errors were still independent of the radial distortion parameter and resembled the graphs in which the images were perspectively projected as shown in Figure 34.

4.4.1.5. Analysis of Occlusion Handling

Figure 37 shows the performance of the projective method when occlusions or missing data are present in the image sequence. The fill pattern for a give fill fraction was created in the manner described in 2.5.1. The behavior is very similar to that for the confidence weight paraperspective factorization method. Initially solution accuracy decreases slowly as the fill fraction decreases. However, as the fill fraction drops below 0.5 or 0.6, the errors increase dramatically. Below these fill fractions, the solution generally is not even qualitatively correct, and the feature point positions predicted by the final solution are very far from the observed position, indicating that the iterative method has apparently become stuck in a local minimum.

Interestingly, the separate refinement method discussed in section 2.4 essentially amounts to using only the block diagonal portion of the matrix. When the off-diagonal elements are ignored, the method is unable to modify both shape and motion to pull various parts of the

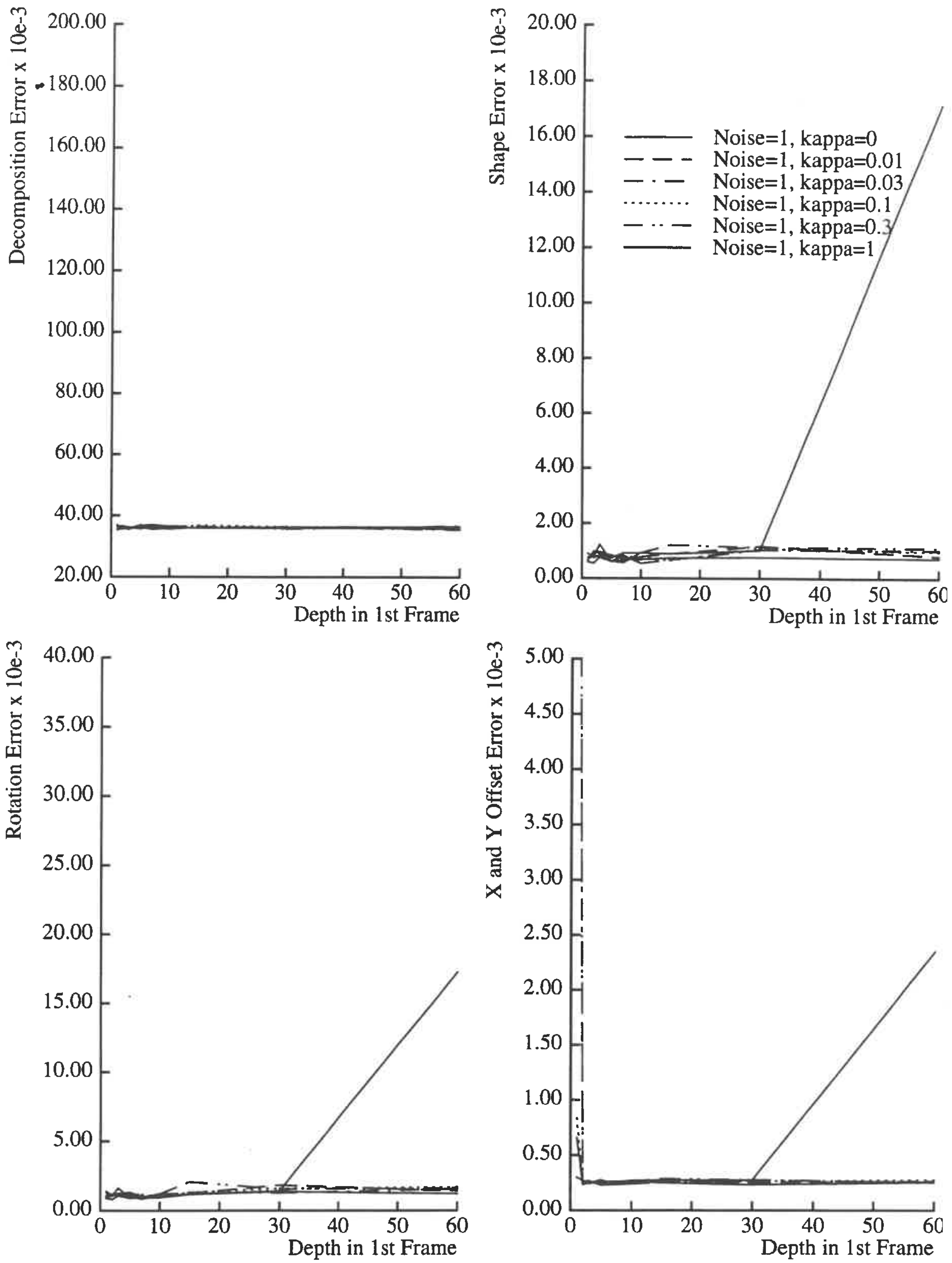


Figure 36. Projective Factorization with Variable Radial Distortion

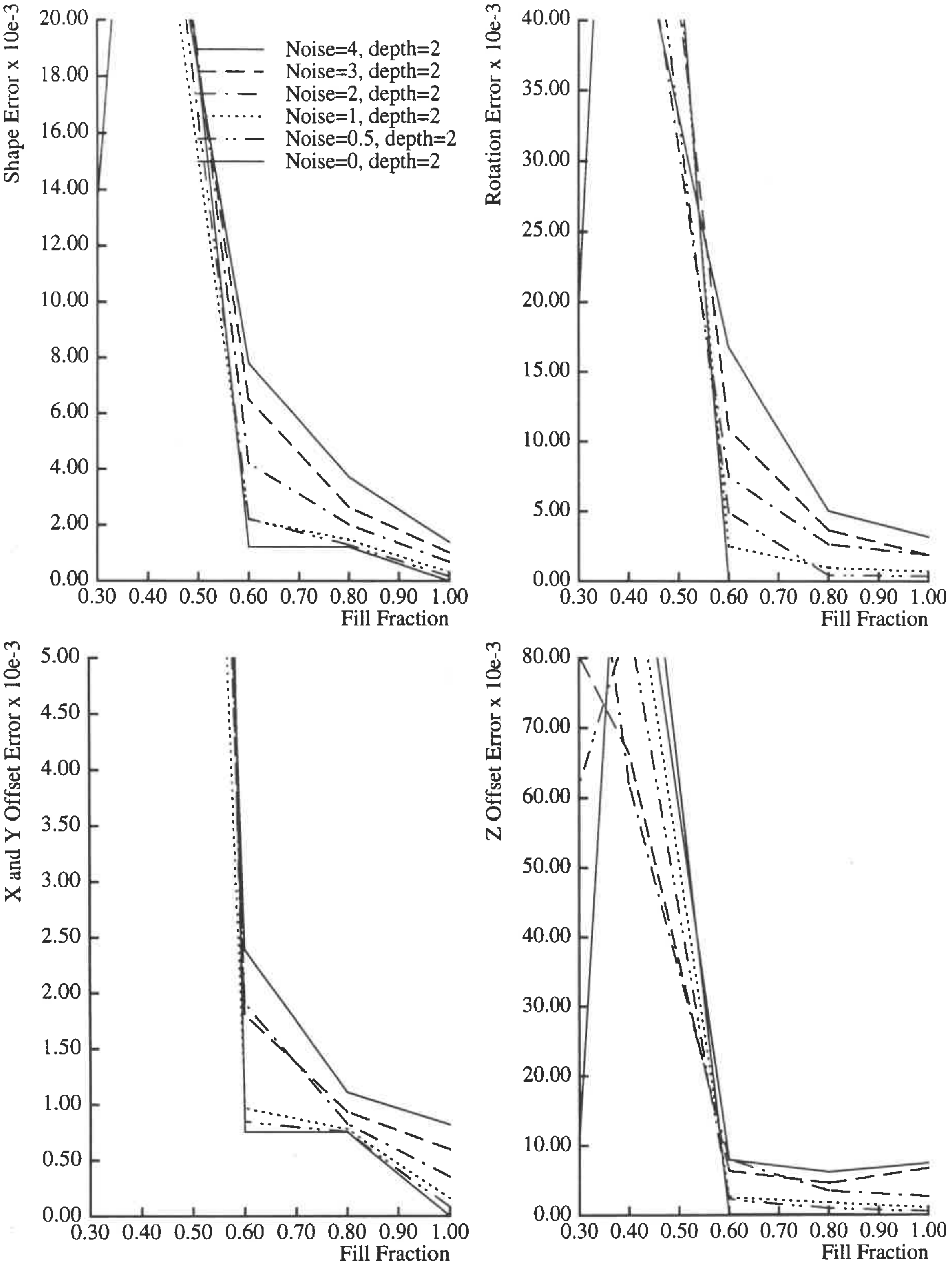


Figure 37. Projective Shape and Motion Recovery with Occlusion

solution together. The full Levenberg-Marquardt method can modify both shape and motion simultaneously. Its use of a quadratic model of the error surface enables it to converge quickly to the minimum after approaching the minimum by gradient descent.

4.4.1.6. Failure of Auto-Camera Calibration

As was shown in section 4.2, the non-linear Euclidean optimization method is theoretically able to refine camera parameters simultaneously with the recovery of shape and motion. We did not fully explore the usefulness of this technique after some initial experiments proved disappointing. If camera calibration parameters were allowed to vary throughout the optimization, invariably even if given the correct calibration parameters as an initial value, the first step of the optimization changed them drastically, often to absurd solutions such as aspect ratios which differed from unity by orders of magnitude, negative focal lengths, and negative radial distortion parameters. These values were clearly incorrect, but since the errors in the initial value overall were large because of our simplistic "flat-plate" initialization, such absurd camera parameters could for the time being reduce the total error. Unfortunately such camera parameters also often led the optimization method inescapably into local minima.

Our next approach was to keep the camera parameters constant until the shape and motion had been substantially refined. When the initial camera parameters were very close to correct, the final refinement did produce the correct camera parameters as well as the correct shape and motion. However, if the initial camera parameters were incorrect, then often the shape and motion recovered while keeping the camera parameters fixed at their initial values would produce projectively or radially deformed shapes. Even once the camera parameters were allowed to vary, the solution remained trapped in a minimum with incorrect camera parameters and a deformed object shape. Depending on the noise level of the images and other effects, this sometimes happened with errors in the initial camera estimates as small as 20%. Perhaps a method which limits the change in the camera parameters at first, and then gradually allows them to change by larger steps over the course of the optimization would be able to accommodate less correct initial camera parameter estimates.

When only the radial distortion parameter is varied, and the aspect ratio and image center are fixed to their correct values, the projective factorization method is able to model the radial distortion in the images and recover a reasonably accurate radial distortion parameter. (Again, for these experiments we first held $\kappa = 0$ and refined the projective shape and motion, and then allowed κ to vary.) Further research should investigate whether other parameters can be recovered in isolation when the remaining parameters are fixed. Since aspect ratio and image center are generally easy to roughly guess, one approach might be to recover the focal length and radial distortion parameter first while keeping the aspect ratio and image center fixed, and then to refine them all as a last step.

4.4.2. Outdoor Building Example

A building at Carnegie Mellon University was imaged using a hand-held portable camcorder. Two hundred features were automatically identified in the first image and tracked from image to image. While many of the features were tracked throughout the entire sequence, some features left the field of view or became occluded. When the number of remaining features fell below 160, new features were detected and tracked. After discarding all features that were tracked for fewer than 8 frames, a total of 207 features were used for this 58 image sequence. The images, with tracked features overlaid, and the tracking fill pattern are shown in Figure 38.

The camera was very roughly calibrated using a simple technique. A flat rectangle of known size was placed at a known distance from the camera, and a single image was taken. From the image, the coordinates of the four corners of the rectangle were manually observed. The focal length in pixels was calculated as the product of the distance to the rectangle in meters and the observed width of the rectangle in image pixels, divided by the known width of the rectangle in meters. The center pixel of the digitized image was used as the image center. This very rough technique is simple enough to enable quick calibration in real applications, and seemed to work sufficient well for use with factorization method.

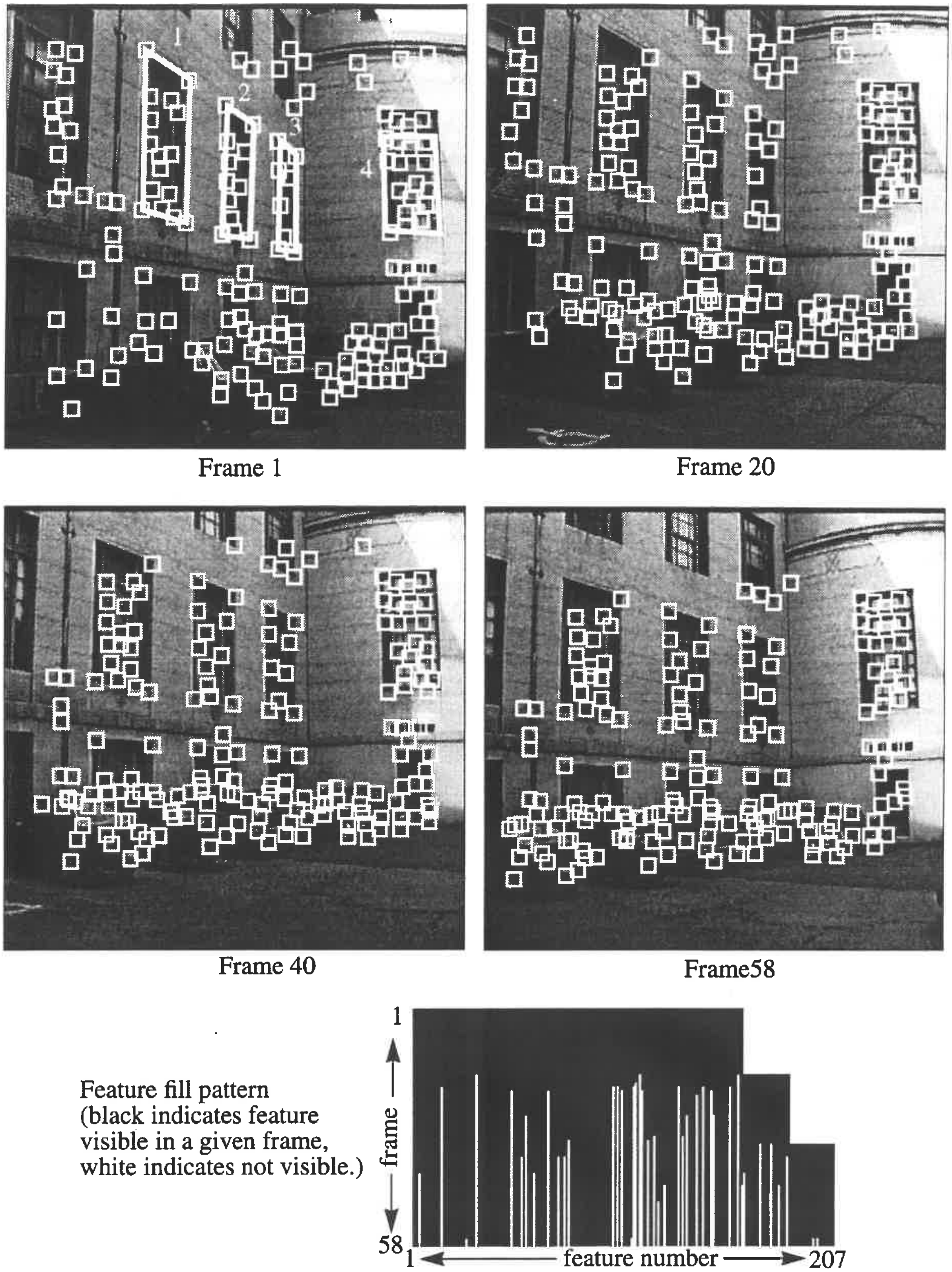
The projective factorization method was used to recover the shape and motion. The projective shape recovered from the factorization step is shown in Figure 39. This shape clearly contains both projective and affine distortions. For example, the object narrows towards the rear, and the wall is at approximately a 60 degree angle to the car rooftops rather than 90 degrees. Furthermore the shape exhibited an extreme scaling of the depth axis, as discussed in the previous section, requiring us to scale the shape's depth axis by a factor of 0.01 before displaying it.

The final shape computed after normalization has correctly removed these artifacts, as shown in Figure 40. The walls have been brought into a roughly 90 degree alignment with the car rooftops, and the windows are now all properly aligned and scaled. Even the position of the upper left corner of the window marked "1" was recovered correctly even though it was lost after the 17th image, in which it briefly left the field of view due to the unsteady motion. Also note that the window at the rear is installed on a wall which curves away from the observer, so its actual angle relative to the other three windows is indeed greater than 90 degrees.

4.4.3. Indoor Kitchen Example

In this example, kitchen was imaged using the same hand-help portable camcorder. The camcorder had a variable zoom, which was set at the widest angle setting in order to view the entire kitchen from fairly close range. At this setting radial distortion effects, as can be seen by noticing the curved appearance of the straight doors at the entrance to the kitchen.

One hundred and twelve features were tracked through the 40 image sequence. Due to relatively large textureless regions in the image, an inter-level weighting constant of 0.5 was used in computing the optical flow estimates. After performing the projective decomposi-



Frame 1

Frame 20

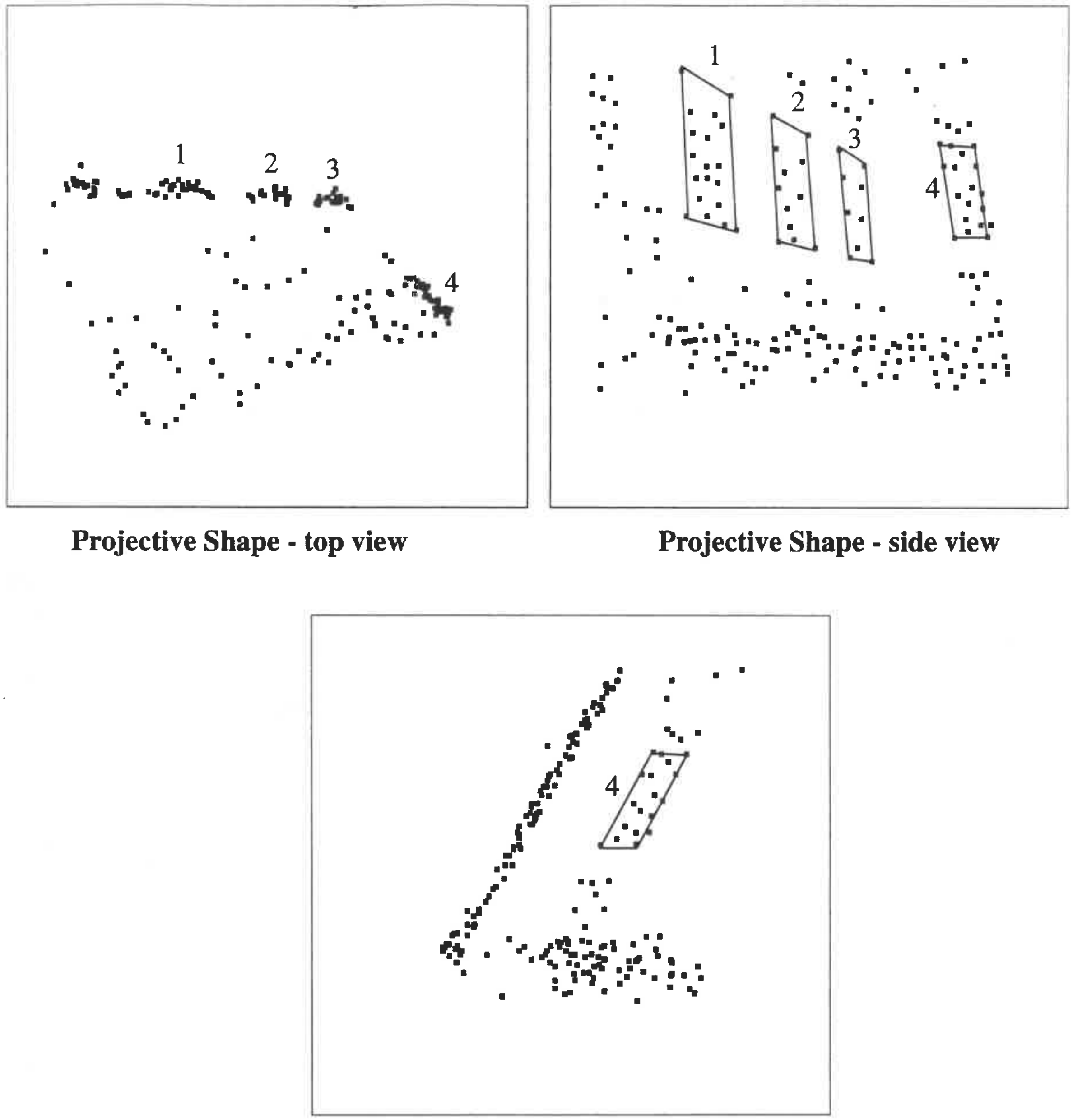
Frame 40

Frame58

Feature fill pattern (black indicates feature visible in a given frame, white indicates not visible.)

Figure 38. Hamburg Hall Sequence and Tracked Features

Four windows have been outlined and numbered manually for later reference.

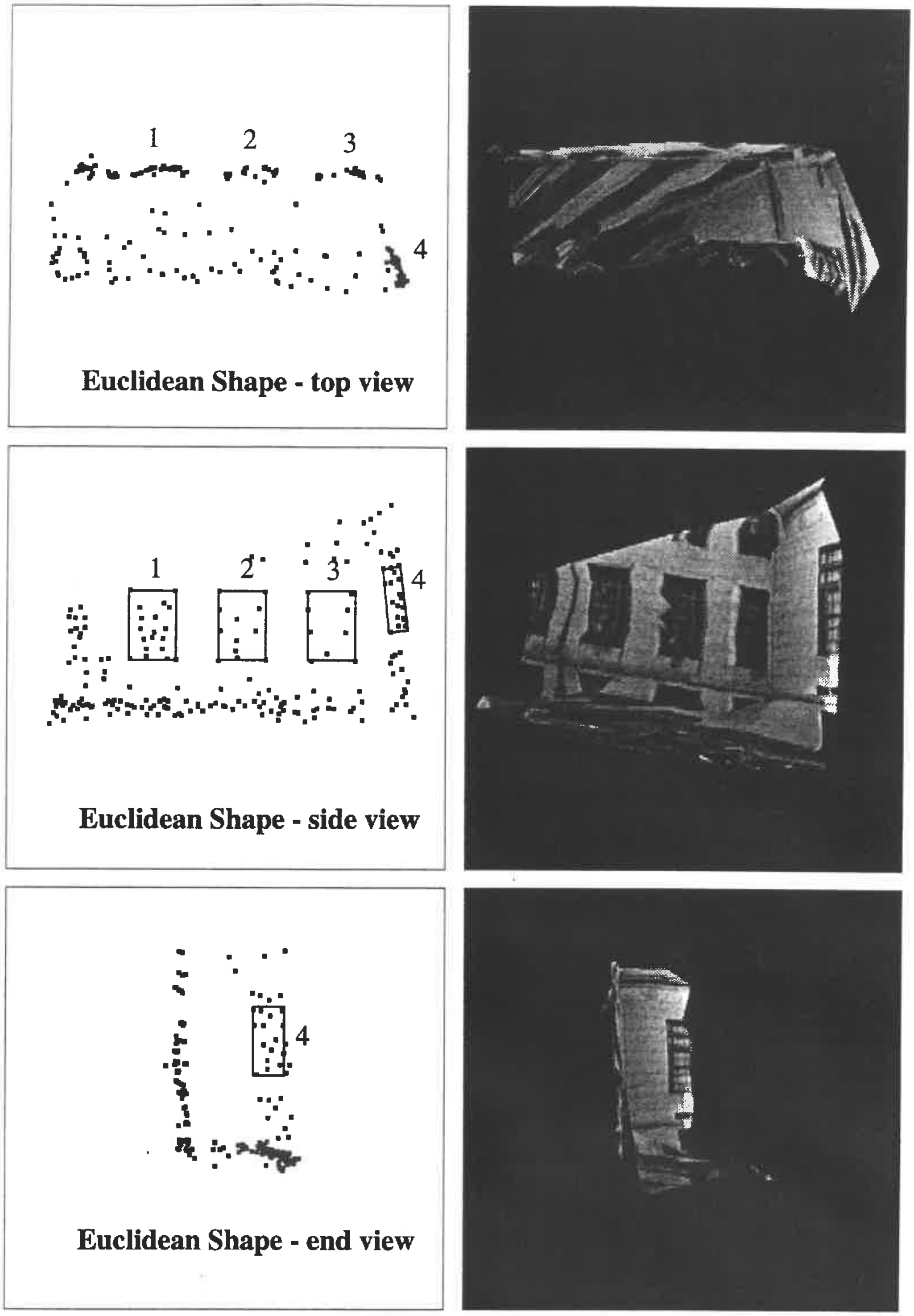


Projective Shape - top view

Projective Shape - side view

Projective Shape - end view

Figure 39. Recovered Projective Shape of Hamburg Hall Sequence
The images shown here are orthographic projections of the projective shape recovered from the decomposition step of the projective factorization method; therefore all *apparent* perspective effects are projective deformations present in the shape itself. The z-axis of the shape has been manually scaled by a factor of 0.01 to facilitate viewing. The four windows have been marked by hand to show that their sizes vary, and their edges are not orthogonal or parallel.

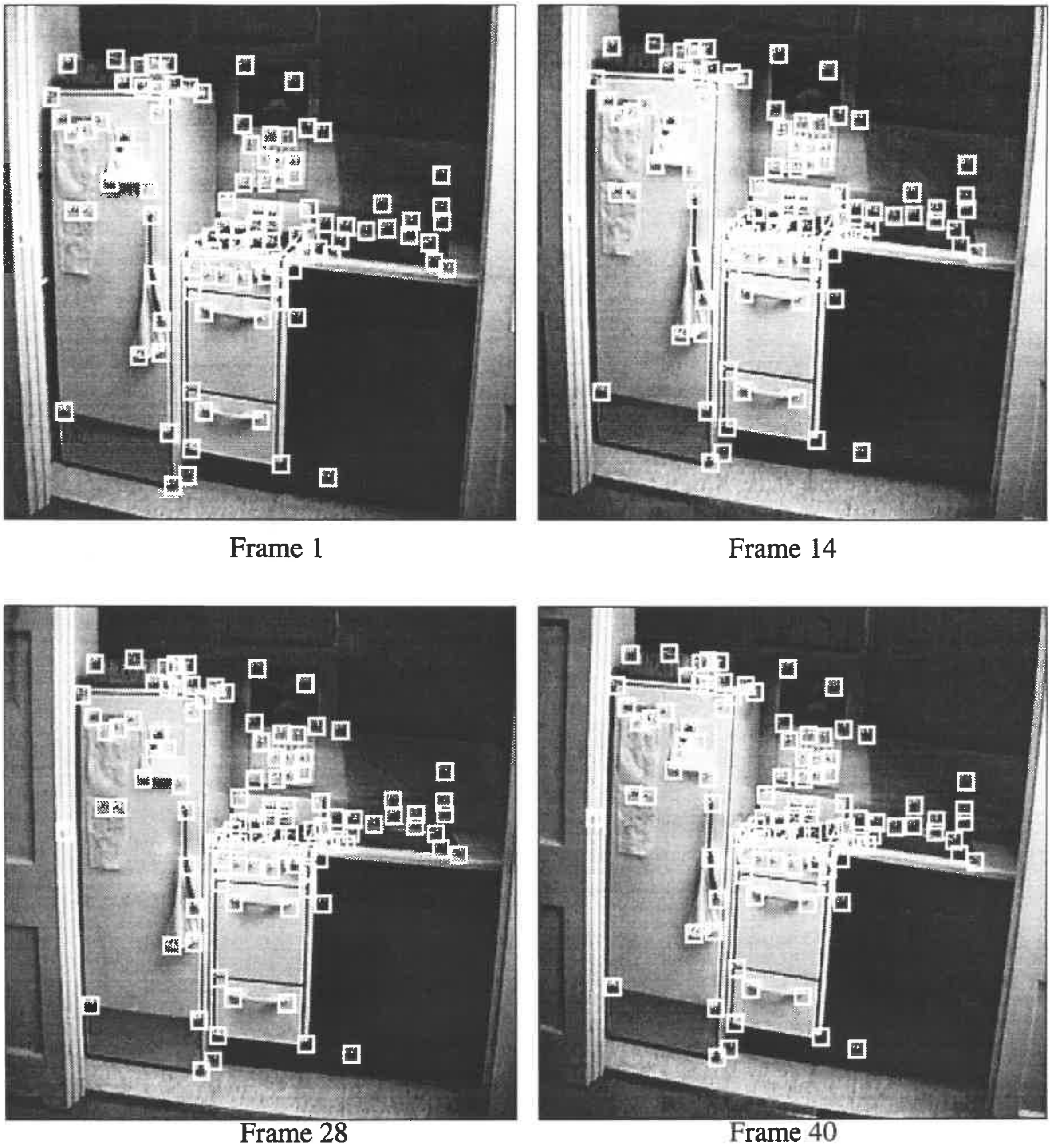


Euclidean Shape - top view

Euclidean Shape - side view

Euclidean Shape - end view

Figure 40. Euclidean Shape Recovered from Projective Factorization
The windows have been marked by hand using rectangles of equal size, to demonstrate that the recovered window shapes are equal-sized rectangles. The figures on the right show the results of mapping image texture onto the recovered shape, from approximately the same viewpoints as the images on the left.



Frame 1

Frame 14

Frame 28

Frame 40

Figure 41. Kitchen Image Sequence and Feature Tracking Results

tion, ten poorly tracked points were automatically removed in a manner similar to that described in section 2.7; points showing a large discrepancy between the tracked feature positions and the positions computed by reprojecting the computed projective shape and motion were discarded. The projective shape and motion were then recomputed and projective normalization was applied. to recover the Euclidean shape. Finally, this Euclidean shape was refined using the radial projection model to produce the final shape shown in Figure 42. A surface was mapped onto the shape to more effectively demonstrate the shape reconstruction. This surface was created by computing the Delauney triangulation of the points in the first frame, and was then edited by hand to remove or refine incorrect surfaces such as those spanning large depth discontinuities.

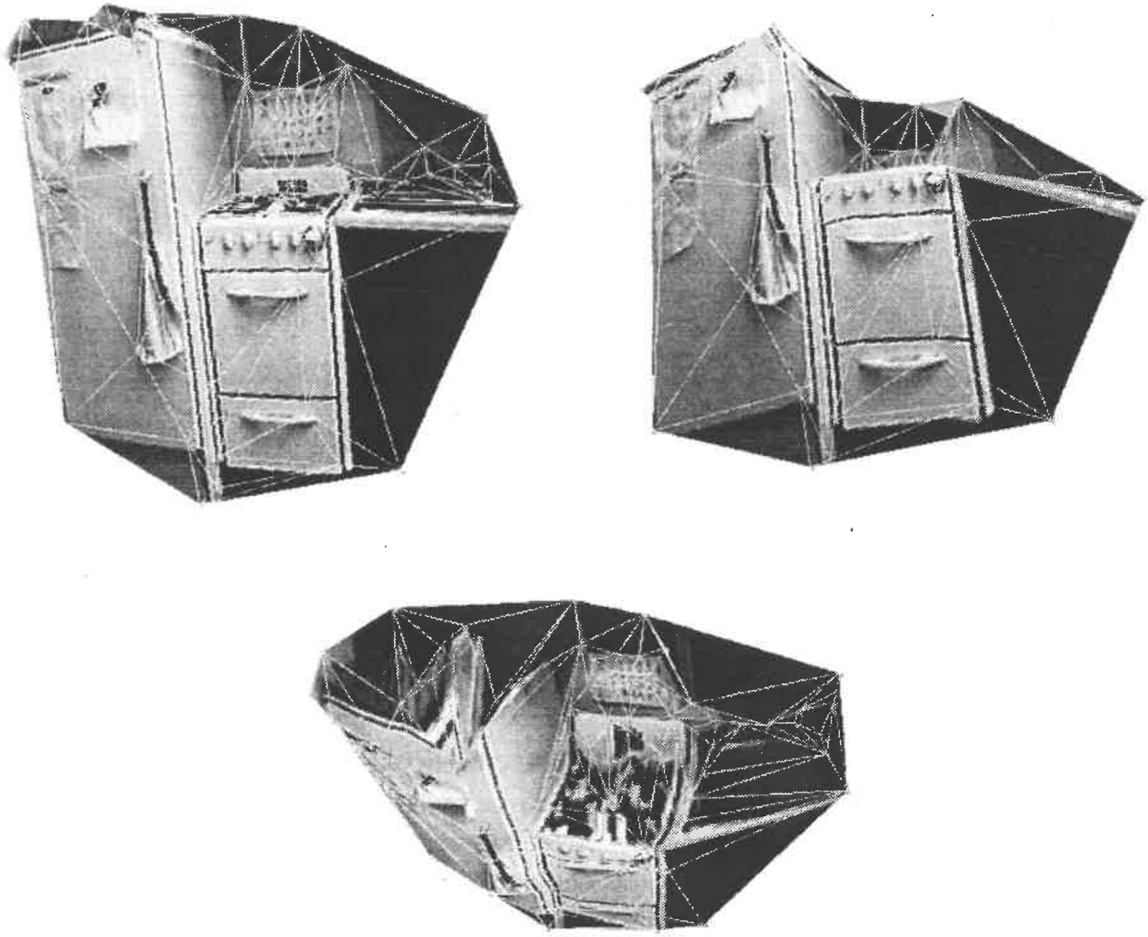


Figure 42. Reconstructed Kitchen Shape

4.5. Direct Shape and Motion Recovery without Tracking

Consider the most common failure modes of typical shape from motion systems. In noisy image sequences, tracking results often have a high error. Certainly sensor noise and quantization effects cause small errors in the feature localization. However, other factors can cause more troublesome problems. For example, a quick motion of the camera can cause the estimated position of a feature to "jump" from the correct region of the image to a nearby region with a similar appearance. Sometimes this may simply be a local minimum; the actual feature position would be a better match, but the tracker has moved the estimated feature position to a new region of the image where it is stuck in a local minimum. Alternatively if the images are noisy or contain repeated texture, both the "true" feature position in the image and the false position may have roughly equal error measurements - there may be no image-level way to differentiate the two positions, as illustrated in Figure 43.

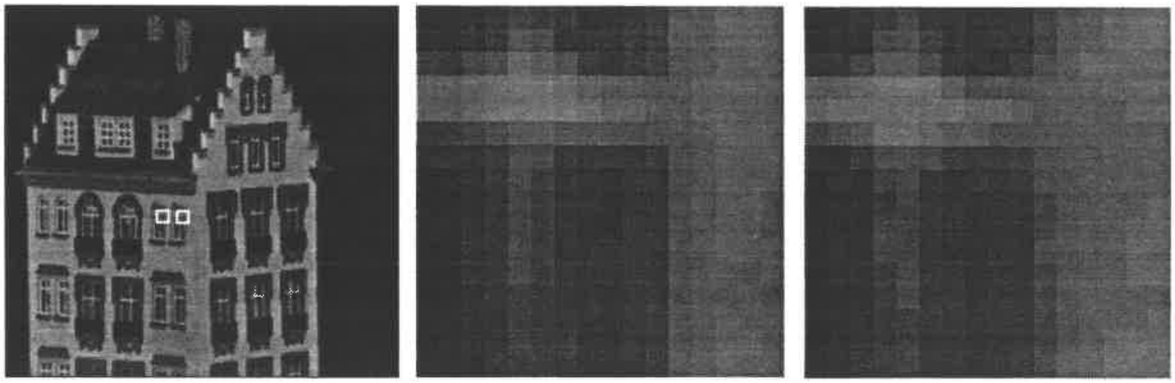


Figure 43. Similarity of Image Features

In the sample image on the left, two nearly identical features are marked. These features are expanded in the center and right images. Because the two windows are so similar, the feature tracker cannot distinguish one from the other.

Of course, when the entire sequence is taken into consideration, the correct feature positions will move in a manner consistent with that of a rigid body, while incorrect feature positions are less likely to move in a rigid manner. We found that in the aerial and kitchen examples that many such improperly-tracked points could detected and removed by careful post-processing. In some cases, such points may cause the shape and motion recovery method to produce an incorrect motion to account for the observed feature motion, making it difficult to assign blame for a poor solution to any one point or set of points. The overall shape and motion recovery error ε will surely be higher if incorrect feature tracks are chosen, but in the tracking stage there is no way to determine this since three-dimensional information is unavailable. Furthermore if such points are simply removed, the recovered shape will be "sparse", since important key points have no recovered shape. This will make it difficult to map a surface onto the recovered 3D points, a post-processing step that is required for many applications.

In this section, we alleviate these problems by presenting a method for structure and motion recovery which incorporates image-level information directly in the shape and motion recovery process. We enforce the rigidity constraint throughout the process by combining the tracking and structure and motion recovery stages into a single hypothesis-based system. Features are detected in the first image just as before. Rather than tracking the features from frame to frame, we formulate the problem as a single minimization problem in which we seek to find the shape and motion which directly minimizes image errors. The advantage of this approach is that solutions which do not correspond to a single rigid object undergoing motion are never considered. The rigidity constraint is applied throughout the process rather than only at the end, and the image data is used directly to compute shape and motion without tracking features as an intermediate step.

Unfortunately, this can also be the method's main disadvantage. The method cannot consider any solutions incompatible with the method's basic model of rigidity, radially projected planar patches whose change in the images can be approximated as purely translational. Of course, most methods state these or similar assumptions, but for a method to be practical it must degrade gracefully when there are a few minor violations of the method's basic assumptions. This method can potentially be more fragile than most, because any error which causes the estimated feature positions to differ from the actual positions by more than a few pixels can lead the method into a local minimum. Therefore we need to be careful to model as many aspects of real projection as possible, and to use robust hierarchical methods to try to avoid the traps of local minima.

We first present the basic least-squares formulation, and then show how we use the basic framework of Levenberg-Marquardt minimization developed in this chapter to find the optimal solution. We then extend the method to a multi-resolution framework in order to enable success even with a poor initial value. We finally show an example of using the method to directly recover the shape and motion without performing feature tracking.

4.5.1. Least Squares Formulation

Consider the method we used to track in image I_f the position of a single feature p, whose image template F_p is defined by the image intensities over the region of interest R (generally a square region of a certain size) in the first image. We solved for the vector $\mathbf{h}_{fp}$ which minimized the error

$$E_{fp} = \sum_{\mathbf{x} \in R} [I_f(\mathbf{x} + \mathbf{h}_{fp}) - F_p(\mathbf{x})]^2 . \qquad (137)$$

In the past, we computed each of the $\mathbf{h}_{fp}$ independently from the others. Now we extend this minimization to simultaneously minimize the error for all points p and all frames f. Thus we minimize

$$\varepsilon = \sum_{f=1}^{F} \sum_{p=1}^{P} \sum_{\mathbf{x} \in R_{fp}} [I_f(\mathbf{x} + \mathbf{h}_{fp}) - F_p(\mathbf{x})]^2 \tag{138}$$

Rather than allowing the $\mathbf{h}_{fp}$ to vary independently, we write each as a function of the unknowns in the problem, the structure and motion variables, using the perspective projection equations, so that

$$\varepsilon_{\text{direct}} = \sum_{f=1}^{F} \sum_{p=1}^{P} \sum_{\mathbf{x} \in R(F_p)} \omega(\mathbf{x})\,(I_f(P(f, p) + \mathbf{x}) - F_p(\mathbf{x}))^2 \tag{139}$$

Here ω is a gaussian defined over the region whose purpose it to emphasize the pixels near the center of the window. Our goal is to find the structure and motion variables which minimize the total error ε. Since our solution is a set of structure and motion values, it automatically satisfies the rigidity constraint.

4.5.2. Levenberg Marquardt Solution

Equation (139) is just a sum of squares error measurement in terms of the shape and motion variables. The number of data points to which the model is being fit has increased to $|R(F_p)|FP$ (where $|R(F_p)|$ is the number of pixels in the feature window) instead of $2FP$, but we can apply essentially the same technique as for the feature-based non-linear minimization. We use the same variables and the same form for the Hessian matrix as for any of the previous methods, depending on whether a projective or Euclidean solution is sought. The difference is that the expected y_i values are intensity values taken from the feature template, defined by the position of the feature in the first frame, and the predicted values Y_i are the intensity values of the image at the position predicted by the shape and motion variables. For each variable υ, the derivative of this projective position is

$$\frac{\partial Y_i}{\partial \upsilon} = \frac{\partial}{\partial \upsilon} I_f(P(f, p) + \mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} I_f(P(f, p) + \mathbf{x}) \cdot \frac{\partial}{\partial \upsilon} P(f, p) \tag{140}$$

The expression $\frac{\partial}{\partial \mathbf{x}} I_f(P(f, p) + \mathbf{x})$ is a 2-vector giving the x- and y-gradients of the image at the predicted position. The $\frac{\partial}{\partial \upsilon} P(f, p)$ terms are the derivative of the predicted position with respect to each of the variables, which are given by equations (106) and (107) for the projective case, (106) (125) (126) and (127) for the Euclidean case, or (134) and (135) if a radial model is used.

The α_{ij} and β_i entries still follow the pattern of equations (103) and (104) but are extended to sum the pixel errors over the region $R(F_p)$ as well. This involves summing $|R(F_p)|$ times as many terms as in the point-based method; for 15×15 windows, this is more than two orders of magnitude more computation. Indeed in the point-based methods, the majority of the computation was spent solving the linear system in α at each iteration, but using the image-based method computing α becomes the primary computational burden. We can reduce the computational requirements considerably by precomputing five terms summations over each window. These values are independent of the particular variable υ and can be used to compute the entries of α and β for all variables for the given point or frame. The summations to compute α_{ij} and β_i are reduced to the same number of terms as for the point based method, and that additional computational load of over the point-based method is only the computation of these five terms for each feature point and each frame.

The β_i entries are computed using the following summation.

$$\sum_{\mathbf{x}\in R(F_p)} \omega(\mathbf{x})\,(y_i - Y_i)\left(\frac{\partial Y_i}{\partial \upsilon_k}\right) =$$

$$= \sum_{\mathbf{x}\in R(F_p)} \omega(\mathbf{x})\,(F_p(\mathbf{x}) - I_f(P(f,p)+\mathbf{x}))\left(\frac{\partial}{\partial \mathbf{x}} I_f(P(f,p)+\mathbf{x})\cdot\frac{\partial}{\partial \upsilon_k}P(f,p)\right)$$

$$= \begin{bmatrix} \frac{\partial}{\partial \upsilon_k}P_x(f,p) \\ \frac{\partial}{\partial \upsilon_k}P_y(f,p) \end{bmatrix}^T \sum_{\mathbf{x}\in R(F_p)} \omega(\mathbf{x})\,(F_p(\mathbf{x}) - I_f(P(f,p)+\mathbf{x})) \left.\begin{bmatrix} \frac{\partial I_f}{\partial x_1} \\ \frac{\partial I_f}{\partial x_2} \end{bmatrix}\right|_{P(f,p)+\mathbf{x}} \quad (141)$$

Note that the 2-vector computed by summation over $R(F_p)$ is the same as the vector $\mathbf{e}$ computed from the image data for tracking in equation (79), where the current translation estimate $\mathbf{h}_n$ has been replaced by $P(f,p)$, the projection of the current shape and motion estimates.

The α_{kj} entries are computed by summing

$$\sum_{\mathbf{x}\in R(F_p)} \omega(\mathbf{x})\left(\frac{\partial Y_i}{\partial \upsilon_k}\right)\left(\frac{\partial Y_i}{\partial \upsilon_j}\right) =$$

$$= \sum_{\mathbf{x}\in R(F_p)} \omega(\mathbf{x})\left(\frac{\partial}{\partial \mathbf{x}} I_f(P(f,p)+\mathbf{x})\cdot\frac{\partial}{\partial \upsilon_k}P(f,p)\right)\left(\frac{\partial}{\partial \mathbf{x}} I_f(P(f,p)+\mathbf{x})\cdot\frac{\partial}{\partial \upsilon_j}P(f,p)\right)$$

$$= \begin{bmatrix} \left(\frac{\partial P_x}{\partial \upsilon_k}\right)\left(\frac{\partial P_x}{\partial \upsilon_j}\right) \\ 2\left(\frac{\partial P_x}{\partial \upsilon_k}\right)\left(\frac{\partial P_y}{\partial \upsilon_j}\right) \\ \left(\frac{\partial P_y}{\partial \upsilon_k}\right)\left(\frac{\partial P_y}{\partial \upsilon_j}\right) \end{bmatrix}^T \sum_{\mathbf{x}\in R(F_p)} \omega(\mathbf{x}) \left.\begin{bmatrix} \left(\frac{\partial I_f}{\partial x_1}\right)^2 \\ \left(\frac{\partial I_f}{\partial x_1}\right)\left(\frac{\partial I_f}{\partial x_2}\right) \\ \left(\frac{\partial I_f}{\partial x_2}\right)^2 \end{bmatrix}\right|_{P(f,p)+\mathbf{x}} \quad (142)$$

Here the 3-vector computed by summation over $R(F_p)$ contains the three unique values of the 2×2 matrix G defined in equation (80), again evaluated at $P(f, p) + \mathbf{x}$ rather than $\mathbf{h}_n + \mathbf{x}$. These variables are the equivalent of $\mathbf{e}$ and G in the tracking section.

The image intensities I_f and their derivatives $\partial I_f / \partial \mathbf{x}$ must be evaluated at the position $P(f, p) + \mathbf{x}$. Since $P(f, p)$ will not in general have an integer value, we compute these intensities by bilinearly interpolating between the four closest pixels as described in section 3.3.1.

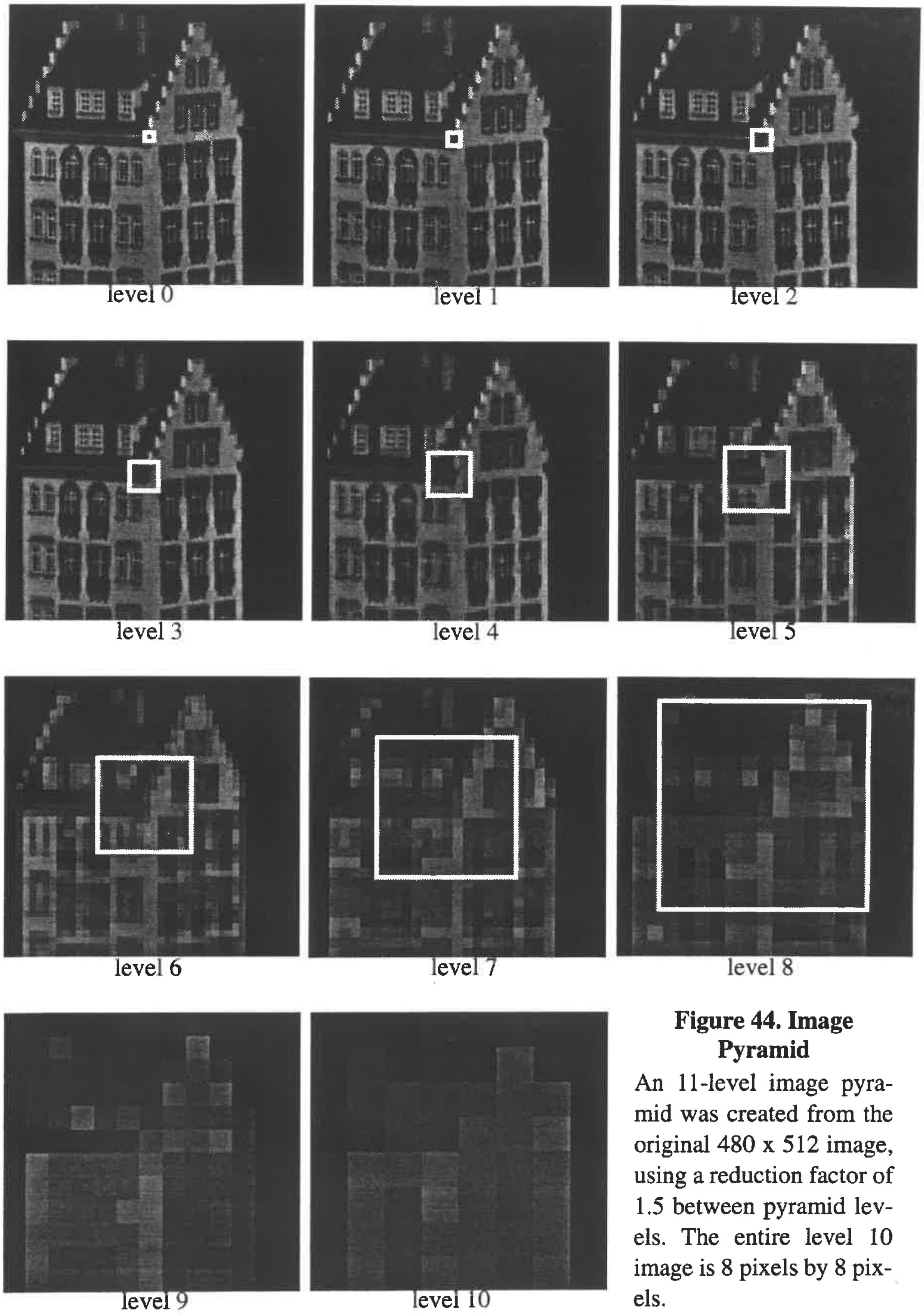
4.5.3. Multi-resolution Technique to Avoid Local Minima

Levenberg-Marquardt iteration is an effective tool for solving non-linear equations when an initial values can be supplied which is sufficiently close to the true, global minimum. However, when doing a gradient-based search in image space, initial estimates which cause the predicted position to differ from the actual position by as little as a few pixels can cause the algorithm to converge to a non-globally optimal local minimum. This is especially true in highly textured regions, where the image derivative at one pixel may bear no relation at all to the derivative at an adjacent pixel. A priori smoothing of the images would make these derivatives more uniform, but this would also smooth out the very information that we depend on for shape recovery. Even smoothing would not help if the initial value predicts features more than one window size away from the correct value.

These problems can be addressed by both widening the feature windows and smoothing, using a hierarchical approach. The problem is solved repeatedly using images of gradually increasing resolution. The image at each level of the pyramid is produced by averaging intensities over some region of the first image. Interpolating to compute intensities at non-integer pixels positions allows an arbitrary reduction between levels of the pyramid, rather than the standard "factor of 2" reduction. The sample pyramid shown in Figure 44 was computed using an inter-level reduction factor of 1.5. Each pixel in the level 1 image represents the average of 2.25 pixels in the original image. Images in the 10th level of the pyramid have been reduced from the first image by a total factor of $1.5^{10} \approx 58$, so each pixel represents the average intensity value of $1.5^{10} \times 1.5^{10} \approx 3325$ pixels.

The feature window size is kept fixed in terms of the number of pixels, so that in the lower resolution images, a larger portion of the image is matched. Our sum of squares error function defined in equation (139) still implicitly assumes a purely translational model of image texture, so when we solve for the shape and motion at the lowest resolution levels, only rough values for the shape over entire region are computed; the entire region around a feature is being modeled as a single planar surface. These estimates will be refined and localized as more detailed and narrowly focused windows are used in the higher resolution images. A sample feature, identified based on its appearance in the first image, is shown with white squares overlaid on the image pyramid of Figure 44.

Occasionally, features which contain significant texture in the highest resolution image do not contain significant texture in the reduced resolution images, since the detail has been



level 0

level 1

level 2

level 3

level 4

level 5

level 6

level 7

level 8

level 9

level 10

Figure 44. Image Pyramid

An 11-level image pyramid was created from the original 480 x 512 image, using a reduction factor of 1.5 between pyramid levels. The entire level 10 image is 8 pixels by 8 pixels.

smoothed away. This will cause the solution for the shape of that feature to wander from the actual feature. By the time the method uses the high resolution images where the feature is clearly trackable, the solution has wandered too far away from the actual minimum, making it impossible to find the correct solution. To address this problem, we widen the feature windows in any level of the pyramid in which the feature contains insufficient texture. A feature is deemed to have insufficient texture when its λ_{min} is below a set threshold.

4.5.4. Extension to Affine Warping Model

The above formulation maintains the implicit translational motion assumption of most previous feature tracking methods. Even if the underlying features are indeed planar, as the object rotates the orientation of the small patches relative to the viewer changes, causing visible warping and scaling effects. For small motions, the translational assumption is reasonable, but as rotations or optical-axis translations become larger, these effects become noticeable in the imagery.

While the image itself may contain perspective deformation, we model each individual feature using an affine warping model. We could allow perspective deformation of the patches as well, but using an affine model allows the system to be solved with reasonable computational power.

Similar to the method used for the translational model, we reduce error by limiting the affine motion of each patch to that which can be explained by the motion of a single rigid object. We represent an affine patch in 3D by the depth to each of its vertices.

$$\varepsilon_{affine} = \sum_{f=1}^{F} \sum_{p=1}^{P} \sum_{\mathbf{x} \in R(F_p)} \left\{ I_f\left(\left[(P_{ur}(f,p) - P_{ul}(f,p)) \;\; (P_{ll}(f,p) - P_{ul}(f,p)) \right] \hat{\mathbf{x}} + P_{ul}(f,p) \right) - F_p(\mathbf{x}) \right\}^2 \tag{143}$$

Here $P_{ur}(f, p)$ is the projection of the upper-right corner of the affine patch, $P_{ul}(f, p)$ is the projection of the upper-left corner of the patch, and $P_{ll}(f, p)$ is the projection of the lower-left corner of the patch. The 2-vector $\hat{\mathbf{x}}$ is simply the $\mathbf{x}$ vector normalized to range from $\begin{bmatrix} 0 & 0 \end{bmatrix}^T$ to $\begin{bmatrix} 1 & 1 \end{bmatrix}^T$.

4.5.5. Direct Shape and Motion Recovery Example

The same image sequence used in section 2.7 is tested here using the direct shape and motion recovery method. The features are identified in the first image, but are not tracked from frame to frame. Only the first 20 images of the sequence are used here. This keeps the total object rotation small, which allows the translational patch assumption to effectively model the image motion.

The shape was recovered using a Euclidean formulation without varying the camera parameters. Since no initial values were provided, a total of ten levels of the image pyramid were used. A reduction factor of 1.5 was used as we found that using the standard factor of 2 reduction did not provide sufficiently smooth transition between levels of the pyramid. Using a reduction factor of 2, sometimes the best shape and motion found at one level still predicted positions with pixel errors of more than one pixel when transformed into the second level, causing the method to become trapped in a local minimum. The initial value used for the shape estimate, as well as the shape estimates at the end of each level of the pyramid, are shown in Figure 45. In the lowest resolution level of the pyramid, each feature covers a large portion of the image. Therefore the recovered shape is a shape estimate for a large portion of the image. For example, features near the corner of the house include image texture from both sides of the house as they slope away from the camera. The depth estimates in those areas tended to be further back, causing a "rounding" of the corner of the house. As higher resolutions and smaller windows are used, these estimates are localized to the specific feature, causing the corner to form a sharp right angle.

Unlike the feature-based example shown in Figure 12, in which several features near the windows were tracked incorrectly, using this method there were no incorrectly tracked features. Every feature remained "attached" to the correct portion of the image throughout the sequence.

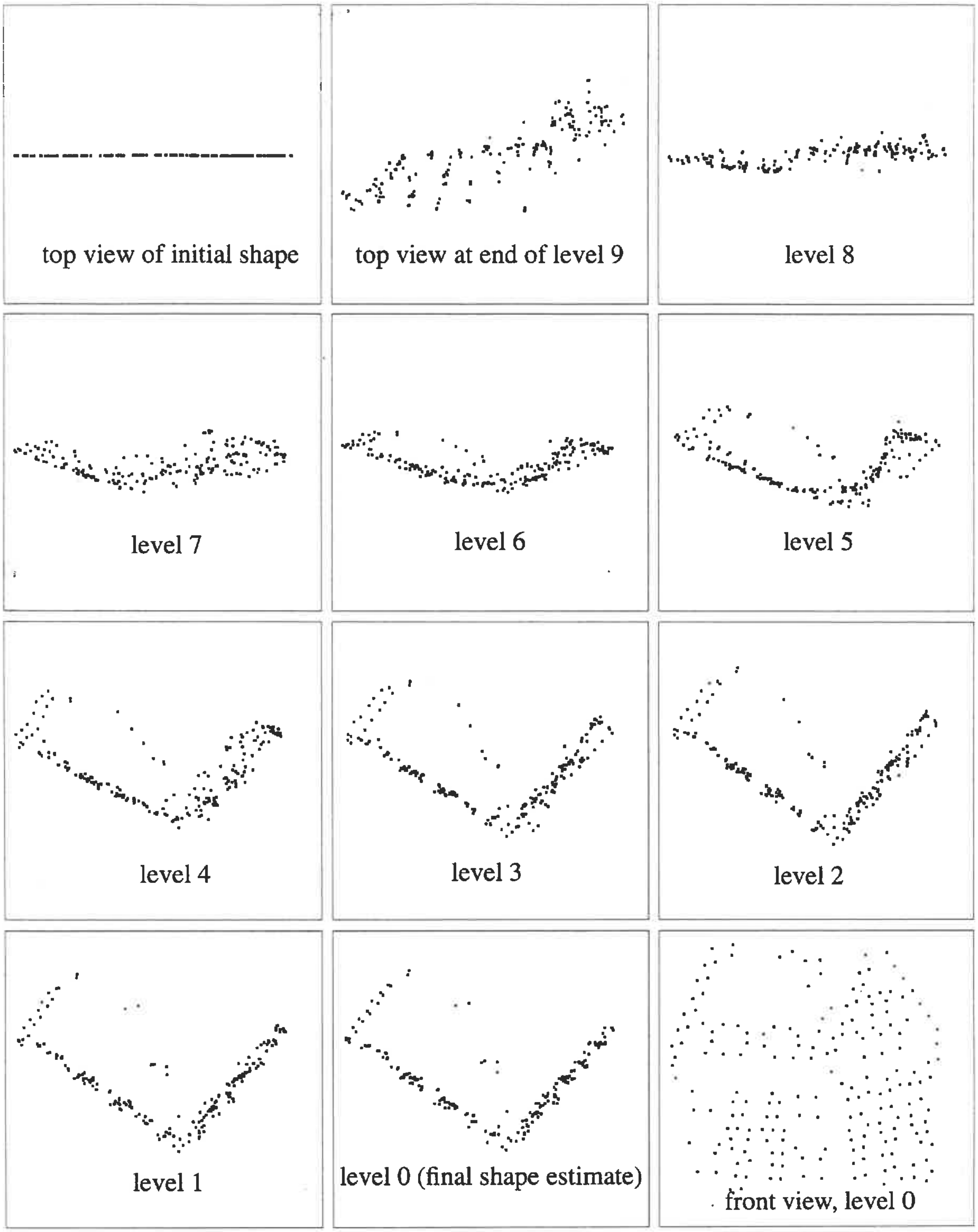


Figure 45. Shape estimates by image pyramid level

The initial object shape was consisted of each point at a location determined by the point's position in the first frame located on a flat plane far from the camera. The front view of the shape changed very little over the course of the iteration. Every point was "tracked" correctly.

5. Conclusions

This research has addressed the problem of recovering shape and motion from noisy image sequences. We have developed methods for robust feature tracking, accommodating missing observations, and shape recovery under a variety of imaging scenarios. We have demonstrated the effectiveness of these techniques on a variety of synthetic and real image sequences.

This research has carefully addressed the problem of shape and motion recovery in the presence of occlusion and missing observations, and has detailed the method's performance as a function of the fill fraction, the fraction of all potential observations for which data is actually available. We furthermore show how to assign confidences to a point based on the tracking results, and how these confidences can be incorporated into the problem formulation.

We have presented a range of methods for recovering shape and motion from image sequences and shown their performance in relation to each other. When the object is far from the camera, bilinear camera models such as scaled orthography and paraperspective can be used to describe the image projection process. The scaled orthographic factorization method requires little knowledge of the camera calibration parameters; only the aspect ratio need be known. When more camera information is available, better results can be obtained by accurately modeling the position effect using the paraperspective factorization method.

These bilinear models are not sufficient for all cases. When the object being imaged is closer to the camera, unmodeled foreshortening effects cause errors in the results of the factorization. The results of the factorization method can be improved considerably using a separate iterative refinement step. While substantial shape improvement can be obtained in just a few iterations of this method, it can take hundreds of iterations to adjust both the shape and motion to the correct minimum-error result. Furthermore, when the object is very close to the camera, unmodeled foreshortening effects can cause the normalization step of the paraperspective factorization method to fail, supplying no initialization for projective refinement.

In such cases in which foreshortening effects are dominant, the projective factorization method works well. This research has demonstrated that projective factorization can produce accurate results when the object is relatively close to the camera, or when noise levels are low, although the factorization is more computationally intensive than the for the bilinear camera models. In addition, we have shown through extensive experimentation that the projective formulation has better convergence properties than direct optimization using a Euclidean formulation.

When a simple initial value is used, the Euclidean optimization method works well only in the same sorts of situations in which paraperspective also works well. By modeling perspective projection completely it avoids the need for a separate refinement step to remove perspective distortion, and is able to converge in a substantially smaller number of iterations than the separate refinement method. The Euclidean optimization method can also be used to refine the results of either the projective or paraperspective factorization method. It can also accommodate radial distortion, as can the projective factorization method. The primary differences between paraperspective factorization, projective factorization, and Euclidean optimization are summarized in Table 4.

Table 4: Comparison of shape recovery techniques

	paraperspective factorization method	projective factorization method	Euclidean optimization method
models foreshortening?	X	☑	☑
insensitive to initial value?	☑	☑	X
accurate for distant objects?	☑	X	☑
approximate running time[a] for 60 points / 60 frames on Sparc 5/85	4 sec	120 sec	400 sec[b]
shape-depth running time	N/A	60 sec	110 sec[b]

a. Does not include zero-noise cases, which often required more iterations than usual since the convergence criteria was not satisfied until machine floating point precision limits were reached.
b. Does not include low-depth sequences in which the method converged to local minima, often after an unusually large number of iterations.

Finally, this thesis has investigated extending the point-based non-linear optimization framework to recover shape and motion directly from image sequences without a separate tracking stage. This enforces object rigidity throughout the entire process, preventing features from following non-rigid trajectories. This method is by far the most computationally expensive, requiring hours to solve a typical image sequence, and still requires modification to address issues such as occlusion and unmodeled image patch changes in order to become a practically useful method.

5.1. Future work

5.1.1. Feature Tracking

When the camera is extremely unsteady or the camera operator pans back and forth across the field of view, the same feature points may enter and leave the field of view several times within a short period of time. The current feature tracker considers a point “lost” when it leaves the field of view; after many features have been lost, the system searches for new features. Sometimes a point leaves the field of view for a single frame, and then reappears. The current tracker cannot recognize when a previous feature has re-entered the field of view, and therefore any point which leaves for even a frame is considered permanently lost. It may detect the feature as a new feature, and continue to track it for future frames, without recognizing it as the same as a previously seen feature. By not recognizing that the two are the same feature, significant information is lost.

The tracking scheme should be extended to try to detect when previous features re-enter the field of view. This is hindered by the fact that no estimate for the points motion is available once it has left the field of view. Searching the image in the area where the feature left would be expensive and would be susceptible to finding false matches, yet in the tracking stage the three-dimensional information that would fix the point's motion even when it is unobserved is unavailable. One possibility is to use projective invariants. A point's position in the current frame could be estimated by requiring that its cross-ratios with respect to some subset of other tracked points be the same as in those frames in which the point was observed. If there are no features in the current frame which were observed simultaneously with the given point, or if the position estimate lies outside the field of view, the point is still considered to be unobserved. However, if the position estimate lies within the field of view, that estimate is used as an initial value for feature tracking, the feature point is tracked, and if the total sum of squares error for that point passes a simple threshold test, the point is considered to be rediscovered.

Additionally, the feature detection method described in this thesis could be extended to allow varying feature sizes. During the feature detection stage, each potential feature window could be expanded until the window contains sufficient texture to enable tracking. This would enable large, relatively textureless regions to be tracked as features, on the assumption that such regions are planar. Similarly, if one eigenvalue of the G matrix is small, the window could be extended in the direction of the corresponding eigenvector until the window contains enough information to precisely track the feature in both directions. This would enable edge-like features to be used by extending them to include the edges' endpoints.

5.1.2. Accommodating Long Sequences

Both the perspective and projective factorization methods have been shown to work in the presence of occlusion and missing data. However, both have the same limitations; as the fill percentage drops, the errors increase, finally reaching a point where local minima in the search space prevent even qualitatively correct solutions. Long sequences in which the camera views several distinct parts of an object, room, or scene may often have fill fractions of ten percent or even less. The methods we have addressed for occlusion handling clearly do not gernally converge at such low fill fractions unless given very accurate initial values.

An obvious solution is to separate the sequence into several smaller sequences which will have larger fill fractions. If the sequences are slightly overlapped so that they share some points or share some images, the final solutions can be "pasted" together to for a single shape and object by scaling, rotating, and translating the solutions so that the common points or common motion parameters coincide. Thus a single motion and single object shape could be produced for the sequence. This solution could be further refined using the Euclidean optimization method.

5.1.3. Further Incorporating Image Data

The direct shape and motion recovery method can work well when used to refine shape and motion estimates provided by traditional techniques. However, it is still based on the translational patch assumption, which may not always be a valid assumption. We have shown in this paper that the approach can be extended to an affine warping model, but in practice this requires reasonable initial estimates of the orientation of each patch. One could imagine even more complex formulation, in which each patch is defined by a number of control points whose depths vary independently. However, this would also require initialization and poor initial values may prevent solution.

Adding even more degrees of freedom without adding information makes the system very sensitive to noise. The primary goal of extending the patch model to an affine model was to recover surface normals, which are necessary to build three-dimensional surface models using point interpolation techniques. A better idea might be to introduce a surface model at the time of the shape and motion reconstruction. The surface model and current shape and motion estimate would be used to predict the appearance of the object in each frame. The shape and motion values would be refined to bring the predicted appearance into correspondence with the actual observed appearance.

For example, we might detect a few hundred feature points in the first image and compute a 2D triangulation of those points. The actual image intensities of each triangle would be used instead of just the intensities of the feature points themselves. As the shape estimates for a particular point are changed, the predicted appearances of all of the surface triangles containing that point as a vertex change. This allows shape and motion recovery to make use of all image information rather than just using image intensity information at the feature points. The basic non-linear formulation described in this thesis could be extended to handle this case, though the block-diagonal nature of the shape portion of the Hessian matrix would be destroyed. Still, with an appropriately chosen preconditioner matrix, the conjugate gradient method should be able to invert the Hessian matrix in a reasonable amount of computation. Such a method would resemble the recent method of Szeliski and Coughlan, but would be a true multi-frame formulation producing shape and motion as a result rather than image motions.

5.1.4. Camera calibration

The framework developed in this paper allows recovery of the camera focal length, aspect ratio, image center, and first order radial distortion parameter. However, our experience indicates that allowing these variables can wander far from their correct values if not further constrained. Our solution is generally to fix the camera parameters until shape and motion estimation is almost complete, and then to allow them all to vary. This requires accurate initial estimates of the camera parameters, however. This dependence should be experimentally quantified to determine precisely how accurate camera estimates need to be, and in what circumstances camera calibration parameters can be recovered simultaneously with shape and motion recovery.

Ideally, camera calibration requirements should be eliminated from both the paraperspective and projective factorization methods. It is possible in theory, since the addition of 4 unknowns (focal length, aspect ratio, and image center) still leaves the metric constraints overconstrained. However, straightforward implementation of non-linear solution techniques to solve for these parameters as well as the matrix A failed except when the initially provided camera calibration parameters were very accurate.

5.1.5. Multiple cameras

This formulation could easily extended to allow multiple cameras. If the cameras are fixed relative to each other, each camera's set of parameters would include both its intrinsic parameters, and its orientation relative to the first camera. Computation of the expected image position of any point would the require an additional transformation from the "frame" coordinate system to the coordinate system of the particular camera from which the scene is being viewed. If the cameras are allowed to move independently, then images from additional cameras would be treated in much the same way as additional images are currently treated. However, there would be multiple sets of camera parameters, and each observation would specify the particular frame, point, and camera to which the observation corresponds.

References

[1] Ajit Singh, Optic Flow Computation: A Unified Perspective, IEEE Computer Society Press, Los Alamitos, California, 1991.

[2] John Y. Aloimonos, *Perspective Approximations*, Image and Vision Computing, 8(3):177-192, August 1990.

[3] A. Azerbayejani and A. Pentland, *Recursive Estimation of Motion, Structure, and Focal Length*, MIT Media Laboratory, Perceptual Computing Technical Report #243, October 15, 1993.

[4] P. Beardsley, A. Zisserman, and D. Murray, *Navigation Using Affine Structure from Motion*, European Conference on Computer Vision, 1994, pp. 85-96.

[5] J. R. Bergen and E. H. Adelson, *Hierarchical, computationally efficient motion estimation algorithms*, Journal of the Optical Society of America, 4:35, 1987.

[6] B. Boufama, R. Mohr, and F. Veillon, *Euclidean Constraints for Uncalibrated Reconstruction*, International Conference on Computer Vision, 1993.

[7] B. Boufama, D. Weinshall, and M. Werman, *Shape from motion algorithms: a comparative analysis of scaled orthography and perspective*, European Conference on Computer Vision, 1994, pp. 199-204.

[8] T. Broida, S. Chandrashekhar, and R. Chellappa, *Recursive 3-D Motion Estimation from a Monocular Image Sequence*, IEEE Transactions on Aerospace and Electronic Systems, 26(4):639-656, July 1990.

[9] Joao Costiera, *Progress Report on Occlusion Detection*, Unpublished, May 6, 1994.

[10] K. Deguchi, T. Sasano, H. Arai, and Y. Yoshikawa, *Feasibility Study of the Factorization Method to Reconstruct Shape from Image Sequences*, Technical Report Meip7-93017-02, Department of Mathematical Engineering and Information Physics, University of Tokyo, March 1994.

[11] S. Demey, A. Zisserman, P. Beardsley, *Affine and Projective Structure from Motion*, Proceedings of the British Machine Vision Conference, September 1992, pp. 49-58.

[12] J. E. Dennis Jr, and Robert B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1983.

[13] R. Dutta and M. A. Snyder, *Robustness of Correspondence-Based Structure from Motion*, International Conference on Computer Vision, December 1990, pp. 299-306.

[14] O. D. Faugeras, *What can be seen in three-dimensions with an uncalibrated stereo rig?*, European Conference on Computer Vision (ECCV), May 1992, pp. 563-578.

[15] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank, *Camera self-calibration: theory and experiments*, Proceedings of the European Conference on Computer Vision, pp. 321-334, 1992.

[16] O. D. Faugeras, F. Lustman, G. Toscani, *Motion and Structure from point and line matches*, International Conference of Computer Vision (ICCV), June 1987, pp. 25-34

[17] Bruce D. Lucas and Takeo Kanade, *An Iterative Image Registration Technique with an Application to Stereo Vision*, Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981.

[18] Richard I. Hartley, *Euclidean Reconstruction from Uncalibrated Views*, In Applications of Invariance in Computer Vision, pp. 237-356, 1993.

[19] C. G. Harris, *DROID Analysis of the NASA Helicopter Images*, Proceedings of the IEEE Special Workshop on Passive Ranging, 10 October 1991.,

[20] E. C. Hildreth, The Measurement of Visual Motion, MIT Press, Cambridge Mass., 1983.

[21] B. K. P. Horn and B. Schunck, *Determining Optical Flow*, Artificial Intelligence, 17:185-203, 1981.

[22] B. K. P. Horn and E.J. Weldon Jr., *Direct Methods for Recovering Motion*, International Journal of Computer Vision, volume 2, pp. 51-76, 1988.

[23] H. C. Longuet-Higgins, *A Computer Algorithm for Reconstructing a Scene from two Projections*, Nature 293, 1981, pp. 133-135.

[24] Larry Matthies, Takeo Kanade, and Richard Szeliski, *Kalman Filter-based Algorithms for Estimating Depth from Image Sequences*, International Journal of Computer Vision, vol. 3, pp. 209-239, 1989.

[25] R. Mohr, F. Veillon, J. Quan, *Relative 3D Reconstruction Using Multiple Uncalibrated Images*, Computer Vision and Pattern Recognition, June 1993, pp. 543-548.

[26] Joseph L. Mundy and Andrew Zisserman, *Geometric Invariance in Computer Vision*, The MIT Press, 1992, p. 512.

[27] H. H. Nagel, *On the estimation of dense displacement maps from image sequences*, Proc. ACM Motion Workshop, Toronto, pp. 59-65, 1983.

[28] Yu-ichi Ohta, Kiyoshi Maenobu, and Toshiyuki Sakai, *Obtaining Surface Orientation from Texels Under Perspective Projection*, Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp. 746-751, August 1981.

[29] Conrad J. Poelman and Takeo Kanade, *A Paraperspective Factorization Method for Shape and Motion Recovery*, Technical Report CMU-CS-93-219, Carnegie Mellon University, Pittsburgh, PA, December 1993.

[30] Jean Ponce, David Marimont, and Todd Cass, *Analytical Methods for Uncalibrated Stereo and Motion Reconstruction*, Tech. Report AI-RCV-93-07, Beckman Institute, Univ. of Illinois, 1993.

[31] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, 1988.

[32] Axel Ruhe and Per Ake Wedin, *Algorithms for Separable Nonlinear Least Squares Problems, SIAM Review*, Vol. 22, No. 3, July 1980.

[33] Amnon Shashua, *Projective Structure from two Uncalibrated Images: Structure from Motion and Recognition*, MIT AI Laboratory A.I. Memo No. 1363, September 1992.

[34] Ajit Singh, *Optic Flow Computation: A Unified Perspective*, IEEE Computer Society Press, Los Alamitos, California, 1991.

[35] M. E. Spetsakis and J. Y. Aloimonos, *Structure from motion using line correspondences*, International Journal of Computer Vision, 4(3):171-185, June 1990.

[36] S. Soatto, P. Perona, R. Frezza, G. Picci, *Recursive Motion and Structure Estimations with Complete Error Characterization*, Computer Vision and Pattern Recognition proceedings, June 1993, pp. 428-433.

[37] Long Quan, *Self-calibration of an Affine Camera from Multiple Views*, Technical Report R.T. Imag-Lifia 26, LIFIA-CNRS-INRIA, Grenoble, France, November 1994.

[38] Richard Szeliski and Sing Bing Kang, *Recovering 3D Shape and Motion from Image Streams using Non-Linear Least Squares*, Technical Report 93/3, Digital Equipment Corporation, Cambridge Research Lab, March 1993.

[39] Camillo Taylor, David Kriegman, and P. Anandan, *Structure and Motion From Multiple Images: A Least Squares Approach*, IEEE Workshop on Visual Motion,

pp. 242-248, October 1991.

[40] J. I. Thomas, A. Hansen, J. Oliensis, *Understanding Noise: The Critical Role of Motion error in Scene Reconstruction*, International Conference of Computer Vision, June 1993, pp. 325-329.

[41] Morris M. Thompson, ed., Manual of Photogrammetry, Third Edition, American Society of Photogrammetry, Falls Church, VA., 1966.

[42] Carlo Tomasi and Takeo Kanade, *Shape and Motion from Image Streams: a Factorization Method - 2. Point Features in 3D Motion*, Technical Report CMU-CS-91-105, Carnegie Mellon University, Pittsburgh, PA, January 1991.

[43] Carlo Tomasi and Takeo Kanade, *Shape and Motion from Image Streams: a Factorization Method*, Technical Report CMU-CS-91-172, Carnegie Mellon University, Pittsburgh, PA, September 1991.

[44] Roger Tsai, *A Versatile Camera Calibration Technique for High-accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses*, IEEE Journal or Robotics and Automation, Vol. RA-3, No. 4, August 1987, pp. 323-344.

[45] Roger Tsai and Thomas Huang, *Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6(1):13-27, January 1984.

[46] Reg Willson and Steve Shafer, *A Perspective Projection Camera Model for Zoom Lenses*, Proceedings of Second Conference on Optical 3-D Measurement Techniques, Institute of Geodesy and Photogrammetry, Federal Institute of Technology, Zurich, October 1993.

[47] Yalin Xiong and Steven A. Shafer, Dense Structure From a Dense Optical Flow Structure, Technical Report CMU-RI-TR-95-10, Carnegie Mellon University, Pittsburgh, PA, April 1995.