# A Paraperspective Factorization Method for Shape and Motion Recovery

Conrad J. Poelman and Takeo Kanade

11 December 1993
CMU-CS-93-219

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

This report supersedes CMU-CS-92-208.

**Abstract**

The factorization method, first developed by Tomasi and Kanade, recovers both the shape of an object and its motion from a sequence of images, using many images and tracking many feature points to obtain highly redundant feature position information. The method robustly processes the feature trajectory information using singular value decomposition (SVD), taking advantage of the linear algebraic properties of orthographic projection. However, an orthographic formulation limits the range of motions the method can accommodate. Paraperspective projection, first introduced by Ohta, is a projection model that closely approximates perspective projection by modelling several effects not modelled under orthographic projection, while retaining linear algebraic properties. Our paraperspective factorization method can be applied to a much wider range of motion scenarios, such as image sequences containing significant translational motion toward the camera or across the image. The method also can accommodate missing or uncertain tracking data, which occurs when feature points are occluded or leave the field of view. We present the results of several experiments which illustrate the method's performance in a wide range of situations, including an aerial image sequence of terrain taken from a low-altitude airplane.

This research was partially supported by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C1465, ARPA Order No. 7597.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Air Force or the U.S. government.

**Keywords:** computer vision, motion, shape, time-varying imagery, factorization method

# 1. Introduction

Recovering the geometry of a scene and the motion of the camera from a stream of images is an important task in a variety of applications, including navigation, robotic manipulation, and aerial cartography. While this is possible in principle, traditional methods have failed to produce reliable results in many situations [2].

Tomasi and Kanade [10][11] developed a robust and efficient method for accurately recovering the shape and motion of an object from a sequence of images, called the *factorization method*. It achieves its accuracy and robustness by applying a well-understood numerical computation, the singular value decomposition (SVD), to a large number of images and feature points, and by directly computing shape without computing the depth as an intermediate step. The method was tested on a variety of real and synthetic images, and was shown to perform well even for distant objects.

The Tomasi-Kanade factorization method, however, assumed an orthographic projection model, since it can be described by linear equations. The applicability of the method is therefore limited to image sequences created from certain types of camera motions. The orthographic model contains no notion of the distance from the camera to the object. As a result, shape reconstruction from image sequences containing large translations toward or away from the camera often produces deformed object shapes, as the method tries to explain the size differences in the images by creating size differences in the object. The method also supplies no estimation of translation along the camera's optical axis, which limits its usefulness for certain tasks.

There exist several perspective approximations which capture more of the effects of perspective projection while remaining linear. Scaled orthographic projection, sometimes referred to as "weak perspective" [4], accounts for the scaling effect of an object as it moves towards and away from the camera. Paraperspective projection, first introduced by Ohta [5] and named by Aloimonos [1], models the position effect (an object is viewed from different angles as it translates across the field of view) as well as the scaling effect.

In this paper, we present a factorization method based on the paraperspective projection model. The paraperspective factorization method is still fast, and robust with respect to noise. It can be applied to a wider realm of situations than the original factorization method, such as sequences containing significant depth translation or containing objects close to the camera, and can be used in applications where it is important to recover the distance to the object in each image, such as navigation.

We begin by describing our camera and world reference frames and introduce the mathematical notation that we use. We review the original factorization method as defined in [11], presenting it in a slightly different manner in order to make its relation to the paraperspective method more apparent. We then present our paraperspective factorization method, followed by an extension which accommodates occlusions. We conclude with the results of several experiments which demonstrate the practicality of our system.

# 2. Problem Description

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. Assume for the time being that we locate $P$ prominent feature points in the first image, and track these points from each image to the next, recording the coordinates $(u_{fp},v_{fp})$ of each point $p$ in each image $f$. Each feature point $p$ that we track corresponds to a single world point, located at position $\mathbf{s}_p$ in some fixed world coordinate system. Each image $f$ was taken at some camera orientation, which we describe by the orthonormal unit vectors $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$, where $\mathbf{i}_f$ and $\mathbf{j}_f$ correspond to the x and y axes of the camera's image plane, and $\mathbf{k}_f$ points along the camera's line of sight. We describe the position of the camera in each frame $f$ by the vector $\mathbf{t}_f$ indicating the camera's focal point. This formulation is illustrated in Figure 1.

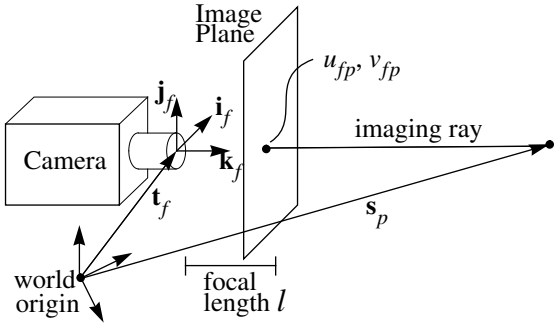



**Figure 1. Coordinate system**

The result of the feature tracker is a set of $P$ feature point coordinates $(u_{fp},v_{fp})$ for each of the $F$ frames of the image sequence. From this information, our goal is to estimate the shape of the object as $\hat{\mathbf{s}}_p$ for each object point, and the motion of the camera as $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, $\hat{\mathbf{k}}_f$ and $\hat{\mathbf{t}}_f$ for each frame in the sequence.

In Section 6 we will relax the requirement that every feature point be visible in every frame, allowing the inclusion of feature points that are observed and tracked through only a portion of the sequence.

# 3. 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 [11].

## 3.1. 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 2. A point $p$ whose location is $\mathbf{s}_p$ will be observed in

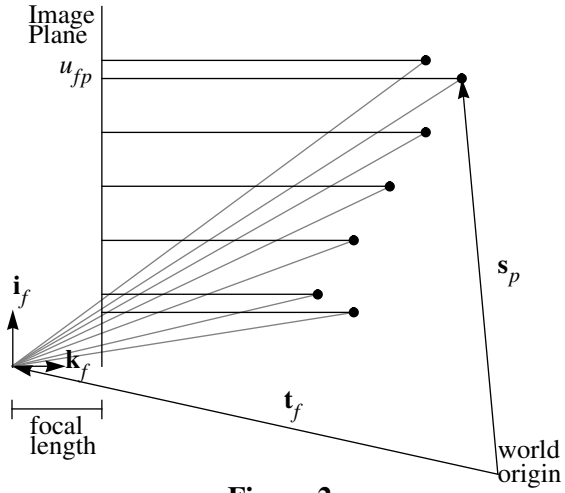



**Figure 2.**
**Orthographic projection in two dimensions**
Dotted lines indicate true perspective projection

frame $f$ at image coordinates $(u_{fp},v_{fp})$, where

$$u_{fp} = \mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) \qquad v_{fp} = \mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) \tag{1}$$

These equations can be rewritten as

$$u_{fp} = \mathbf{m}_f \cdot \mathbf{s}_p + x_f \qquad v_{fp} = \mathbf{n}_f \cdot \mathbf{s}_p + y_f \tag{2}$$

where

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

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

## 3.2. 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. 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 the subtracted from $W$, leaving a "registered" measurement matrix $W^* = W - T\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}$ . Because $W^*$ is the product of a $2F \times 3$ motion matrix $M$ and a $3 \times P$ shape matrix $S$, its rank is at most 3. When noise is present in the input, the $W^*$ will not be exactly of rank 3, so the Tomasi-Kanade factorization method uses the SVD to find the best rank 3 approximation to $W^*$ , factoring it into the product

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

## 3.3. Normalization

The decomposition of equation (8) is only determined up to a linear transformation. Any non-singular $3 \times 3$ 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 $3 \times 3$ 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) repre-

sent 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 $3 \times 3$ 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).

# 4. The Paraperspective Factorization Method

The Tomasi-Kanade factorization method was shown to be computationally inexpensive and highly accurate, but its use of an orthographic projection assumption limited the method's applicability. For example, the method does not produce accurate results when there is significant translation along the camera's optical axis, because orthography does not account for the fact that an object appears larger when it is closer to the camera. We must model this and other perspective effects in order to successfully recover shape and motion in a wider range of situations. We choose an approximation to perspective projection known as *paraperspective projection*, which was introduced by Ohta [5] in order to solve a shape from texture problem. Although the paraperspective projection equations are more complex than those for orthography, their basic form is the same, enabling us develop a method analogous to that developed by Tomasi and Kanade.

## 4.1. Paraperspective Projection

Paraperspective projection closely approximates perspective projection by modelling both the scaling effect (closer objects appear larger than distant ones) and the position effect (objects in the periphery of the image are viewed from a different angle than those near the center of projection [1]), while retaining the linear properties of orthographic projection. The paraperspective projection of an object onto an image, illustrated in Figure 3, involves two steps.

1. An object point is projected along the direction of 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.[1]

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

$$\mathbf{p}' = \mathbf{p} - \frac{\mathbf{p} \cdot \mathbf{n} - d}{\mathbf{r} \cdot \mathbf{n}} \mathbf{r} \ . \tag{12}$$

In frame $f$, each object point $\mathbf{s}_p$ is projected along the direction $\mathbf{c} - \mathbf{t}_f$ (which is 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

1. The scaled orthographic projection model (also known as "weak perspective") is similar to paraperspective projection, except that the direction of the initial projection in step 1 is parallel to the camera's optical axis rather than parallel to the line connecting the object's center of mass to the camera's focal point. This model captures the scaling effect of perspective projection, but not the position effect. (See Appendix I.)

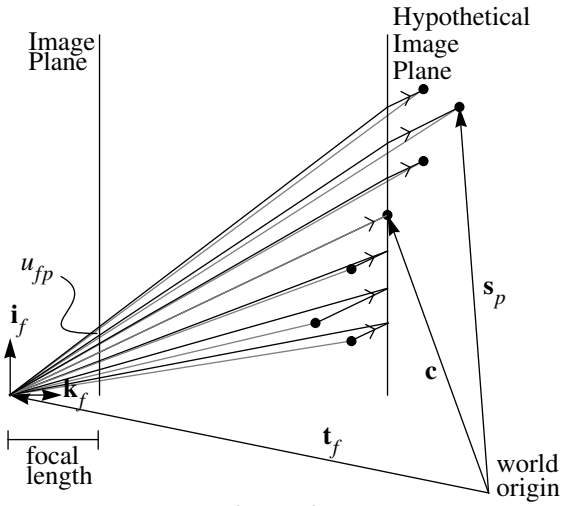



**Figure 3.**
**Paraperspective projection in two dimensions**
Dotted lines indicate true perspective projection
indicate parallel lines.

$$\mathbf{s'}_{fp} = \mathbf{s}_p - \frac{(\mathbf{s}_p \cdot \mathbf{k}_f) - (\mathbf{c} \cdot \mathbf{k}_f)}{(\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{k}_f} (\mathbf{c} - \mathbf{t}_f) \tag{13}$$

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,

$$u_{fp} = \frac{l\mathbf{i}_f}{z_f}(\mathbf{s'}_{fp} - \mathbf{t}_f) \qquad v_{fp} = \frac{l\mathbf{j}_f}{z_f}(\mathbf{s'}_{fp} - \mathbf{t}_f) \quad , \qquad \text{where } z_f = (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{k}_f \tag{14}$$

Substituting (13) into (14) and simplifying gives the general paraperspective equations for $u_{fp}$ and $v_{fp}$

$$\begin{aligned} u_{fp} &= \frac{l}{z_f}\left\{\left[\mathbf{i}_f - \frac{\mathbf{i}_f \cdot (\mathbf{c} - \mathbf{t}_f)}{z_f}\mathbf{k}_f\right] \cdot (\mathbf{s}_p - \mathbf{c}) + (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{i}_f\right\} \\ v_{fp} &= \frac{l}{z_f}\left\{\left[\mathbf{j}_f - \frac{\mathbf{j}_f \cdot (\mathbf{c} - \mathbf{t}_f)}{z_f}\mathbf{k}_f\right] \cdot (\mathbf{s}_p - \mathbf{c}) + (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{j}_f\right\} \end{aligned} \tag{15}$$

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

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

$$\mathbf{c} = \frac{1}{P}\sum_{p=1}^{P} \mathbf{s}_p = 0 \quad . \tag{16}$$

This reduces (15) to

$$\begin{aligned} u_{fp} &= \frac{1}{z_f}\left\{\left[\mathbf{i}_f + \frac{\mathbf{i}_f \cdot \mathbf{t}_f}{z_f}\mathbf{k}_f\right]\cdot \mathbf{s}_p - (\mathbf{t}_f \cdot \mathbf{i}_f)\right\} \\ v_{fp} &= \frac{1}{z_f}\left\{\left[\mathbf{j}_f + \frac{\mathbf{j}_f \cdot \mathbf{t}_f}{z_f}\mathbf{k}_f\right]\cdot \mathbf{s}_p - (\mathbf{t}_f \cdot \mathbf{j}_f)\right\} \end{aligned} \tag{17}$$

These equations can be rewritten as

$$u_{fp} = \mathbf{m}_f \cdot \mathbf{s}_p + x_f \qquad v_{fp} = \mathbf{n}_f \cdot \mathbf{s}_p + y_f \tag{18}$$

where

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \tag{19}$$

$$x_f = -\frac{\mathbf{t}_f \cdot \mathbf{i}_f}{z_f} \qquad y_f = -\frac{\mathbf{t}_f \cdot \mathbf{j}_f}{z_f} \tag{20}$$

$$\mathbf{m}_f = \frac{\mathbf{i}_f - x_f \mathbf{k}_f}{z_f} \qquad \mathbf{n}_f = \frac{\mathbf{j}_f - y_f \mathbf{k}_f}{z_f} \tag{21}$$

Notice that equation (18) 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.

## 4.2. Paraperspective Decomposition

We can combine equation (18), for all points $p$ from 1 to $P$, and all frames $f$ from 1 to $F$, into the single matrix equation

$$\begin{bmatrix} u_{11} & \cdots & u_{1P} \\ \cdots & \cdots & \cdots \\ u_{F1} & \cdots & u_{FP} \\ v_{11} & \cdots & v_{1P} \\ \cdots & \cdots & \cdots \\ v_{F1} & \cdots & v_{FP} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1 \\ \cdots \\ \mathbf{m}_F \\ \mathbf{n}_1 \\ \cdots \\ \mathbf{n}_F \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_P \end{bmatrix} + \begin{bmatrix} x_1 \\ \cdots \\ x_F \\ y_1 \\ \cdots \\ y_F \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \quad , \tag{22}$$

or in short

$$W = MS + T\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \ , \tag{23}$$

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 (16) and (18) we can write

$$\begin{aligned} \sum_{p=1}^{P} u_{fp} &= \sum_{p=1}^{P} (\mathbf{m}_f \cdot \mathbf{s}_p + x_f) = \mathbf{m}_f \cdot \sum_{p=1}^{P} \mathbf{s}_p + P x_f = P x_f \\ \sum_{p=1}^{P} v_{fp} &= \sum_{p=1}^{P} (\mathbf{n}_f \cdot \mathbf{s}_p + y_f) = \mathbf{n}_f \cdot \sum_{p=1}^{P} \mathbf{s}_p + P y_f = P y_f \end{aligned} \tag{24}$$

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

$$x_f = \frac{1}{P} \sum_{p=1}^{P} u_{fp} \qquad y_f = \frac{1}{P} \sum_{p=1}^{P} v_{fp} \quad . \tag{25}$$

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

$$W^* = W - T\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} = MS \quad . \tag{26}$$

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

$$W^* = \hat{M}\hat{S} \, , \tag{27}$$

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

## 4.3. Paraperspective Normalization

Just as in the orthographic case, the decomposition of $W^*$ into the product of $\hat{M}$ and $\hat{S}$ by equation (27) is only determined up to a linear transformation matrix $A$. Again, we determine this matrix $A$ by observing that the rows of the motion matrix $M$ (the $\mathbf{m}_f$ and $\mathbf{n}_f$ vectors) must be of a certain form. Taking advantage of the fact that $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ are unit vectors, from equation (21) we observe that

$$|\mathbf{m}_f|^2 = \frac{1 + x_f^2}{z_f^2} \qquad |\mathbf{n}_f|^2 = \frac{1 + y_f^2}{z_f^2} \quad . \tag{28}$$

We know the values of $x_f$ and $y_f$ from our initial registration step, but we do not know the value of the depth $z_f$. Thus we cannot impose individual constraints on the magnitudes of $\mathbf{m}_f$ and $\mathbf{n}_f$ as was done in the orthographic factorization method. Instead we adopt the following constraint on the magnitudes of $\mathbf{m}_f$ and $\mathbf{n}_f$

$$\frac{|\mathbf{m}_f|^2}{1+x_f^2} = \frac{|\mathbf{n}_f|^2}{1+y_f^2} \qquad \left( = \frac{1}{z_f^2} \right) . \tag{29}$$

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

There is also a constraint on the angle relationship of $\mathbf{m}_f$ and $\mathbf{n}_f$. From equation (21), and the knowledge that $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ are orthogonal unit vectors,

$$\mathbf{m}_f \cdot \mathbf{n}_f = \frac{\mathbf{i}_f - x_f \mathbf{k}_f}{z_f} \cdot \frac{\mathbf{j}_f - y_f \mathbf{k}_f}{z_f} = \frac{x_f y_f}{z_f^2} . \tag{30}$$

The problem with this constraint is that, again, $z_f$ is unknown. We could use either of the two values given in equation (29) 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

$$\mathbf{m}_f \cdot \mathbf{n}_f = x_f y_f \frac{1}{2} \left( \frac{|\mathbf{m}_f|^2}{1+x_f^2} + \frac{|\mathbf{n}_f|^2}{1+y_f^2} \right) . \tag{31}$$

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 (29) and (31) 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

$$|\mathbf{m}_1| = 1 . \tag{32}$$

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

Equations (29), (31), and (32) gives us $2F+1$ equations, which are the paraperspective version of the *metric constraints*. We compute the $3 \times 3$ 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 $3 \times 3$ matrix $Q = A^T A$. We use the metric constraints to compute $Q$, compute its Jacobi Transformation $Q = L \Lambda L^T$, where $\Lambda$ is the diagonal eigenvalue matrix, and as long as $Q$ is positive definite, $A = (L \Lambda^{1/2})^T$.

## 4.4. Paraperspective 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 (21) we see that

$$\hat{\mathbf{i}}_f = z_f\mathbf{m}_f + x_f\hat{\mathbf{k}}_f \qquad \hat{\mathbf{j}}_f = z_f\mathbf{n}_f + y_f\hat{\mathbf{k}}_f \quad . \tag{33}$$

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

$$\begin{aligned} \hat{\mathbf{i}}_f \times \hat{\mathbf{j}}_f &= (z_f\mathbf{m}_f + x_f\hat{\mathbf{k}}_f) \times (z_f\mathbf{n}_f + y_f\hat{\mathbf{k}}_f) = \hat{\mathbf{k}}_f \\ \left|\hat{\mathbf{i}}_f\right| &= \left|z_f\mathbf{m}_f + x_f\hat{\mathbf{k}}_f\right| = 1 \\ \left|\hat{\mathbf{j}}_f\right| &= \left|z_f\mathbf{n}_f + y_f\hat{\mathbf{k}}_f\right| = 1 \end{aligned} \tag{34}$$

Again, we do not know a value for $z_f$, but using the relations specified in equation (29) and the additional knowledge that $\left|\hat{\mathbf{k}}_f\right| = 1$, equation (34) can be reduced to

$$G_f\hat{\mathbf{k}}_f = H_f\,, \tag{35}$$

where

$$G_f = \begin{bmatrix} (\tilde{\mathbf{m}}_f \times \tilde{\mathbf{n}}_f) \\ \tilde{\mathbf{m}}_f \\ \tilde{\mathbf{n}}_f \end{bmatrix} \qquad H_f = \begin{bmatrix} 1 \\ -x_f \\ -y_f \end{bmatrix} \tag{36}$$

$$\tilde{\mathbf{m}}_f = \sqrt{1 + x_f^2}\,\frac{\mathbf{m}_f}{|\mathbf{m}_f|} \qquad \tilde{\mathbf{n}}_f = \sqrt{1 + y_f^2}\,\frac{\mathbf{n}_f}{|\mathbf{n}_f|} \tag{37}$$

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

$$\hat{\mathbf{k}}_f = G_f^{-1}H_f \tag{38}$$

and then compute

$$\hat{\mathbf{i}}_f = \tilde{\mathbf{n}}_f \times \hat{\mathbf{k}}_f \qquad \hat{\mathbf{j}}_f = \hat{\mathbf{k}}_f \times \tilde{\mathbf{m}}_f \quad . \tag{39}$$

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 (39). 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 (29). 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 (19) and (20).

## 4.5. Solution Ambiguity Removal

In order to solve for the shape and motion at the end of Section 4.3., we computed the matrix $Q = A^T A$ that best satisfied the metric constraints, and then computed $A$ from $Q$. There is a sign ambiguity in this process, since any matrix of the form $A \begin{bmatrix} \pm 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & \pm 1 \end{bmatrix}$ produces the same matrix $Q$ when multiplied by its transpose. Thus there are actually several equally plausible motion and shape matrices, since changing the sign of a column of $M$ and the corresponding row of $S$ still produces a solution that satisfies the metric constraints equally well. This sign ambiguity in the first two columns of $M$ was removed when we aligned the world coordinate axes with the first frame's camera axes, at the end of Section 4.4. 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.

Geometrically, this ambiguity arises because paraperspective projection does not account for any different perspective deformation within the object itself. It is not possible to distinguish the "front" of the object from the "back" of the object, as can be seen from Figure 4(a). However, in real scenarios there will be additional queues due to occlusion information as in Figure 4(b) and (c), or due to perspective distortion of the object, as in Figure 4(d), if the object is not too distant from the camera. Simple methods based on either of these phenomena should be sufficient to determine which of the two solutions given by the paraperspective factorization method is consistent with the image data.

## 4.6. Paraperspective Projection as an Approximation to Perspective Projection

In Section 4.1., 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 $(u_{fp}, v_{fp})$, where

$$u_{fp} = l\frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_{fp}}$$
$$v_{fp} = l\frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_{fp}} \tag{40}$$

$$z_{fp} = \mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) \tag{41}$$

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

We define the term

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \tag{42}$$

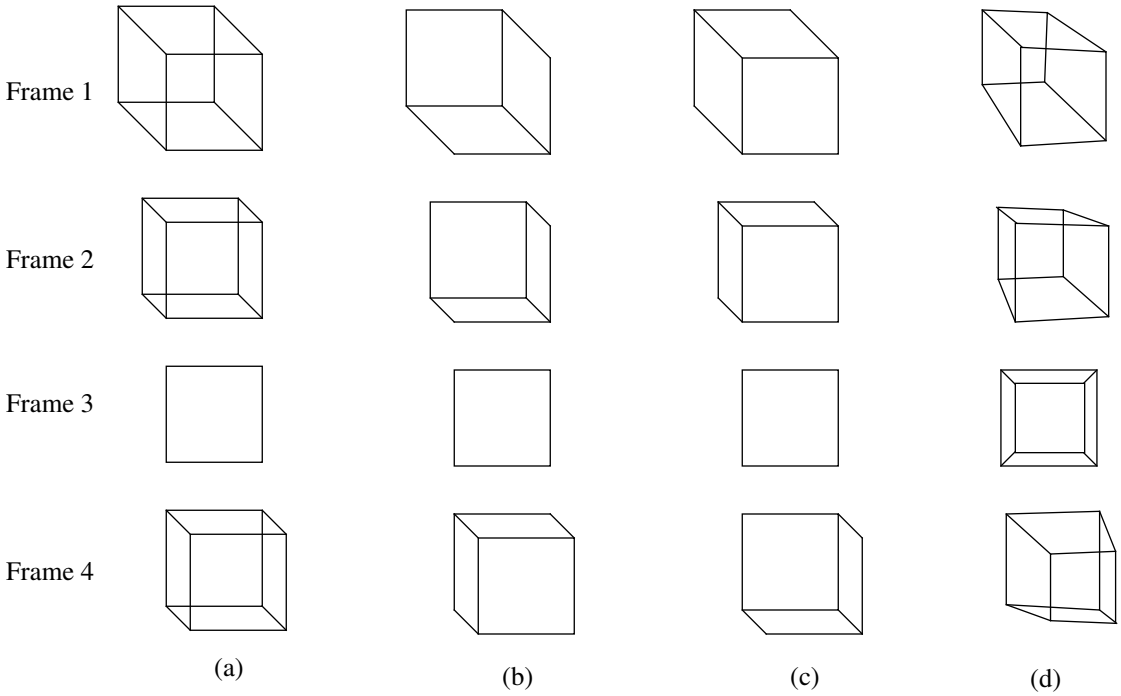



**Figure 4 Ambiguity of Solution**

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

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

$$z_{fp} \approx z_f \tag{43}$$

yielding

$$\begin{aligned} u_{fp} &= \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(z_{fp} - z_f) + \frac{1}{2}\frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(z_{fp} - z_f)^2 + \ldots \\ v_{fp} &= \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(z_{fp} - z_f) + \frac{1}{2}\frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(z_{fp} - z_f)^2 + \ldots \end{aligned} \tag{44}$$

We combine equations (41) and (42) to determine that $z_{fp} - z_f = \mathbf{k}_f \cdot \mathbf{s}_p$, and substitute this into equation (44) to produce

$$\begin{aligned} u_{fp} &= \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p) + \frac{1}{2}\frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(\mathbf{k}_f \cdot \mathbf{s}_p)^2 + \ldots \\ v_{fp} &= \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p) + \frac{1}{2}\frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(\mathbf{k}_f \cdot \mathbf{s}_p)^2 + \ldots \end{aligned} \tag{45}$$

Ignoring all but the first term of the Taylor series yields the equations for scaled orthographic projection (See Appendix I.) 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 (45) to

$$\begin{aligned} u_{fp} &= \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{i}_f \cdot (-\mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p) \\ v_{fp} &= \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{j}_f \cdot (-\mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p) \end{aligned} \tag{46}$$

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

$$\begin{aligned} u_{fp} &= \frac{1}{z_f}\left(\mathbf{i}_f \cdot \mathbf{s}_p + \frac{\mathbf{i}_f \cdot \mathbf{t}_f}{z_f}(\mathbf{k}_f \cdot \mathbf{s}_p) - (\mathbf{i}_f \cdot \mathbf{t}_f)\right) \\ v_{fp} &= \frac{1}{z_f}\left(\mathbf{j}_f \cdot \mathbf{s}_p + \frac{\mathbf{j}_f \cdot \mathbf{t}_f}{z_f}(\mathbf{k}_f \cdot \mathbf{s}_p) - (\mathbf{j}_f \cdot \mathbf{t}_f)\right) \end{aligned} \tag{47}$$

These equations are equivalent to the paraperspective projection equations given by equation (17).

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 4.1. 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.

# 5. Comparison of Methods using Synthetic Data

In this section we compare the performance of our new paraperspective factorization method with the previous orthographic factorization method. The comparison also includes a factorization method based on scaled orthographic projection (also known as "weak perspective"), which models the scaling effect of perspective projection but not the position effect, in order to demonstrate the importance of modelling the position effect for objects at close range. (See Appendix I.) Our results show that the paraperspective factorization method is a vast improvement over the orthographic method, and underscore the importance of modelling both the scaling and position effects.

## 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 each sequence, the object 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.

## 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 $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{k}}_f$. 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.

## 5.3. Discussion of Results

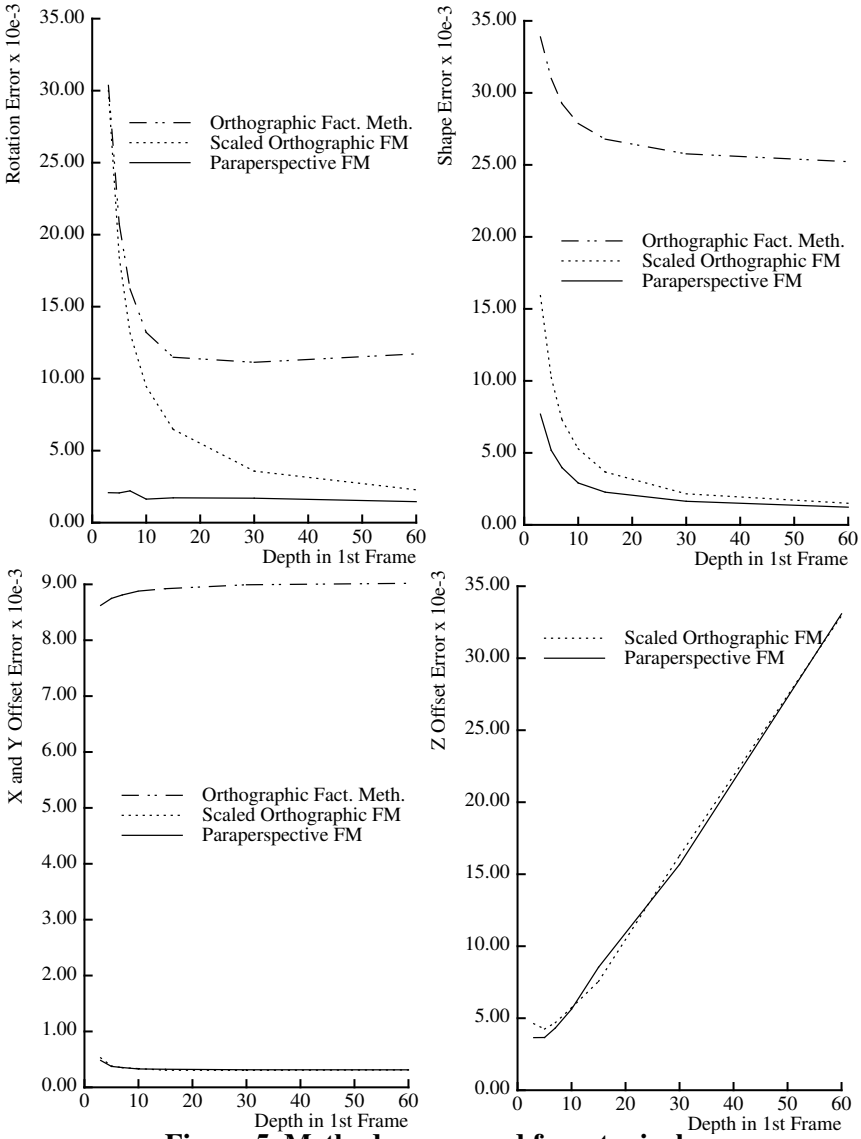



**Figure 5. Methods compared for a typical case**

noise standard deviation = 2 pixels

Figure 5 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 modelling the position effect when objects are near the camera.

In other experiments in which the object was centered in the image and there was no translation across the field of view, the paraperspective method and the scaled orthographic method performed equally well, as we would expect since such image sequences contain no position effects. Similarly, we found that when the object remained centered in the image and there was no depth translation, the orthographic factorization method performed very well, and the paraperspective factorization method provided no significant improvement since such sequences contain neither scaling effects nor position effects.

To examine the impact of those effects of perspective projection which are not modelled by paraperspective projection, we implemented an iterative method which refines the results of the paraperspective method using a perspective projection model. (See Appendix II.) Computing this refined solution required more than ten times as long as computing the initial paraperspective results. We tested the method on the same sequences that were tested to produce Figure 5, and found the resulting shape and motion errors to be nearly invariant to depth. The perspective refinement method only minimally improved the motion results over the paraperspective solution. However, the shape results were significantly improved for those cases in which the depth was less than 30. This implies that unmodelled 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.

## 5.4. Analysis of Paraperspective Method using Synthetic Data

Now that we have shown the advantages of the paraperspective factorization method over the previous 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 6, 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.

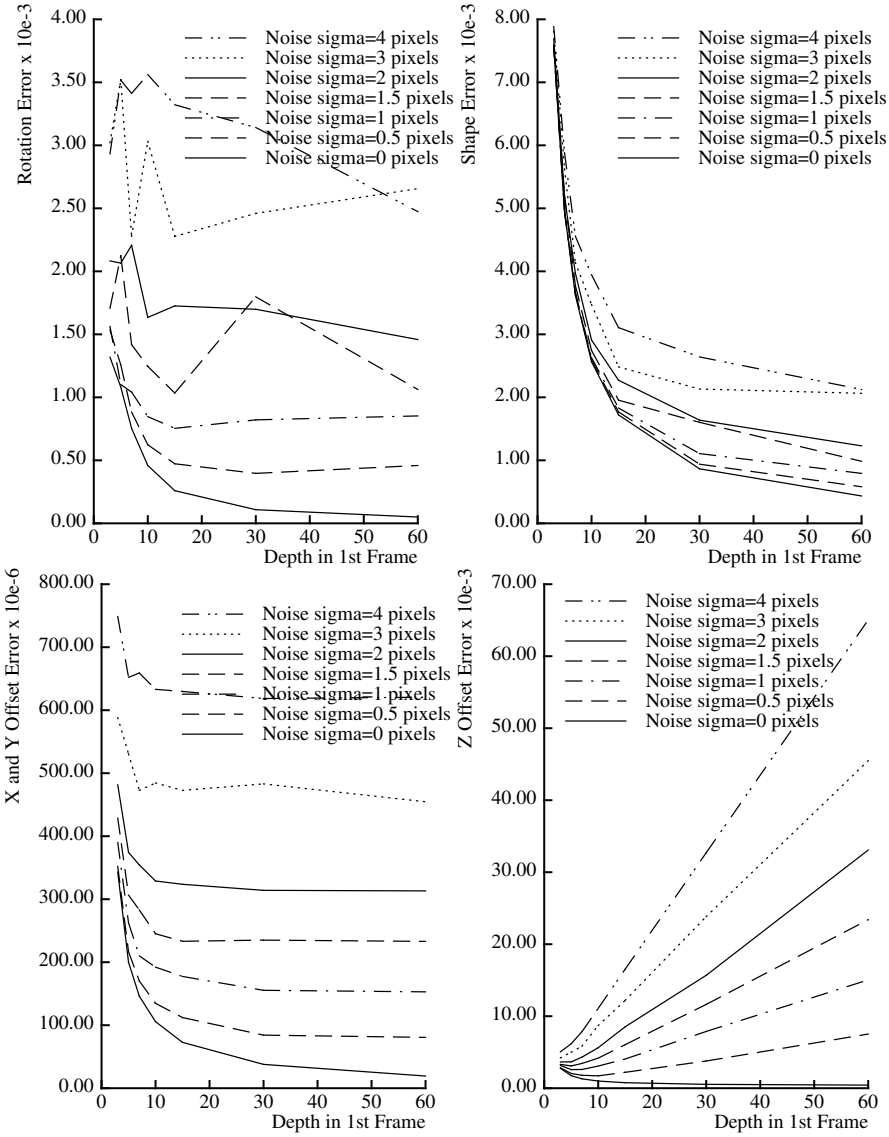



**Figure 6. Paraperspective shape and motion recovery by noise level**

# 6. Accommodating Occlusion and Uncertain Tracking Data

So far, we have assumed that all of the entries of the measurement matrix are known and are equally reliable. In real image sequences, this is not 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 unmodelled 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, and modifying our method to weight each measurement by its corresponding confidence value. If a feature is not observed in some frames, the confidence values corresponding to those measurements are set to zero.

## 6.1. Confidence-Weighted Formulation and Solution Method

We can view the decomposition step of Section 4.2. as a way, given the measurement matrix $W$, to compute an $\hat{M}$, $\hat{S}$, and $T$ that satisfy the equation

$$W = \hat{M}\hat{S} + T\begin{bmatrix}1 & \dots & 1\end{bmatrix} \tag{48}$$

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

$$\varepsilon_0 = \sum_{r=1}^{2F} \sum_{p=1}^{P} \left(W_{rp} - \left(\hat{M}_{r1}\hat{S}_{1p} + \hat{M}_{r2}\hat{S}_{2p} + \hat{M}_{r3}\hat{S}_{3p} + T_r\right)\right)^2 \tag{49}$$

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

$$\varepsilon = \sum_{r=1}^{2F} \sum_{p=1}^{P} \gamma^2_{rp} \left( W_{rp} - (\hat{M}_{r1}\hat{S}_{1p} + \hat{M}_{r2}\hat{S}_{2p} + \hat{M}_{r3}\hat{S}_{3p} + T_r) \right)^2 \tag{50}$$

Once we have solved this minimization problem for $\hat{M}$, $\hat{S}$, and $T$, we can proceed with the normalization step and the rest of the shape and motion recovery process precisely as before.

There is sufficient mathematical constraint to compute $\hat{M}$, $\hat{S}$, and $T$, provided the number of known elements of $W$ exceeds the total number of variables ( $8F + 3P$ ). The minimization of equation (50) 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. We use a variant of an algorithm suggested by Ruhe and Wedin [8] 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 $\varepsilon$. Then holding $\hat{M}$ and $T$ fixed, we solve for the $\hat{S}$ which minimizes $\varepsilon$. 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 $\varepsilon$.

To compute $\hat{M}$ and $T$ for a given $\hat{S}$, we first rewrite the minimization of equation (50) as

$$\varepsilon = \sum_{r=1}^{2F} \varepsilon_r , \qquad \text{where} \qquad \varepsilon_r = \sum_{p=1}^{P} \gamma^2_{rp} \left( W_{rp} - (\hat{M}_{r1}\hat{S}_{1p} + \hat{M}_{r2}\hat{S}_{2p} + \hat{M}_{r3}\hat{S}_{3p} + T_r) \right)^2 . \tag{51}$$

For a fixed $\hat{S}$, the total error $\varepsilon$ can be minimized by independently minimizing each error $\varepsilon_r$, since no variable appears in more than one $\varepsilon_r$ equation. Each $\varepsilon_r$ 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

$$\begin{bmatrix} \hat{S}_{11}\gamma_{r1} & \hat{S}_{21}\gamma_{r1} & \hat{S}_{31}\gamma_{r1} & \gamma_{r1} \\ \cdots & \cdots & \cdots & \cdots \\ \hat{S}_{1P}\gamma_{rP} & \hat{S}_{2P}\gamma_{rP} & \hat{S}_{3P}\gamma_{rP} & \gamma_{rP} \end{bmatrix} \begin{bmatrix} \hat{M}_{r1} \\ \hat{M}_{r2} \\ \hat{M}_{r3} \\ T_r \end{bmatrix} = \begin{bmatrix} W_{r1}\gamma_{r1} \\ \cdots \\ W_{rP}\gamma_{rP} \end{bmatrix} \tag{52}$$

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

$$\begin{bmatrix} \hat{M}_{11}\gamma_{1p} & \hat{M}_{12}\gamma_{1p} & \hat{M}_{13}\gamma_{1p} \\ \cdots & \cdots & \cdots \\ \hat{M}_{2F,1}\gamma_{2F,p} & \hat{M}_{2F,2}\gamma_{2F,p} & \hat{M}_{2F,3}\gamma_{2F,p} \end{bmatrix} \begin{bmatrix} \hat{S}_{1p} \\ \hat{S}_{2p} \\ \hat{S}_{3p} \end{bmatrix} = \begin{bmatrix} (W_{1p} - T_1)\,\gamma_{1p} \\ \cdots \\ (W_{2F,p} - T_{2F})\,\gamma_{2F,p} \end{bmatrix} \tag{53}$$

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 $\gamma_{rp}$ were randomly given values ranging from 1 to 10, the method generally converged in 10 or fewer iterations; and with a $\gamma$ 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 sometimes 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 [11]. We first find a subset of rows and columns of $W$ for which all of the confidences are non-zero, and solve for the corresponding rows of $\hat{M}$ and $T$ and columns of $\hat{S}$ by running the iterative method starting with a random initial value. As indicated above, 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 (52) 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 (53) 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.

## 6.2. 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[1]. Figure 7 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 5.

Errors in the recovered motion only increase slightly as the fill fraction decreases from 1.0

1. Creating a confidence matrix that has a given fill fraction and a fairly realistic fill pattern (the arrangement of zero and non-zero entries in the matrix) is a non-trivial task. We first create the synthetic feature tracks resulting from the given object and motion, by detecting when points become occluded by some surface of the object. These feature tracks are then extended or shortened until the desired fill fraction is achieved.

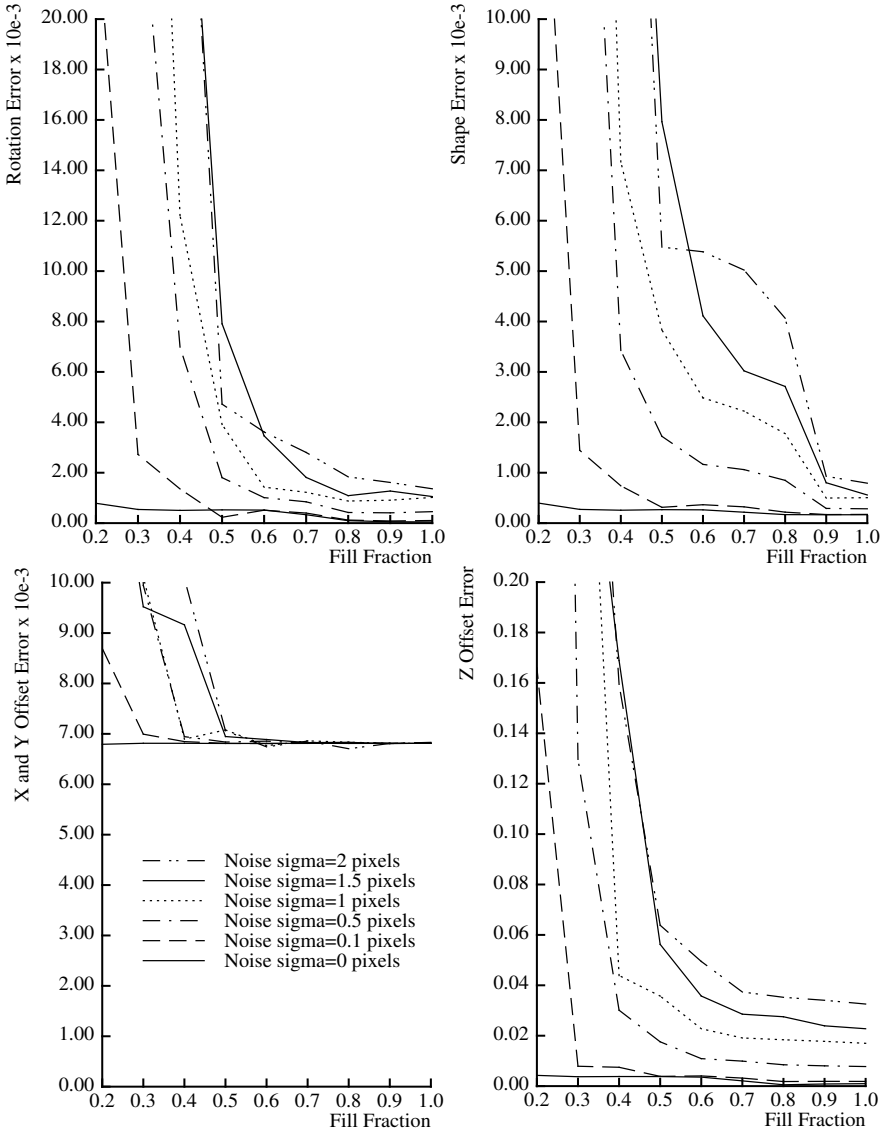



**Figure 7. Confidence-Weighted Shape and Motion Recovery by Noise Level, Fill Fraction**

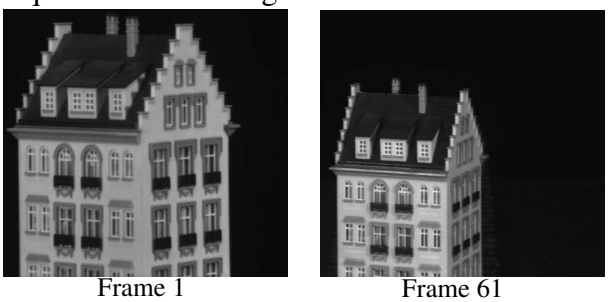
to 0.5. At a very low noise level, such as 0.1 pixels, this behavior continues down to a fill fraction of 0.3. When the fill fraction is decreased below this range, however, the error in the recovered motion increases very sharply. The shape results appear more sensitive to decreased fill fractions; 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 (50) is still overconstrained at these lower fill fractions, apparently there is insufficient redundancy to overcome the effects of the noise in the data.

# 7. Shape and Motion Recovery from Real Image Sequences

We tested the paraperspective factorization method on two real image sequences - a laboratory experiment in which a small model building was imaged, and an aerial sequence taken from a low-altitude plane using a hand-held video camera. Both sequences contain significant perspective effects, due to translations along the optical axis and across the field of view. We implemented a system to automatically identify and track features, based on [11] and [3]. This tracker computes the position of a square feature window by minimizing the sum of the squares of the intensity difference over the feature window from one image to the next.

## 7.1. Hotel Model 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 (see Figure 8). The feature tracker automatically identified and tracked 197 points throughout the sequence of 181 images.



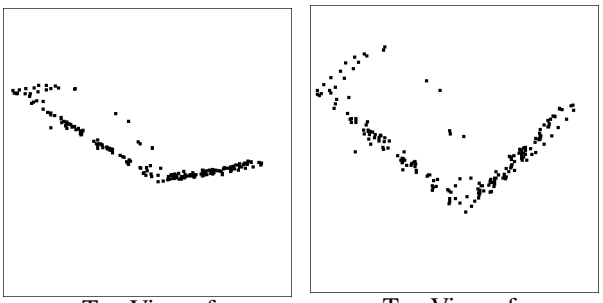
Frame 1 Frame 61

Frame 121 Frame 151

**Figure 8. 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 9) 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 produced an accurate shape and accurate motion.

Several features in the sequence were poorly tracked, and as a result their recovered 3D



Top View of Orthographic Solution

Top View of Paraperspective Solution

**Figure 9. Comparison of Orthographic and Paraperspective Shape Results**

positions were incorrect. While they did not disrupt the overall solution greatly, we found 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 10, 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.

## 7.2. 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 11.

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. The original feature tracker could not track motions of more than approximately 3 pixels, so we implemented a coarse-to-fine tracker. The tracker first estimated the translation using low resolution images, and then refined that value using the same methods as the initial tracker.

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 11) 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.

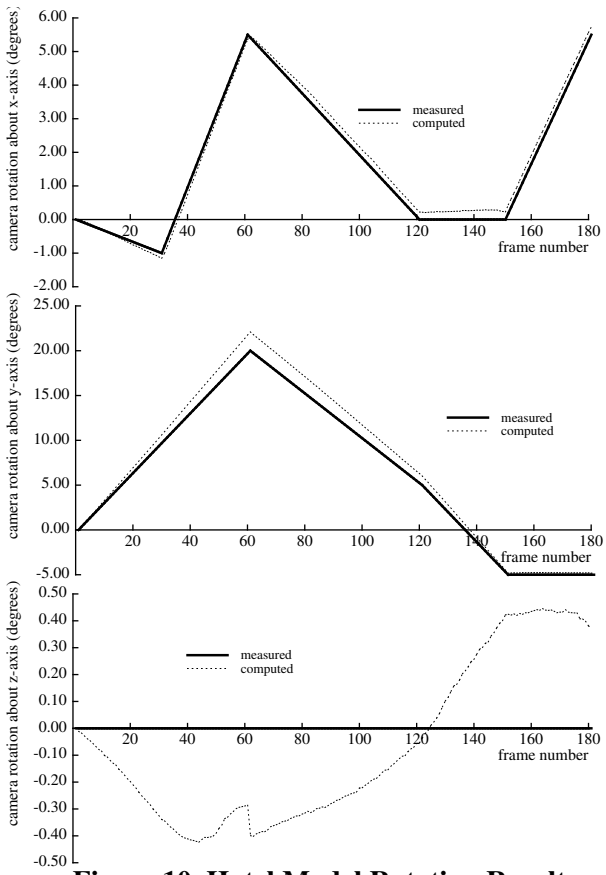
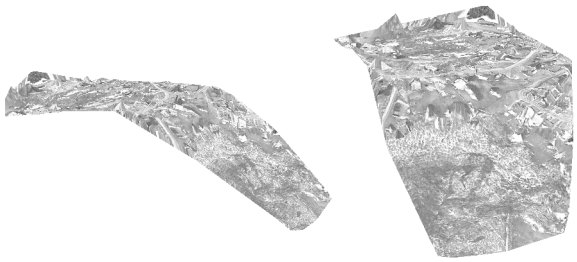



**Figure 10. Hotel Model Rotation Results**

The confidence-weighted paraperspective factorization method was used to recover the shape of the terrain and the motion of the airplane. Two views of the reconstructed terrain map are shown in Figure 12. While no ground-truth was available for the shape or the



Two views of reconstructed terrain

**Figure 12. Reconstructed Terrain**

motion, we observed that the terrain was qualitatively correct, capturing the flat residential area and the steep hillside as well, and that the recovered positions of features on buildings were elevated from the surrounding terrain.

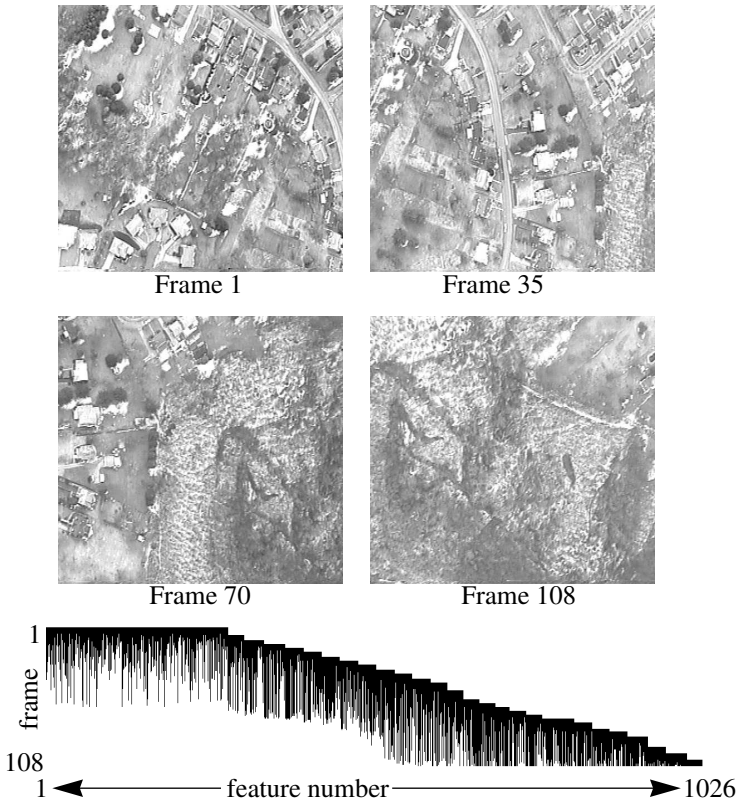



**Figure 11. Aerial Image Sequence**

# 8. Conclusions

The principle that the measurement matrix has rank 3, as put forth by Tomasi and Kanade in [10], was dependent on the use of an orthographic projection model. We have shown in this paper that this important result also holds for the case of paraperspective projection, which closely approximates perspective projection. We have devised a paraperspective factorization method based on this model, which uses different metric constraints and motion recovery techniques, but retains many of the features of the original factorization method.

In image sequences in which the object being viewed translates significantly toward or away from the camera or across the camera's field of view, the paraperspective factorization method performs significantly better than the orthographic method. The paraperspective factorization method also computes the distance from the camera to the object in each image and can accommodate missing or uncertain tracking data, which enables its use in a variety of applications.

The C implementation of the paraperspective factorization method required about 20-24 seconds to solve a system of 60 frames and 60 points on a Sun 4/65, with most of this time spent computing the singular value decomposition of the measurement matrix. Running times for the confidence-weighted method were comparable, but varied depending on the number of iterations required for the method to converge. While this is not sufficient for real-time use, we hope to develop a faster implementation.

The confidence-weighted factorization method performs well when the fill fraction of the confidence matrix is high or when the noise level is very low. In future work we hope to determine more precisely in what circumstances this method can be expected to perform well, and to investigate ways to extend its range so that the method can be applied to longer sequences in which the fill fraction is much lower.

# Acknowledgments

The authors wish to thank Radu Jasinschi for pointing out the existence of the paraperspective projection model and suggesting its applicability to the factorization method. Additional thanks goes to Carlo Tomasi and Toshihiko Morita for their helpful and insightful comments.

# 9. References

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

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

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

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

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

**[6]** Conrad J. Poelman and Takeo Kanade, *A Paraperspective Factorization Method for Shape and Motion Recovery*, Technical Report CMU-CS-92-208, Carnegie Mellon University, Pittsburgh, PA, October 1992.

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

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

**[9]** Camillo Taylor, David Kriegman, and P. Anandan, *Structure and Motion From Multiple Images: A Least Squares Approach*, IEEE Workshop on Visual Motion, pp. 242-248, October 1991.

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

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

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

# Appendix I. The Scaled Orthographic Factorization Method

Scaled orthographic projection, also known as "weak perspective" [4], 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 relative to the size of the object.

## I.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 (see Figure 13).

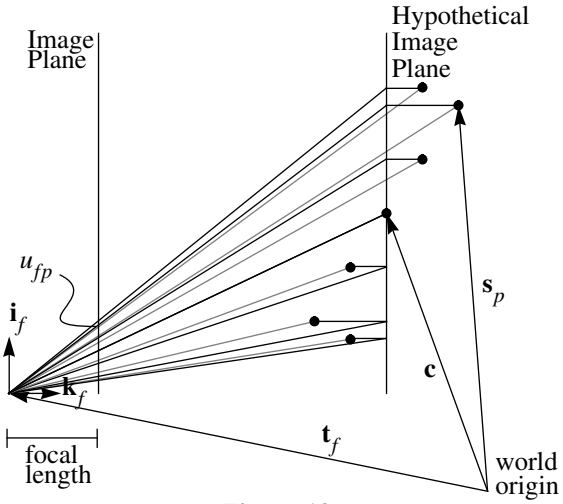



**Figure 13**
**Scaled Orthographic Projection in two dimensions**

Dotted lines indicate true perspective projection
indicate parallel lines.

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

$$z_f = (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{k}_f. \tag{54}$$

Thus 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 $z_f$.

$$\begin{aligned} u_{fp} &= \frac{l}{z_f}\left(\mathbf{i}_f \cdot \left(\mathbf{s}_p - \mathbf{t}_f\right)\right) \\ v_{fp} &= \frac{l}{z_f}\left(\mathbf{j}_f \cdot \left(\mathbf{s}_p - \mathbf{t}_f\right)\right) \end{aligned} . \tag{55}$$

To simplify the equations we assume unit focal length, $l = 1$. 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

$$u_{fp} = \mathbf{m}_f \cdot \mathbf{s}_p + x_f \qquad v_{fp} = \mathbf{n}_f \cdot \mathbf{s}_p + y_f \tag{56}$$

where

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \tag{57}$$

$$x_f = -\frac{\mathbf{t}_f \cdot \mathbf{i}_f}{z_f} \qquad y_f = -\frac{\mathbf{t}_f \cdot \mathbf{j}_f}{z_f} \tag{58}$$

$$\mathbf{m}_f = \frac{\mathbf{i}_f}{z_f} \qquad \mathbf{n}_f = \frac{\mathbf{j}_f}{z_f} \tag{59}$$

## I.2. Decomposition

Because equation (56) 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 using equation (25), and use singular value decomposition to factor the registered measurement matrix $W^*$ into the product of $\hat{M}$ and $\hat{S}$.

## I.3. Normalization

Again, the decomposition is not unique and we must determine the $3 \times 3$ matrix $A$ which produces the actual motion matrix $M = \hat{M}A$ and the shape matrix $S = A^{-1}\hat{S}$. From equation (59),

$$|\mathbf{m}_f|^2 = \frac{1}{z_f^2} \qquad |\mathbf{n}_f|^2 = \frac{1}{z_f^2} \tag{60}$$

We do not know the value of the depth $z_f$, 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 as we did in the paraperspective case, to impose the constraint

$$|\mathbf{m}_f|^2 = |\mathbf{n}_f|^2. \tag{61}$$

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

$$\mathbf{m}_f \cdot \mathbf{n}_f = 0. \tag{62}$$

As in the paraperspective case, equations (61) and (62) are homogeneous constraints, which

could be trivially satisfied by the solution $M = 0$, so to avoid this solution we add the constraint that

$$|\mathbf{m}_1| = 1. \tag{63}$$

Equations (61), (62), and (63) are the scaled orthographic version of the *metric constraints*. We can compute the $3 \times 3$ matrix $A$ which best satisfies them very easily, because the constraints are linear in the 6 unique elements of the symmetric $3 \times 3$ matrix $Q = A^T A$.

## I.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

$$\hat{\mathbf{i}}_f = \frac{\mathbf{m}_f}{|\mathbf{m}_f|} \qquad \hat{\mathbf{j}}_f = \frac{\mathbf{n}_f}{|\mathbf{n}_f|}. \tag{64}$$

Unlike the orthographic case, we can now compute $z_f$, the component of translation along the camera's optical axis, from equation (60).

# Appendix II. Perspective Method

This section presents an iterative method used to recover the shape and motion using a perspective projection model. Although our algorithm was developed independently and handles the full three dimensional case, this method is quite similar to a two dimensional algorithm developed by Taylor, Kriegman, and Anandan as reported in [9].

## II.1. Perspective Projection

In the perspective projection model, sometimes 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 14.

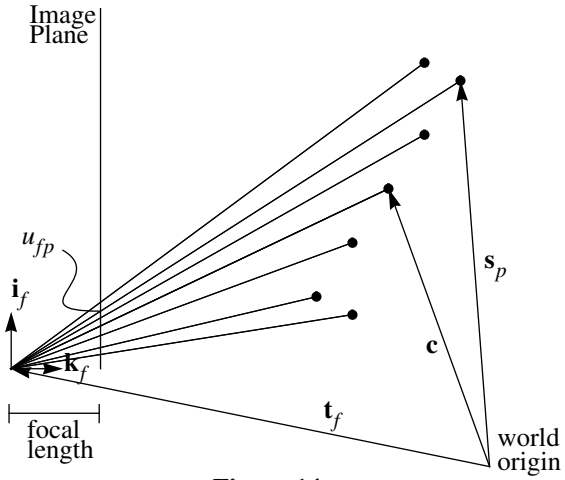



**Figure 14**
**Perspective Projection in two dimensions**

Simple geometry using similar triangles produces the perspective projection equations

$$u_{fp} = l\frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{\mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}$$
$$v_{fp} = l\frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{\mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)} \tag{65}$$

We assume unit focal length, and rewrite the equations in the form

$$u_{fp} = \frac{\mathbf{i}_f \cdot \mathbf{s}_p + x_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f}$$
$$v_{fp} = \frac{\mathbf{j}_f \cdot \mathbf{s}_p + y_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f} \tag{66}$$

where

$$x_f = -\mathbf{i}_f \cdot \mathbf{t}_f \qquad y_f = -\mathbf{j}_f \cdot \mathbf{t}_f \qquad z_f = -\mathbf{k}_f \cdot \mathbf{t}_f \tag{67}$$

## II.2. Iterative Minimization Method

The above equations are non-linear in the shape and motion variables. There is no apparent way to combine the equations for all points and frames into a single matrix equation, to separate the shape from the motion, or to compute the translational components directly from the image measurements, as we did in the orthographic and paraperspective cases. Instead we formulate the problem as an overconstrained non-linear least squares problem in the motion and shape variables, in which we seek to minimize the error

$$\varepsilon = \sum_{f=1}^{F} \sum_{p=1}^{P} \{ \left( u_{fp} - \frac{\mathbf{i}_f \cdot \mathbf{s}_p + x_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f} \right)^2 + \left( v_{fp} - \frac{\mathbf{j}_f \cdot \mathbf{s}_p + y_f}{\mathbf{k}_f \cdot \mathbf{s}_p + z_f} \right)^2 \} . \tag{68}$$

In the above formulation, there appear to be 12 motion variables for each frame, since each image frame is defined by three orientation vectors and a translation vector. In reality, since $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ must be orthonormal vectors, they can be written as functions of only three independent rotational parameters $\alpha_f$, $\beta_f$, and $\gamma_f$.

$$\begin{bmatrix} \mathbf{i}_f & \mathbf{j}_f & \mathbf{k}_f \end{bmatrix} = \begin{bmatrix} \cos\alpha_f\cos\beta_f & (\cos\alpha_f\sin\beta_f\sin\gamma_f - \sin\alpha_f\cos\gamma_f) & (\cos\alpha_f\sin\beta_f\cos\gamma_f + \sin\alpha_f\sin\gamma_f) \\ \sin\alpha_f\cos\beta_f & (\sin\alpha_f\sin\beta_f\sin\gamma_f + \cos\alpha_f\cos\gamma_f) & (\sin\alpha_f\sin\beta_f\cos\gamma_f - \cos\alpha_f\sin\gamma_f) \\ -\sin\beta_f & \cos\beta_f\sin\gamma_f & \cos\beta_f\cos\gamma_f \end{bmatrix} \tag{69}$$

Therefore we have six motion parameters, $x_f$, $y_f$, $z_f$, $\alpha_f$, $\beta_f$, and $\gamma_f$, for each frame, and three shape parameters, $\mathbf{s}_p = \begin{bmatrix} s_{p1} & s_{p2} & s_{p3} \end{bmatrix}$ for each point. Equation (66) defines an overconstrained set of $2FP$ equations in these $6F + 3P$ variables, and we carry out the minimization of equation (68) with $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ defined by equation (69) as functions of $\alpha_f$, $\beta_f$, and $\gamma_f$.

We could in theory apply any one of a number of non-linear equation solution techniques to this problem. Such methods begin with a set of initial variable values, and iteratively refine those values to reduce the error. We know the mathematical form of the equations, so we can use derivative information to guide our numerical search. However, general non-linear least square techniques would not take full advantage of the structure of our equations. The Levenberg-Marquardt technique [7] would require the creation and inversion of a $(6F + 3P) \times (6F + 3P)$ matrix at each step of the iteration. This is unacceptably slow, since we often use hundreds of points and frames.

Our method takes advantage of the particular structure of the equations by separately refining the shape and motion parameters. We hold the shape constant and solve for the motion parameters which minimize the error. We then hold the motion constant, and solve for the shape parameters which minimize the error. We repeat this process until an iteration produces no significant reduction in the total error $\varepsilon$.

While holding the shape constant, the minimization with respect to the motion variables can be performed independently for each frame. This minimization requires solving an overcon-

strained system of 6 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 3 variables. This not only reduces the problem to manageable complexity, but as pointed out in [9], it lends itself well to parallel implementation.

We perform the individual minimizations, fitting 6 motion variables to $P$ equations or fitting 3 shape variables to $2F$ equations, using the Levenberg-Marquardt method [7], a method which uses steepest descent when far from the minimum and varies continuously towards the inverse-Hessian method as the minimum is approached. Each step of the iteration requires $P$ inversions of $6 \times 6$ matrices and $2F$ inversions of $3 \times 3$ matrices.

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. However, experimentally we found that the algorithm converges significantly faster when the shape and motion parameters are all allowed to vary. Once the algorithm has converged to a solution, we adjust the final shape and translation to place the origin at the object's center of mass, scale the solution so that the depth in the first frame is 1.0, and rotate the solution 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 $\alpha_1 = \beta_1 = \Gamma_1 = 0$.

One problem with any iterative method is that the final result can be highly dependent on the initial values. Taylor, Kriegman, and Anandan [9] 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 perspective iteration method.