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

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

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

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 handheld 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.

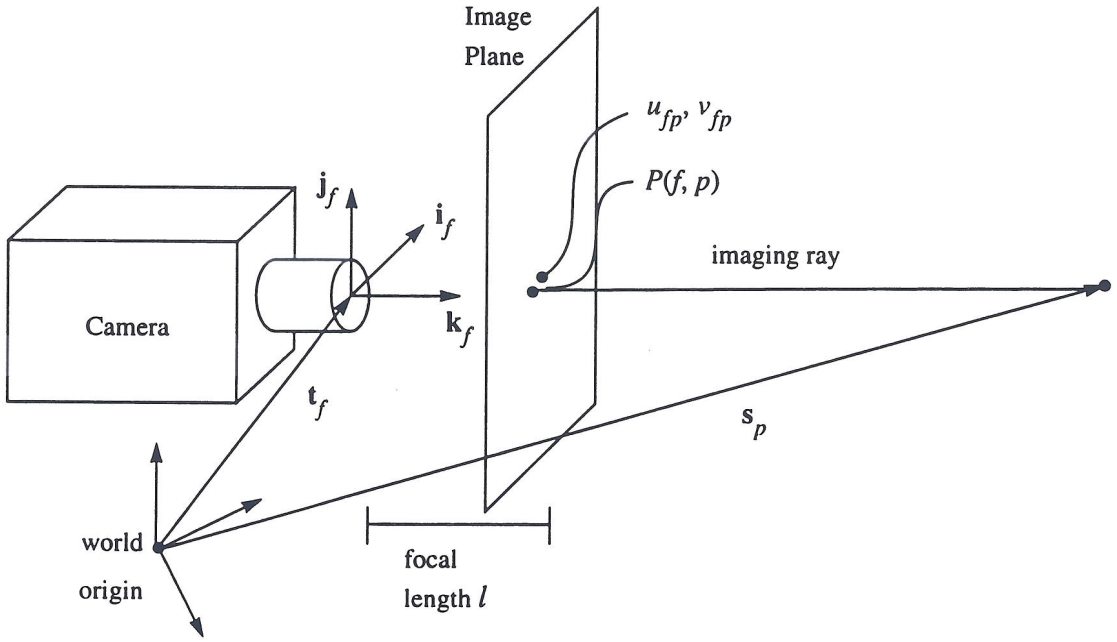


Figure 1. Coordinate system and Notation

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

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

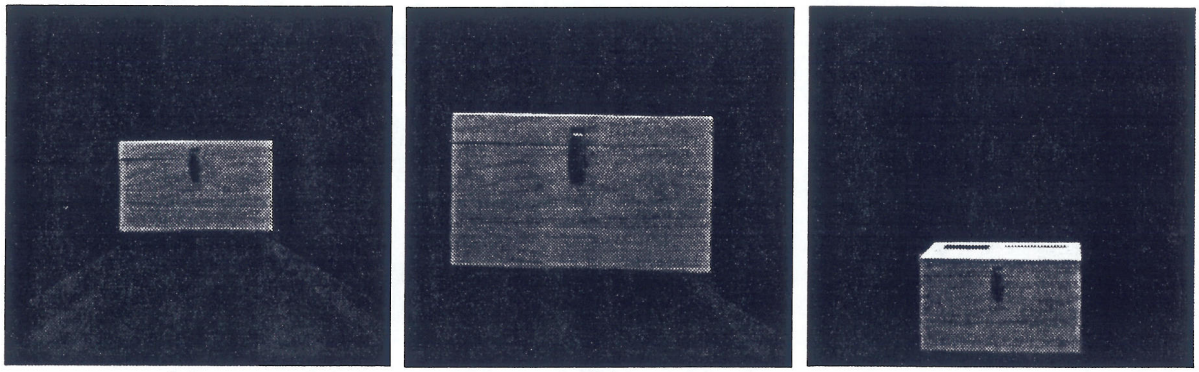
1.5. Thesis Overview

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

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.



(a) (b) (c)

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.

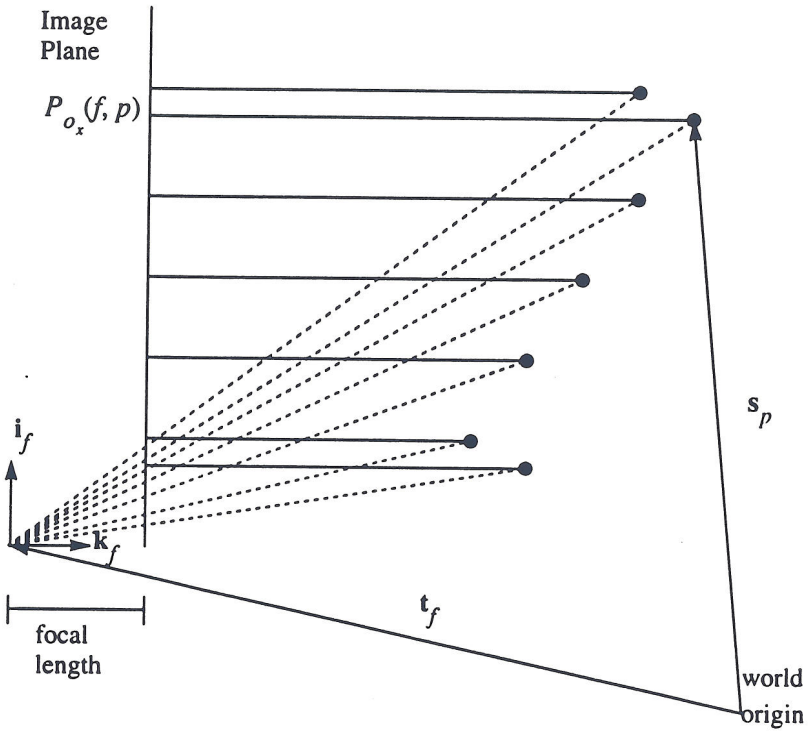


Figure 3. Orthographic projection in two dimensions
Dotted lines indicate perspective projection

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

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

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 $W^* = W - T[1 \ \ldots \ 1]$.

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

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

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.

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

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

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

where

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \tag{15}$$

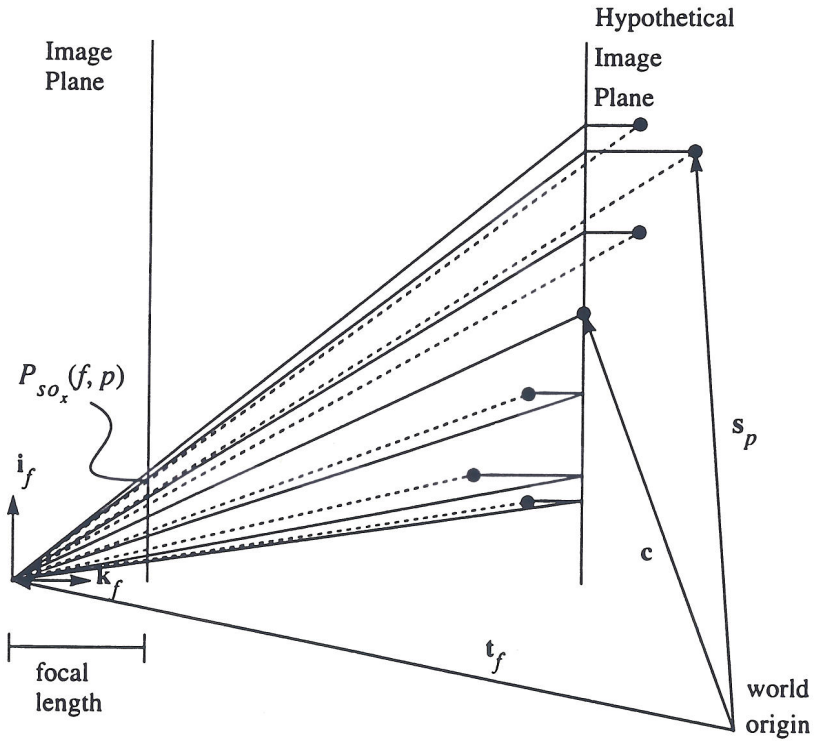


Figure 4 Scaled Orthographic Projection in two dimensions
Dotted lines indicate true perspective projection

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

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

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 per-

spective 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.

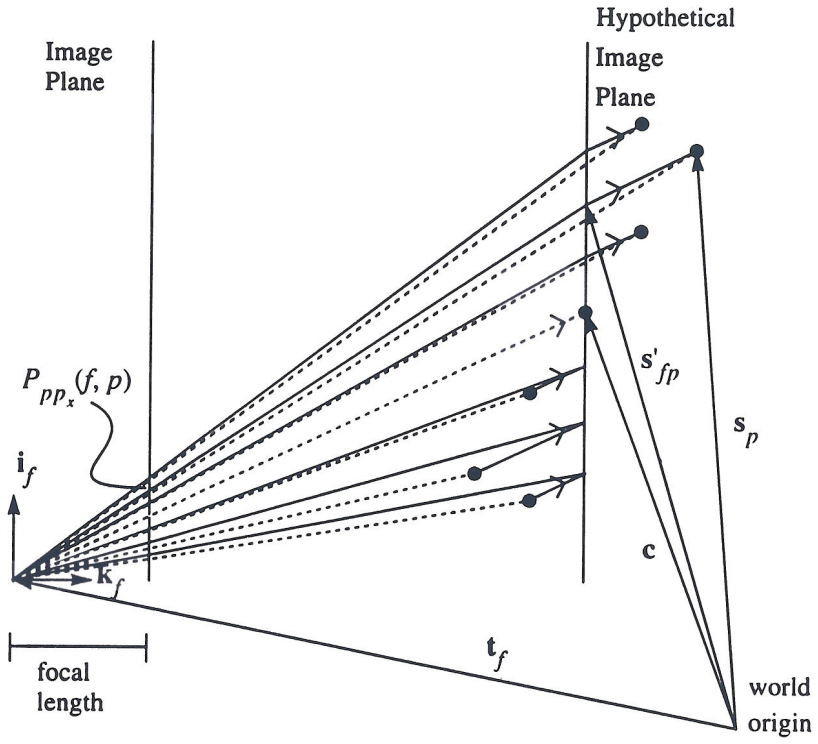


Figure 5. Paraperspective projection in two dimensions
Dotted lines indicate perspective projection
↗↗ indicates parallel lines

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,

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

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

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

These equations can be rewritten as

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

where

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \quad (30)$$

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

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

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

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

where the m_{ij} are free to take on any values. In motion applications, this matrix is com-

monly 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

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

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

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

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

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

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

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

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

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

Therefore we can compute x_f and y_f, which are the elements of the translation vector T,

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

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

Again, we do not know a value for z_f, but using the relations specified in equation (53) and the additional knowledge that $\left| \hat{\mathbf{k}}_f \right| = 1$, 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 $A\begin{bmatrix}\pm1 & 0 & 0\\ 0 & \pm1 & 0\\ 0 & 0 & \pm1\end{bmatrix}$ 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

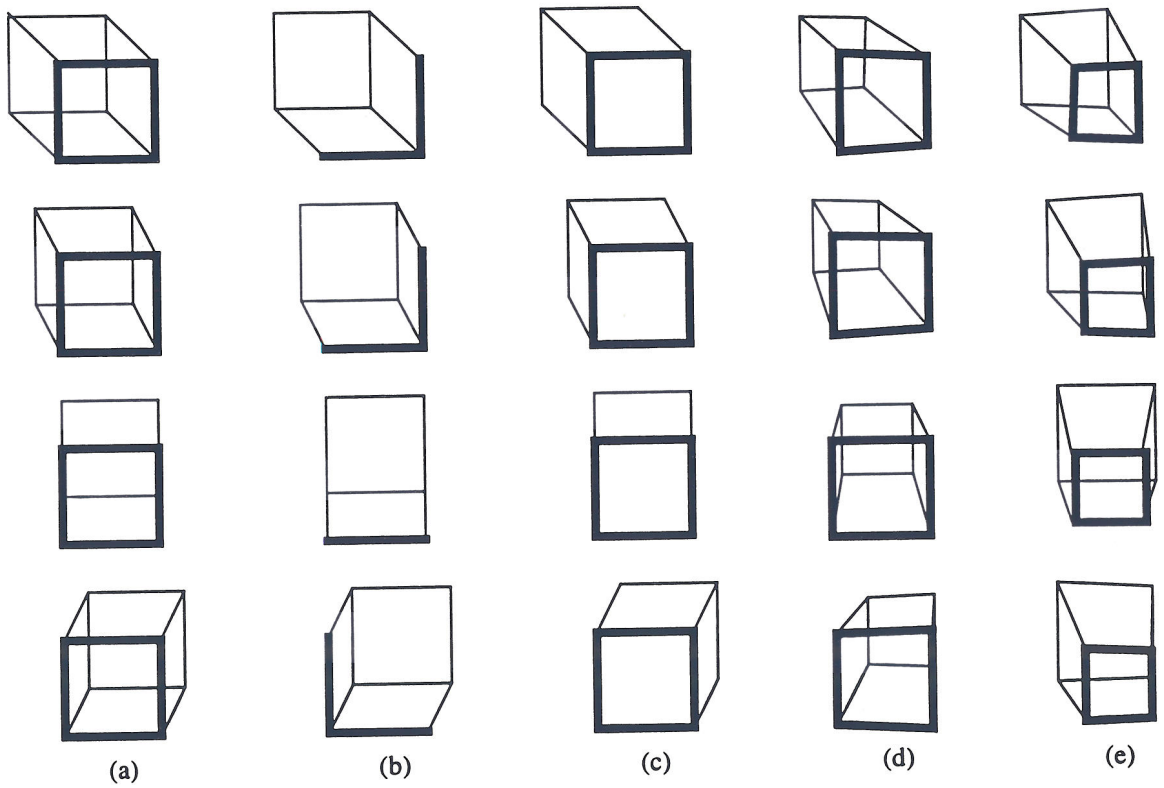


Figure 6 Ambiguity of Solution

(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

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

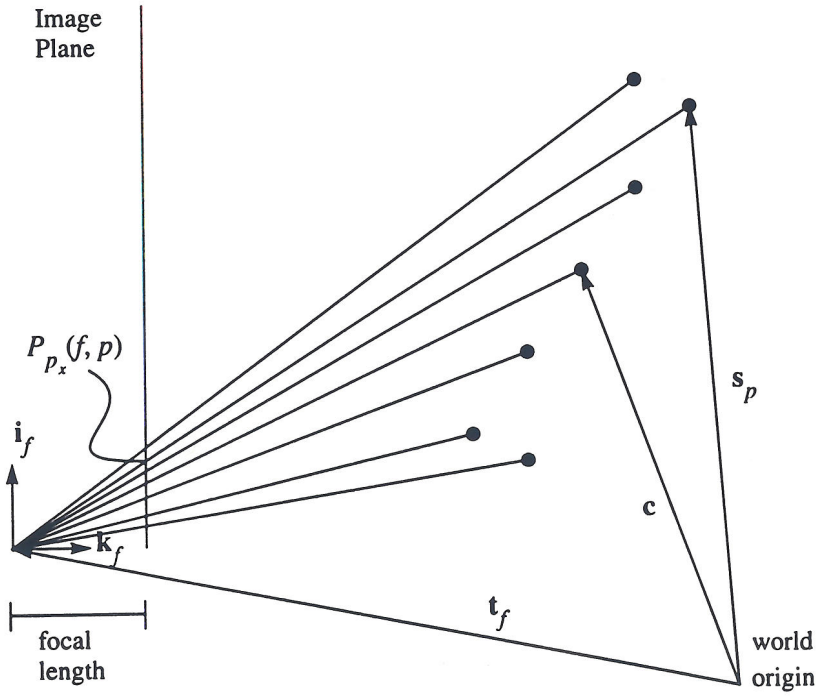


Figure 7. Perspective Projection in two dimensions

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

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

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

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

The vector β is defined by

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

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

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

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 ($s_p = \begin{bmatrix} s_{p1} & s_{p2} & s_{p3} \end{bmatrix}$) 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.

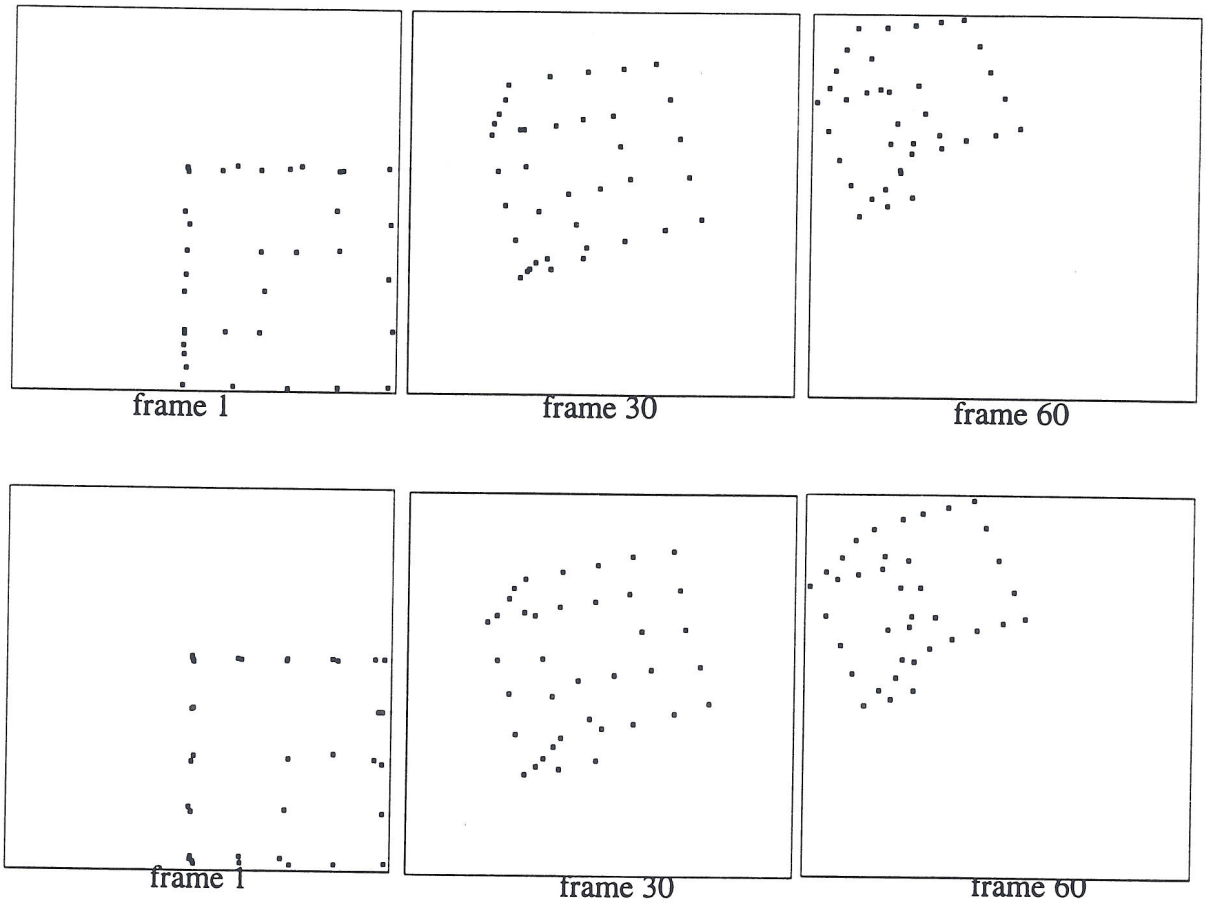


Figure 8. Sample Synthetic Image Sequences

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-

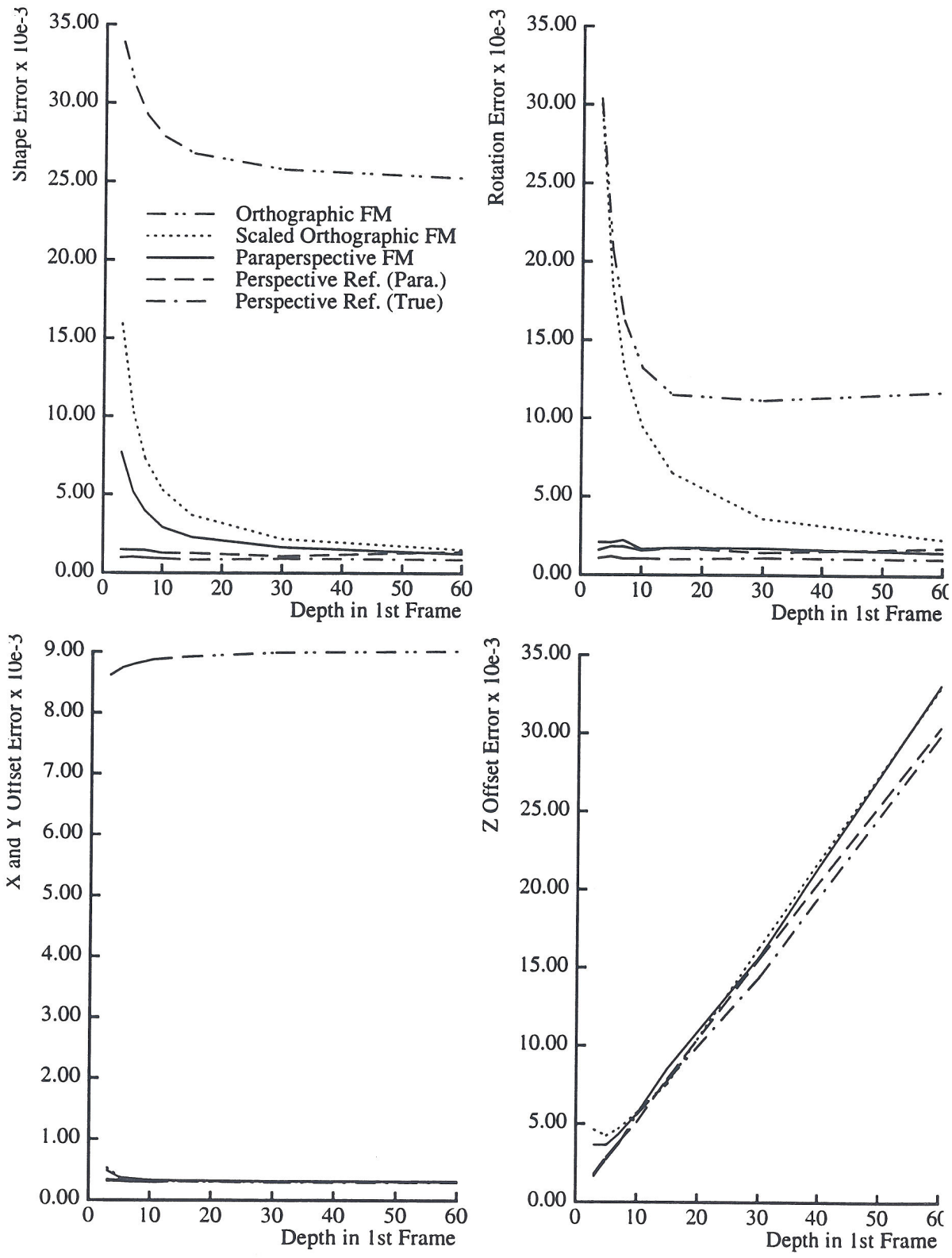


Figure 9. Methods compared for a typical case
noise standard deviation = 2 pixels

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.

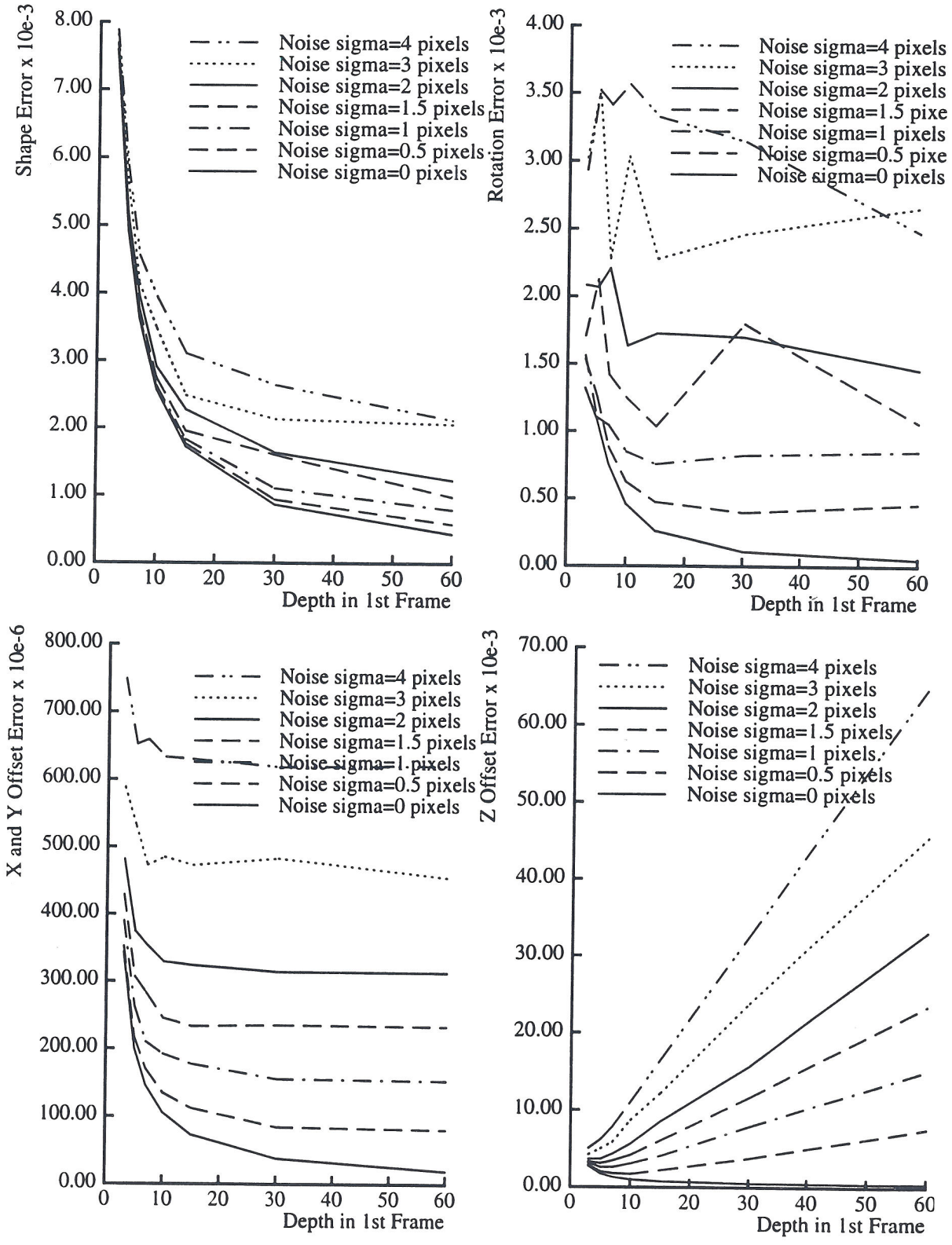
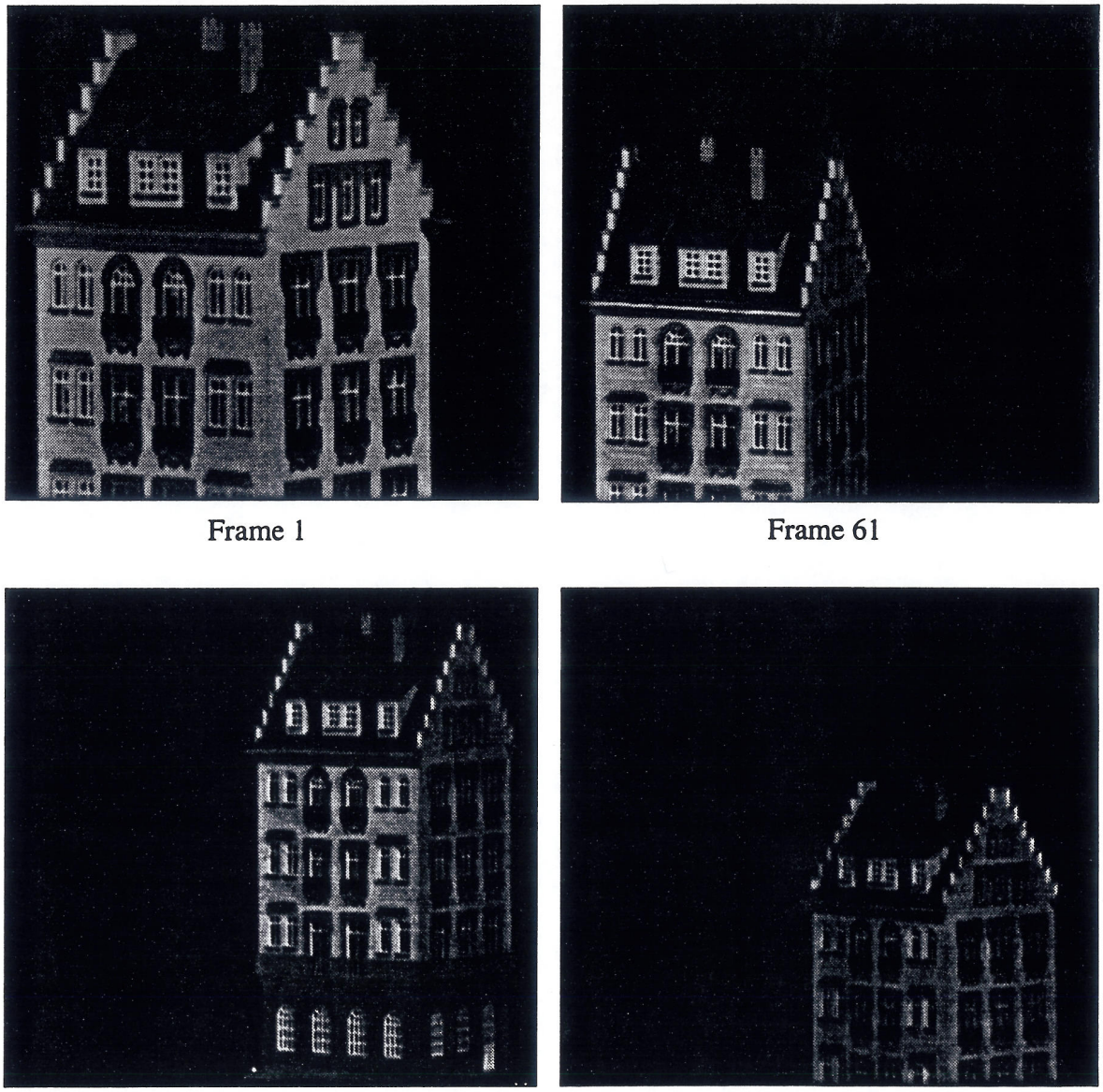


Figure 10. Paraperspective shape and motion recovery by noise level



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

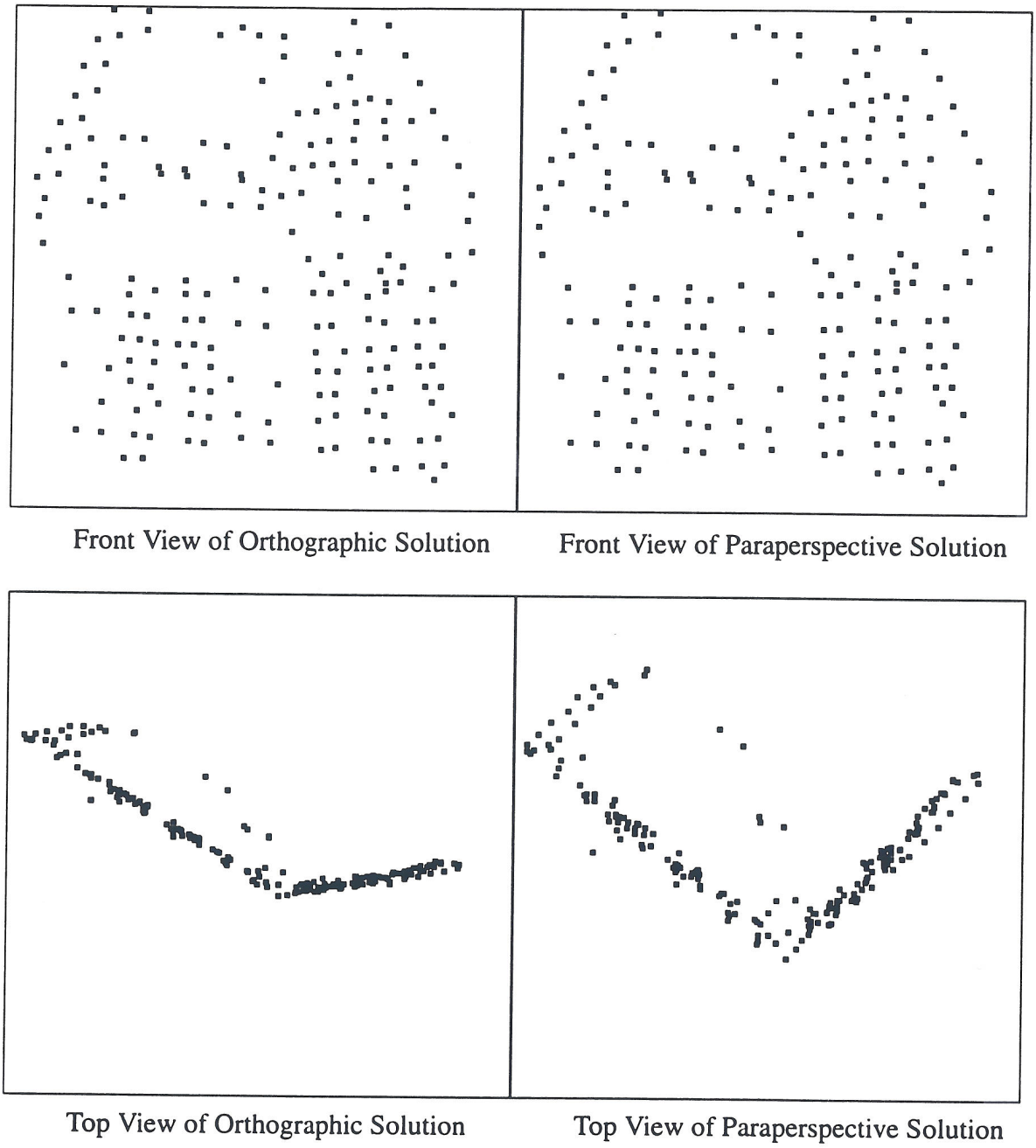


Figure 12. Comparison of Orthographic and Paraperspective Shape Results

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.

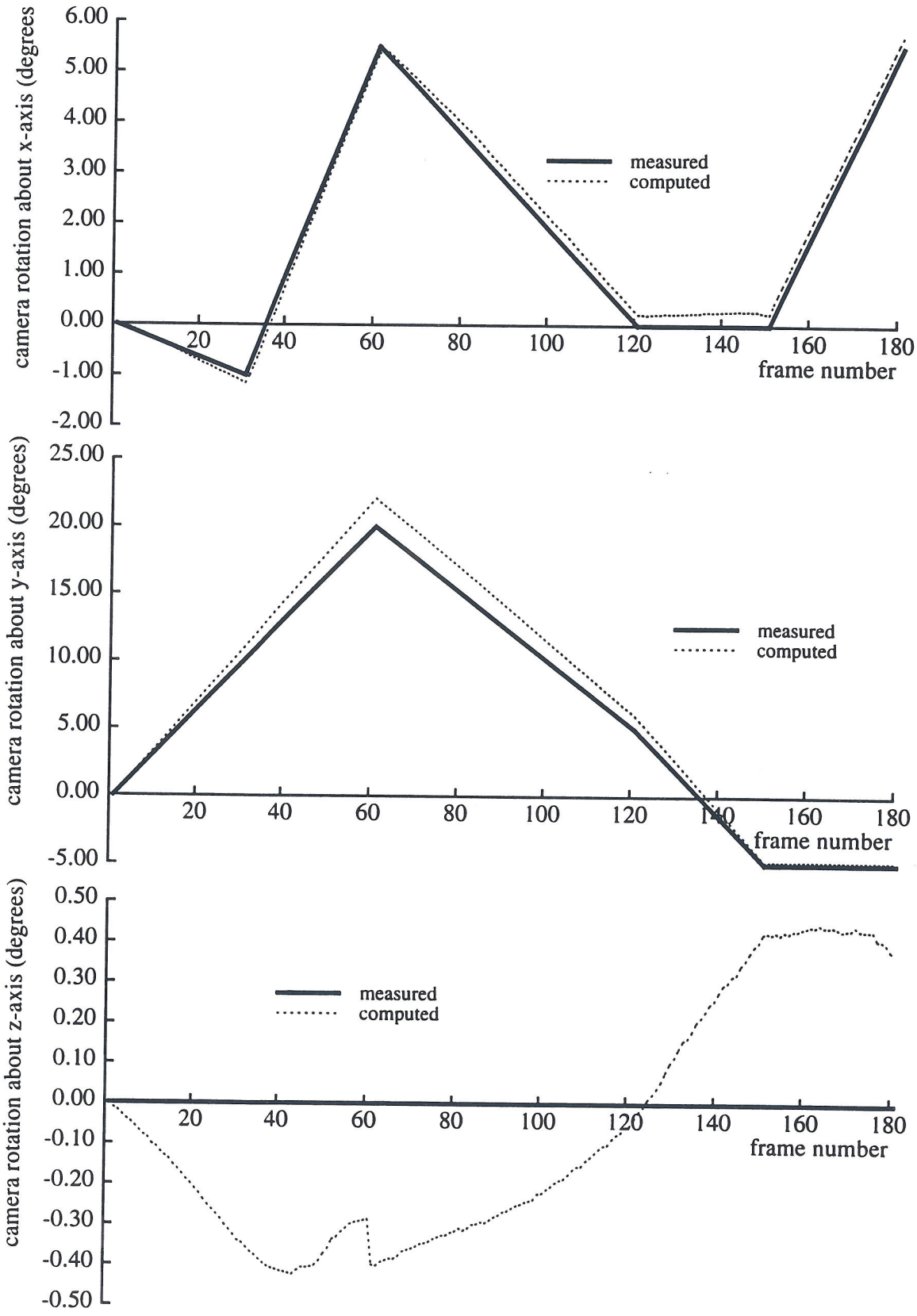


Figure 13. Hotel Model Rotation Results

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.

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

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.

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

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

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.

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

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.

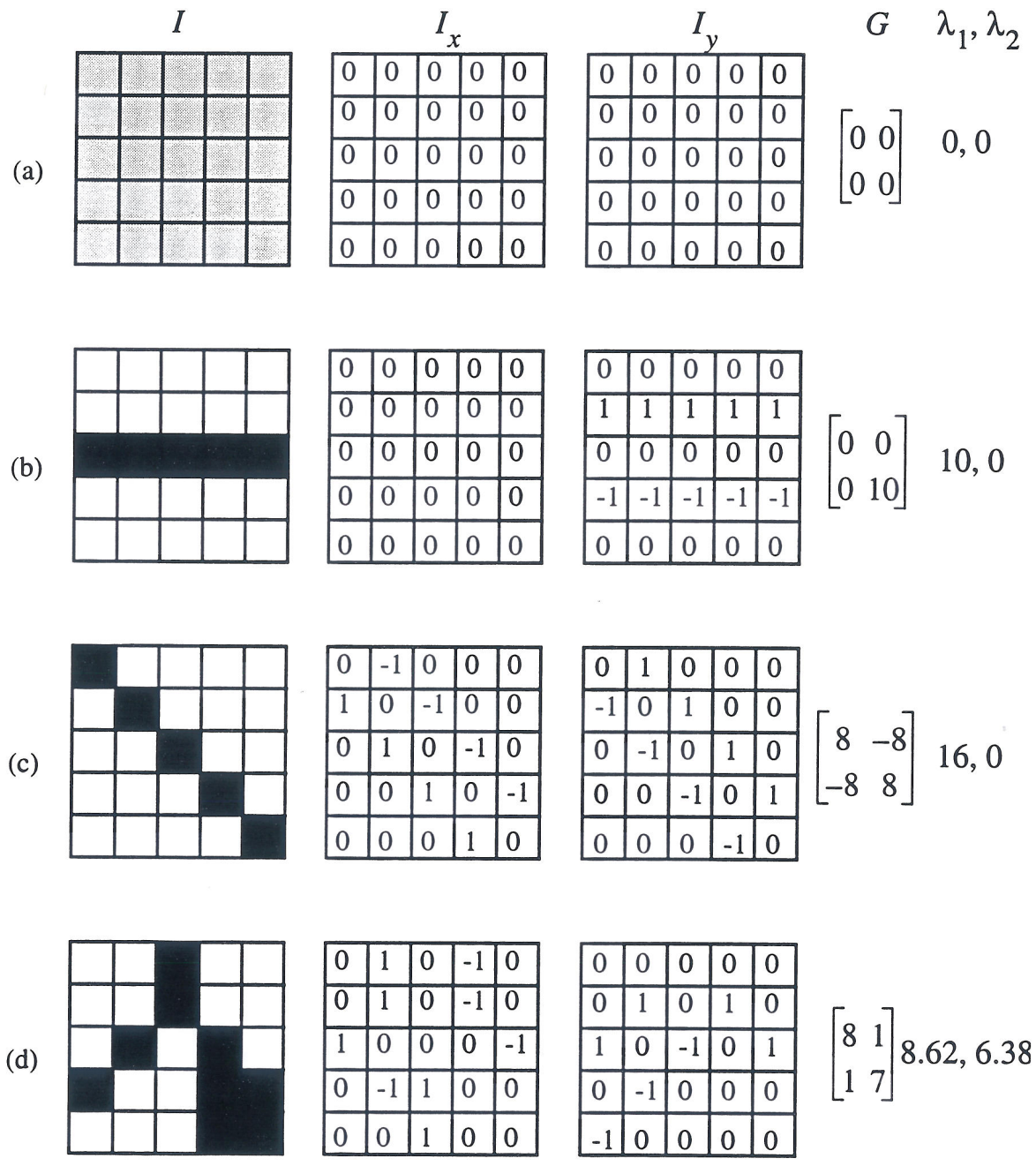


Figure 14. Feature trackability of example image regions

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 track-

ing 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-

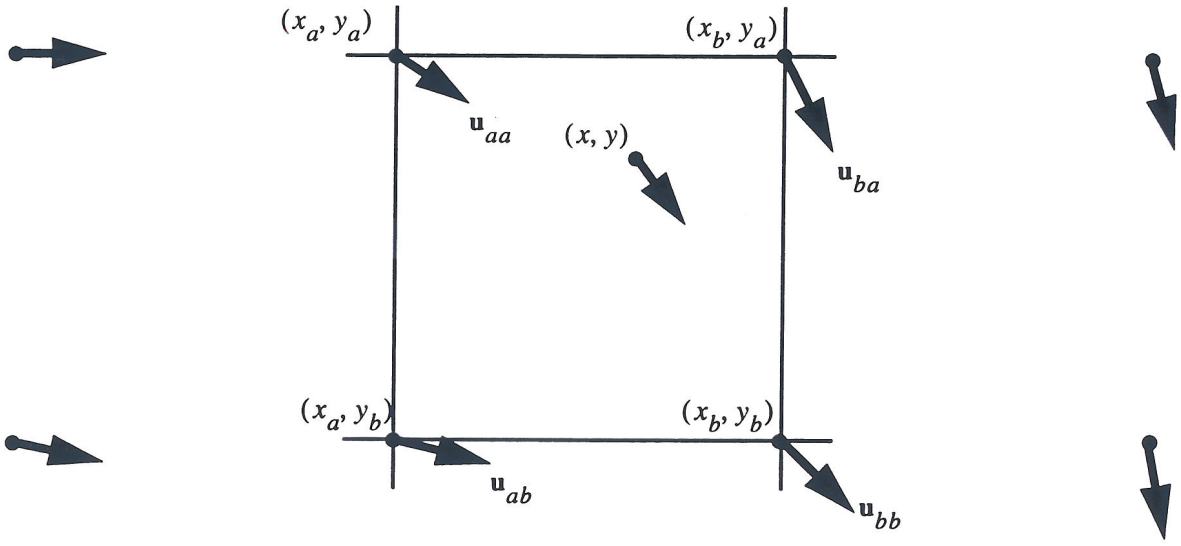


Figure 15. Interpolation of Sparse Optical Flow

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.

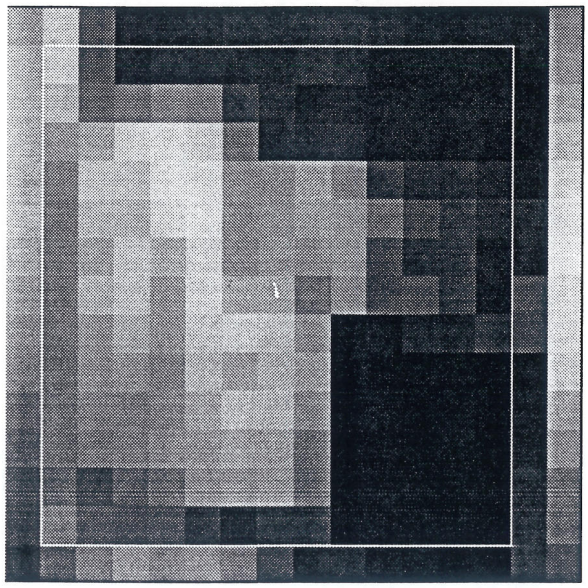
$$\begin{aligned} \gamma_{l_n}(x,y) &= (1-k)\,\gamma + k\gamma_{l-1}(x,y) \\ \mathbf{u}_{l_n}(x,y) &= \gamma_{l_n}^{-1}\left[\,(1-k)\,(\gamma_R \mathbf{h}_n) + k\gamma_{l-1}(x,y)\,h_{l-1}(x,y)\,\right] \end{aligned} \tag{89}$$

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.

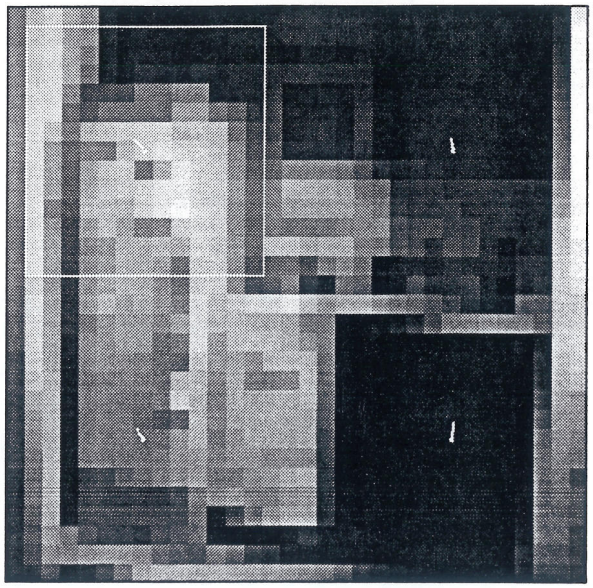
3.3.2. Sparse Optical Flow Results

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.

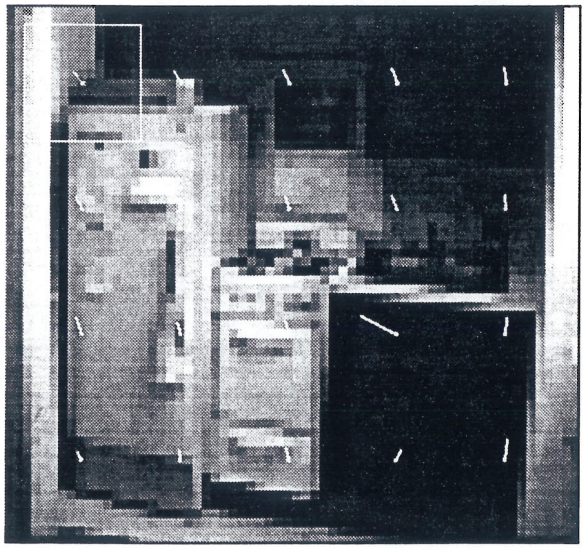
Figure 18 shows the results on the same pair of images using an inter-level weighting constant of $k = 0.5$, though we observed roughly equivalent behavior with k ranging from 0.3 to 0.7. The flow results were substantially improved in portions of the image which lacked significant image texture, enabling 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.



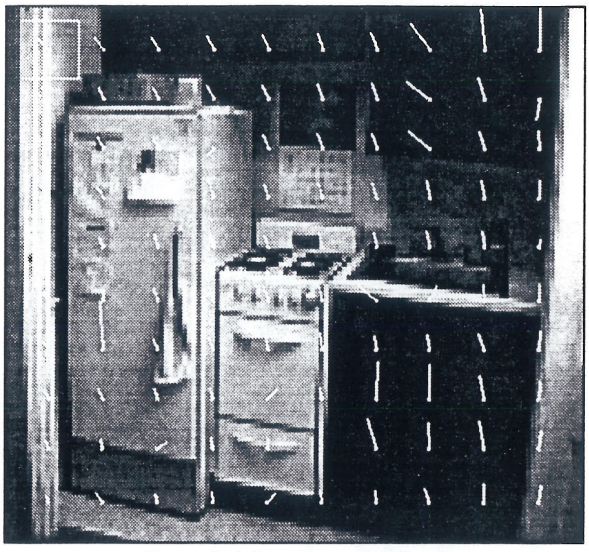
Level 1 image and flow



Level 2 image and flow



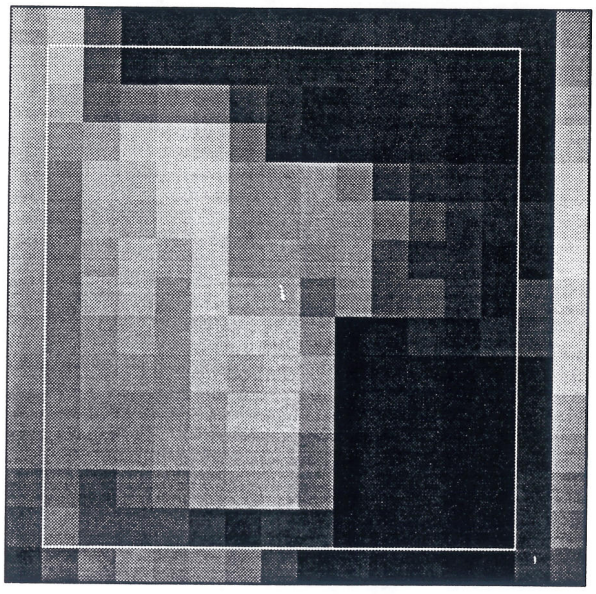
Level 3 image and flow



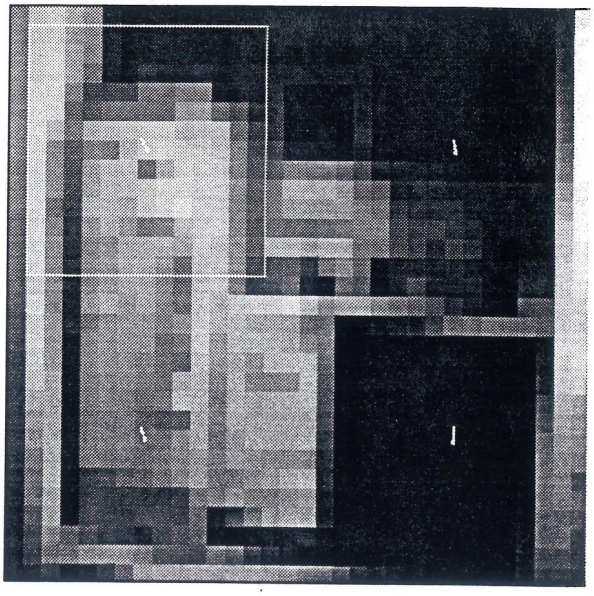
Level 4 image and flow

Figure 17. Unsmoothed optical flow - Kitchen Sequence

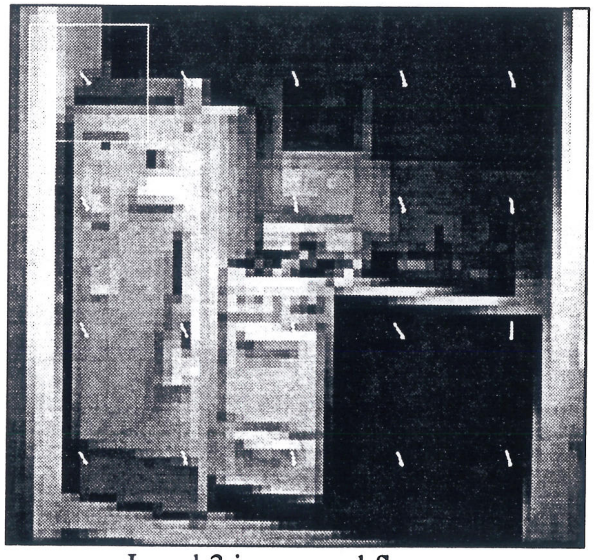
While the flow results are correct in highly textured regions of the image, the flow results are very poor in regions containing insufficient texture. The box in the upper-left corner of each image indicates the size of the region used for flow computation.



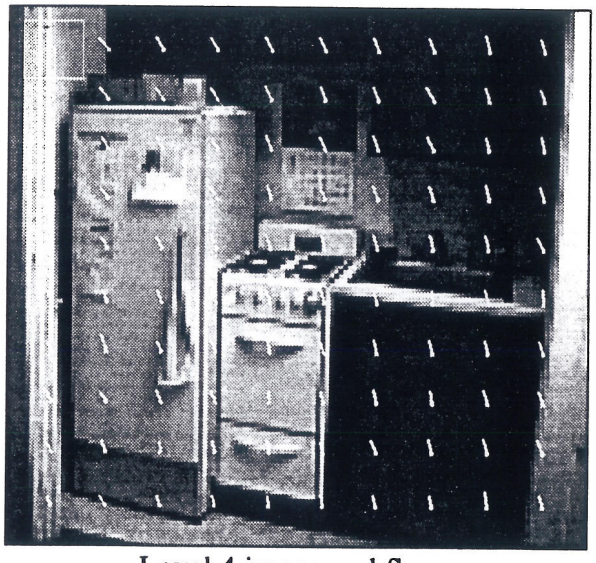
Level 1 image and flow



Level 2 image and flow

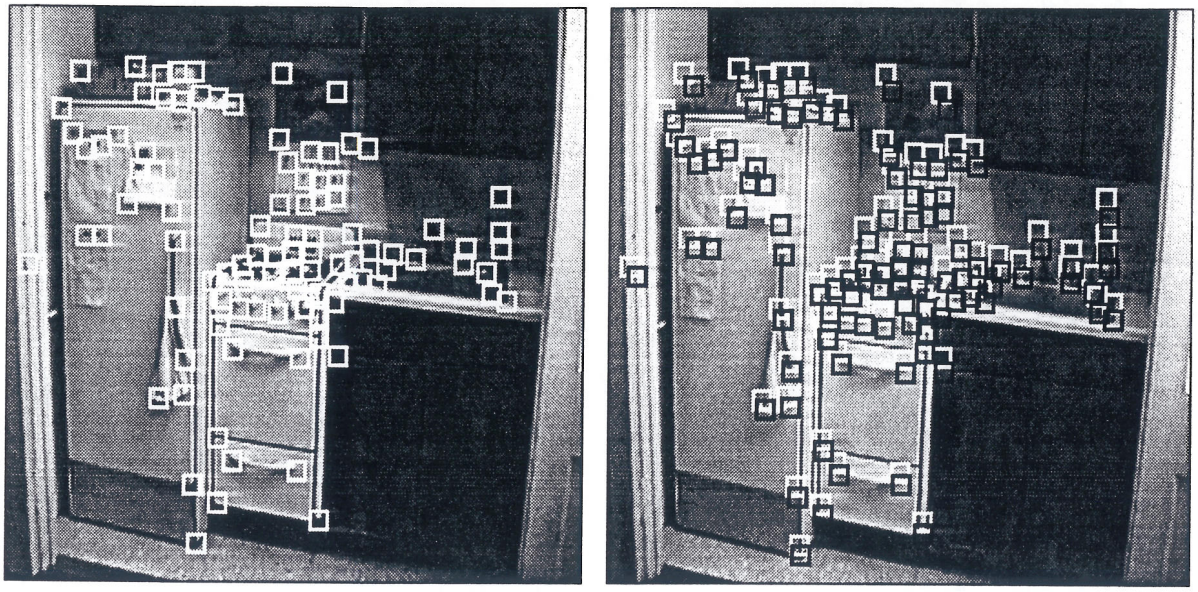


Level 3 image and flow



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

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

There, our first step was to compute T directly from W. We then used the SVD to factor the matrix $W^* = W - T\begin{bmatrix}1 & \cdots & 1\end{bmatrix}$ 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

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

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

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

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

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

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

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

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.

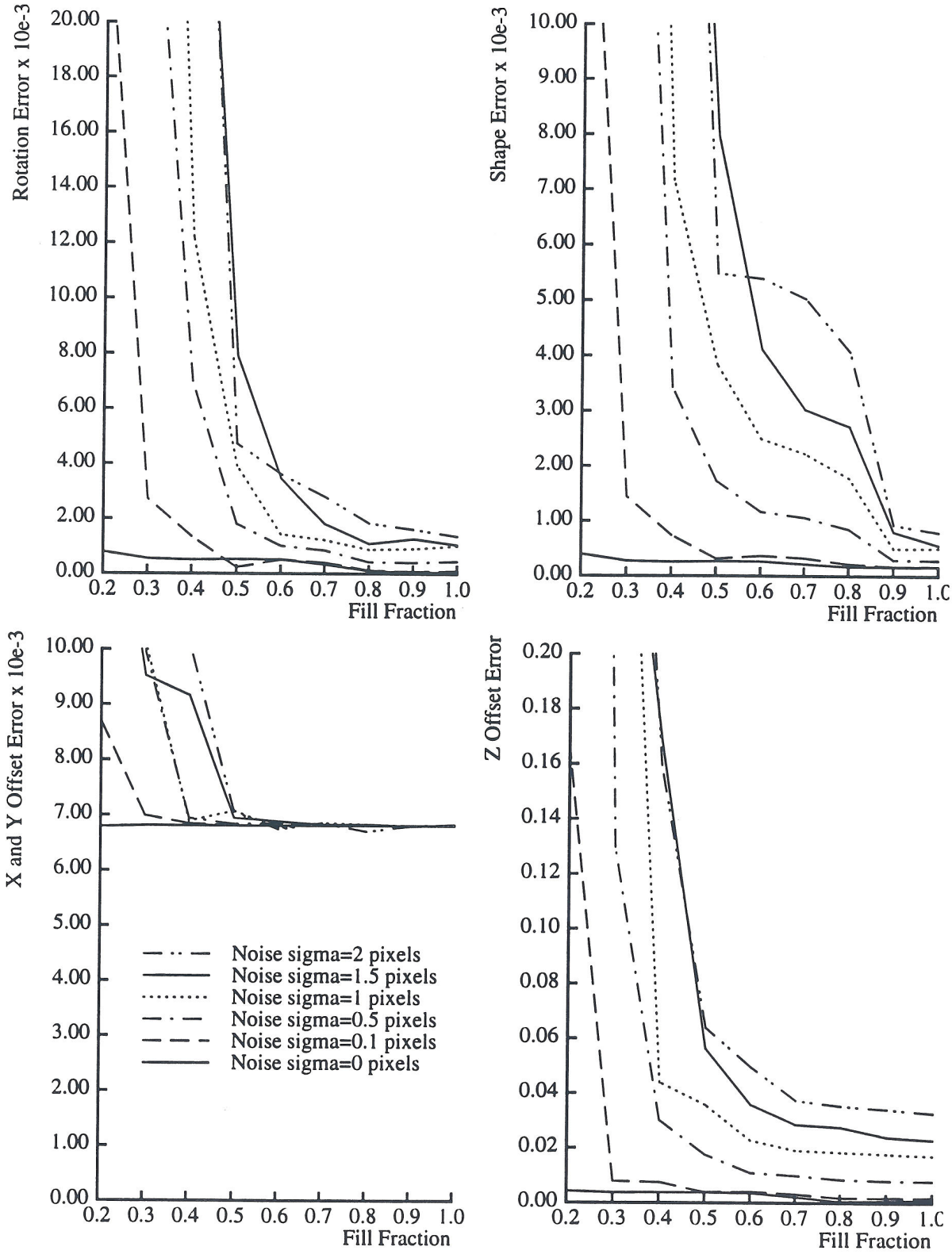


Figure 21. Confidence-Weighted Shape and Motion Recovery by Noise Level, Fill Fraction

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%.

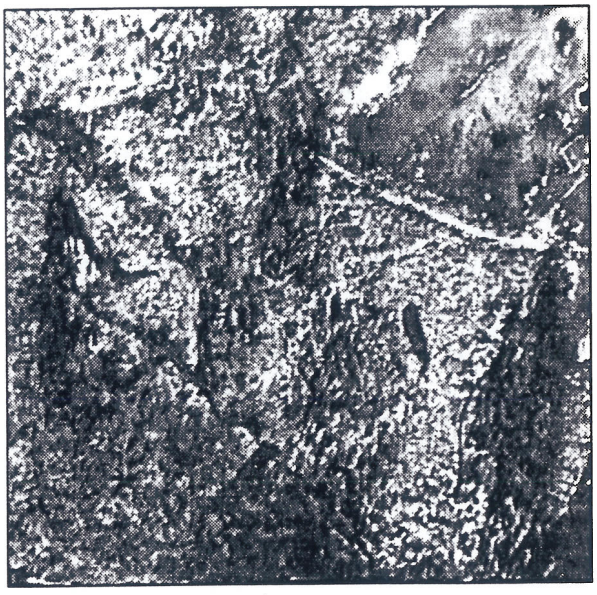
Feature detection required 22 seconds to detect the initial 300 features, and tracking between frames required about 7 seconds per frame on a Sparc 5/85 workstation. New features were detected 26 times during the course of the sequence, whenever the number of visible features dropped below 250. These 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

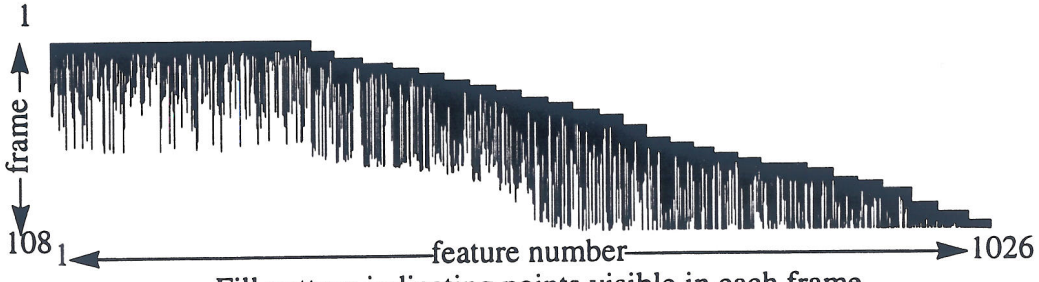
Frame 1

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

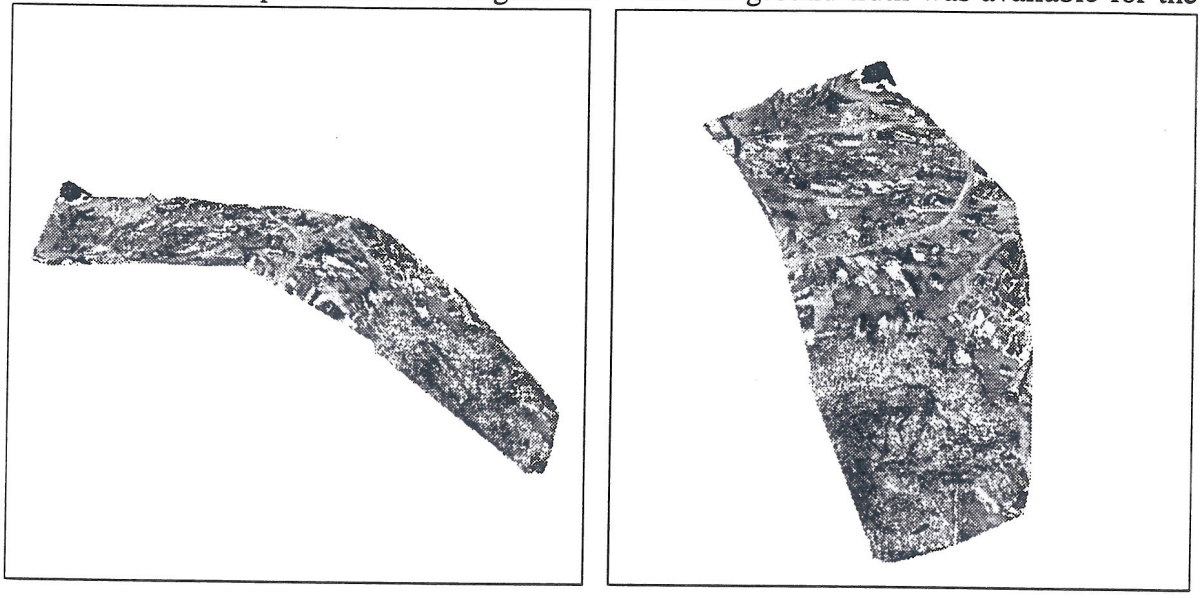


Figure 23. Reconstructed Terrain from Two Viewpoints

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.