4. Projective Factorization

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

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

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

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

Finally we show that the non-linear minimization framework developed for projective factorization and Euclidean optimization can be extended to incorporate image data directly rather than relying solely on tracked features. The shape from motion problem is reformulated in terms of minimizing differences in image intensity rather than minimizing errors between predicted and tracked feature point positions. While the method could in theory

obviate the need for feature tracking altogether, in practice it is primarily useful for refining results in which some of the initially tracked features were poorly tracked, thus allowing a more dense shape recovery.

4.1. Projective Factorization

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

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

4.1.1. Least Squares Formulation

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

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

This can be rewritten in the form

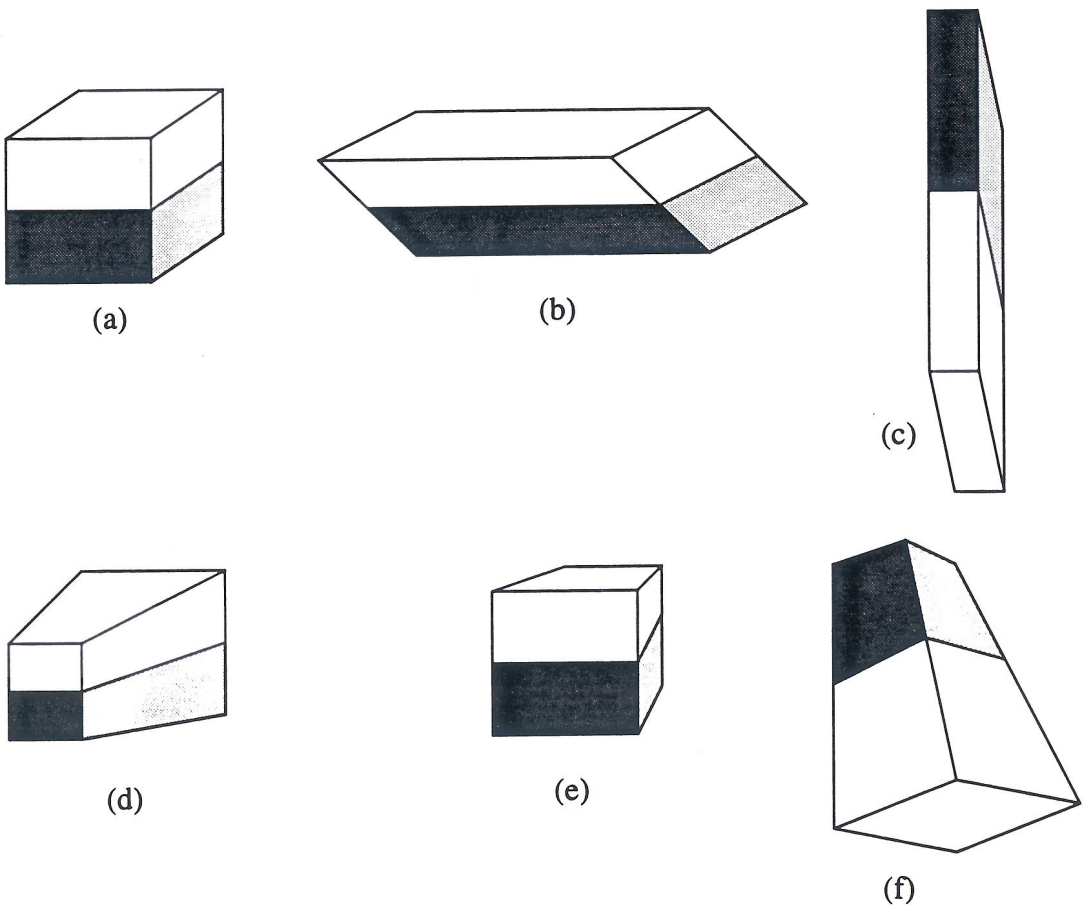


Figure 24. Affine Equivalence and Projective Equivalence
The objects (a), (b), and (c) are all affinely equivalent, since each is the result of arbitrarily scaling the axes or changing the angles between the axes of another. (a), (b), (c), (d), (e), and (f) are projectively equivalent, since each is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

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

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

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

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

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

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

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

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

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

4.1.2. Levenberg-Marquardt Solution Method

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

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

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

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

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

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

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

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

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

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

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

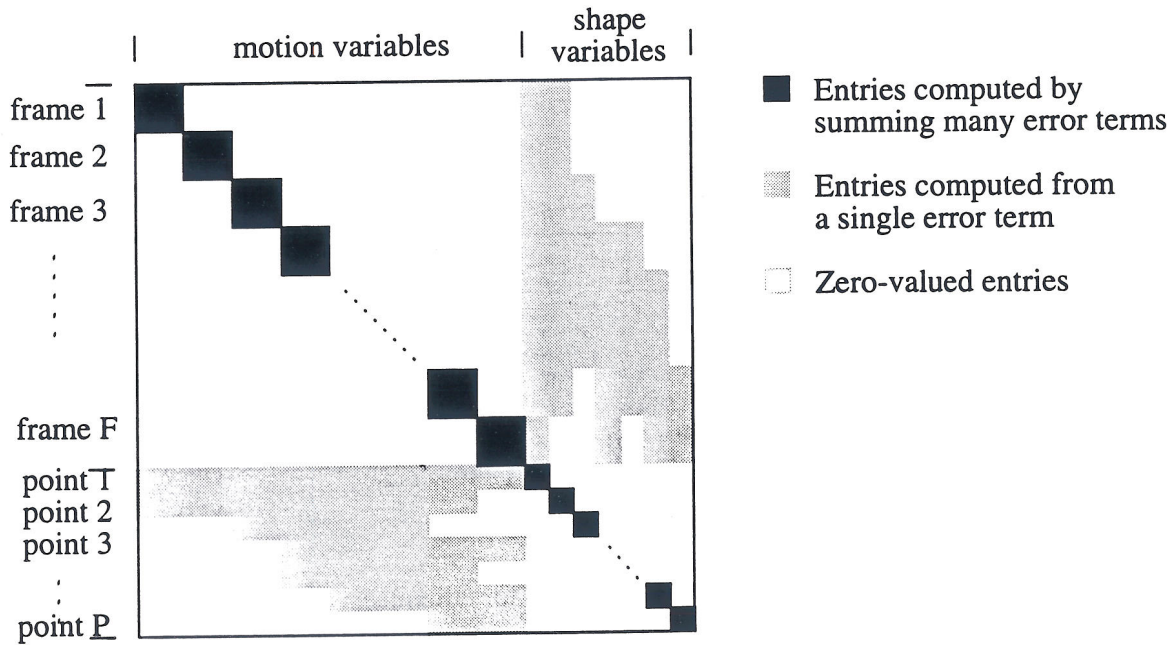


Figure 25. Hessian matrix for projective shape from motion recovery
The diagonal blocks are 11x11 for the motion variables and 3x3 for the shape variables. The actual shape of the shaded area is determined by the fill pattern of feature observability for the particular sequence.

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

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

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

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

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

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

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

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

4.1.3. Depth-shape Formulation

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

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

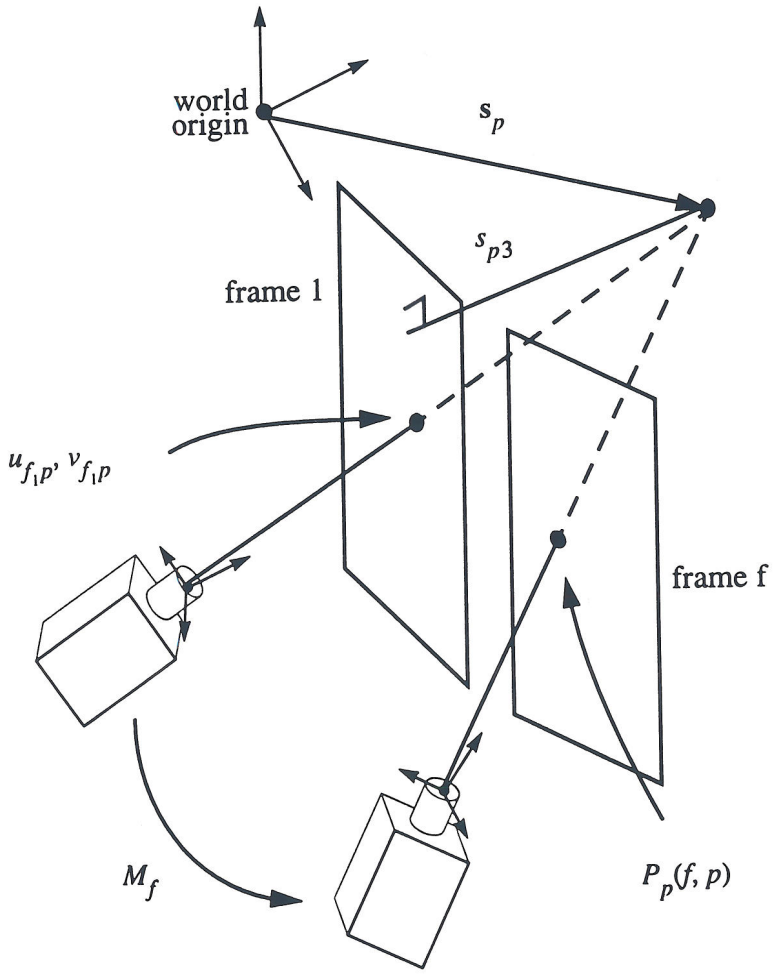


Figure 26. Shape-depth formulation

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

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

$$\begin{bmatrix} u_{f_1 p} \\ v_{f_1 p} \end{bmatrix} = \begin{bmatrix} \dfrac{m_{f_1 11} s_{p1} + m_{f_1 12} s_{p2} + m_{f_1 13} s_{p3} + m_{f_1 14}}{m_{f_1 31} s_{p1} + m_{f_1 32} s_{p2} + m_{f_1 33} s_{p3} + m_{f_1 34}} \\ \dfrac{m_{f_1 21} s_{p1} + m_{f_1 22} s_{p2} + m_{f_1 23} s_{p3} + m_{f_1 24}}{m_{f_1 31} s_{p1} + m_{f_1 32} s_{p2} + m_{f_1 33} s_{p3} + m_{f_1 34}} \end{bmatrix} \tag{108}$$

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

$$\begin{bmatrix} \left(m_{f_1 31} u_{1p} - m_{f_1 11}\right) & \left(m_{f_1 32} u_{1p} - m_{f_1 12}\right) \\ \left(m_{f_1 31} v_{1p} - m_{f_1 21}\right) & \left(m_{f_1 32} v_{1p} - m_{f_1 22}\right) \end{bmatrix} \begin{bmatrix} s_{p1} \\ s_{p2} \end{bmatrix} = -\begin{bmatrix} \left(m_{f_1 33} u_{1p} - m_{f_1 13}\right) s_{p3} + \left(m_{f_1 34} u_{1p} - m_{f_1 14}\right) \\ \left(m_{f_1 33} v_{1p} - m_{f_1 23}\right) s_{p3} + \left(m_{f_1 34} v_{1p} - m_{f_1 24}\right) \end{bmatrix} \tag{109}$$

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

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

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

This single variable "depth-shape" formulation can only be used when all of the feature points are visible in the first image of the sequence. Since s_{p1} and s_{p2} are in fact functions not only of s_{p3}, but also of the motion parameters in the first frame in which the point was seen, if that frame's motion parameters were variables and not constants, then we would need to update the Hessian matrix to account for this. Error terms for the observation for point p and f would not only depend on the shape variables for point p and the motion variables for frame f, but would also depend on the motion variables for some frame f', the frame in which point p was first seen. Although this could be easily accommodated within the Levenberg-Marquardt minimization framework, it would alter the pattern of the Hessian matrix. The upper-left block of the matrix would no longer be block diagonal, but would contain smaller blocks scattered throughout the currently unfilled areas. This could destroy its block-diagonal dominance and significantly increase the number of iterations of the gra-

dient descent method required to solve the large system. However, this increase might be offset by the reduction of the number of variables from $6F+3P$ to $6F+P$. We have not implemented a system capable of applying the depth-shape formulation to sequences in which some of the points are not seen in the first image, so currently such sequences must be solved using three variables per point.

4.1.4. Projective Normalization

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

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

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

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

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

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

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

using equality.

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

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

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

where

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

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

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

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

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

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

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

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

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

to avoid the solution $Q = 0$.

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

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

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

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

4.1.5. Projective Normalization for Distant Objects

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The problem is actually not so different from the projective factorization step. We still use Levenberg-Marquardt optimization to find the shape and motion which minimize the error sum defined in equation (100), and still use the perspective projection equations defined in (97). However, we now minimize the errors in the perspective projection equations using the Euclidean shape, motion, and camera parameters directly as the variables to be refined, rather than using the projective motion variables m_{fij}. We write the rotation in terms of the

three rotation angles θ_f, φ_f, and ω_f introduced in equations (74), so that each frame's $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ vectors are guaranteed to be orthonormal.

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

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

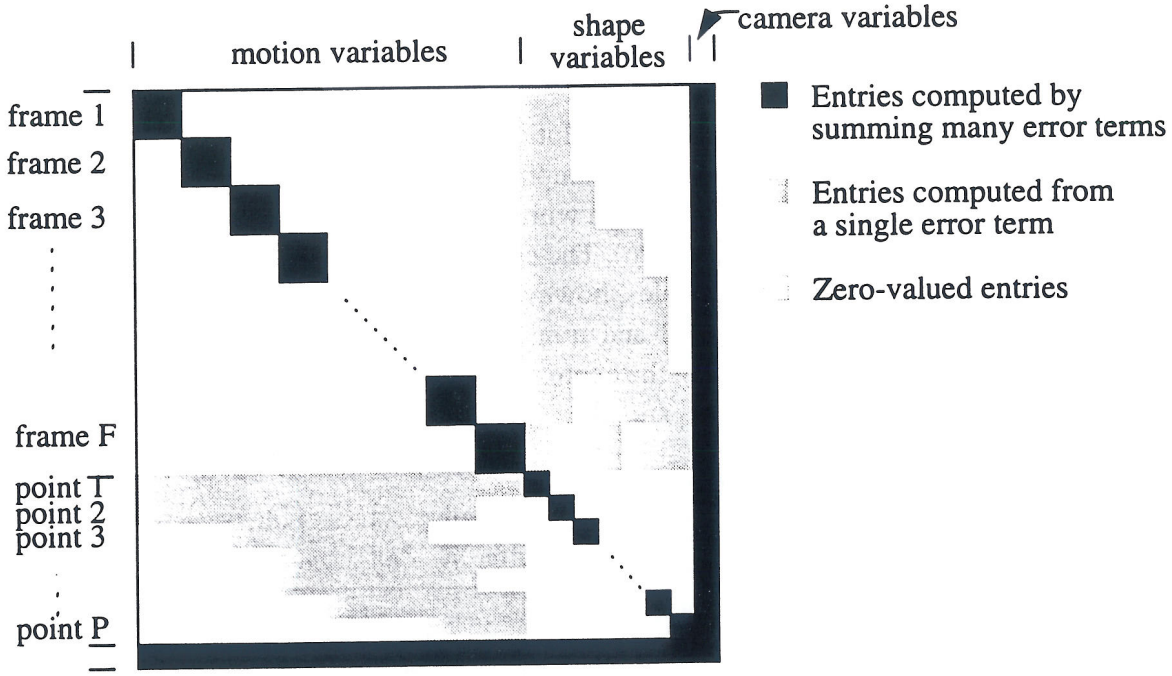


Figure 27. Hessian matrix for Euclidean shape from motion recovery. The diagonal blocks are 6x6 for the motion variables and 3x3 for the shape variables, while there are 4 camera variables.

Our experience shows that the focal length l trades off very easily with the depth to the object z_f. which makes their precise recovery difficult and slows the rate of the system's convergence. Since the scaling factor of M_f is arbitrary, we divide it by l and introduce a

perspective distortion factor $\mu = 1/l$ and a relative depth factor $d_f = z_f/l$. Solving for these variables is more numerically stable than solving for l and z_f. In cases in which the object is far from the camera and foreshortening effects are overcome by noise, the best solution may be one with infinite focal length, i.e. a scaled orthographic solution. Using the μ and d_f formulation allows a graceful change to scaled orthography by letting μ approach zero, while a method explicitly representing the focal length would need the focal length to approach infinity in order to model scaled orthography. The motion matrix is defined in terms of these parameters as

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

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

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

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

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

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

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

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

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

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

4.3. Shape and Motion Recovery under Radial Projection

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

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

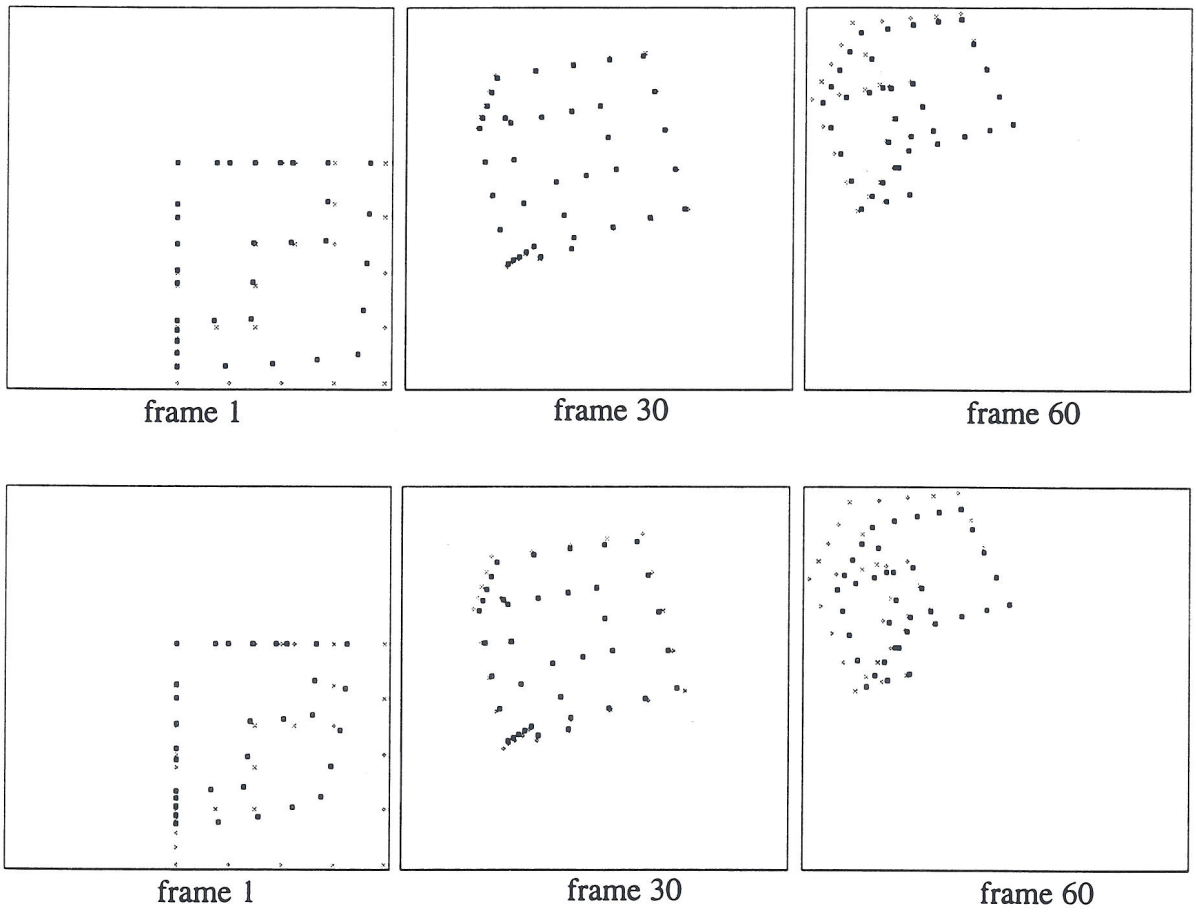


Figure 28. Illustration of Radial Distortion

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

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

4.3.1. Standard "2D" Radial Distortion Model

Many camera calibration techniques are based on a radial camera model [41][44][46]. This camera model relates the perspectively projected but non-radially-distorted position of a

point (X_u, Y_u) to the distorted position (X_d, Y_d) by the equations [41]

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

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

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

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

$$e = \begin{cases} \left(\frac{d}{2}\right)^{1/3}\left(1 + \sqrt{1 + \frac{4d}{27}}\right)^{1/3} - \frac{\sqrt[3]{2}}{3}d^{2/3}\left(1 + \sqrt{1 + \frac{4d}{27}}\right)^{-1/3} & d \text{ defined} \\ 1 & d \text{ undefined} \end{cases} \tag{130}$$

$$\text{where} \quad d = \frac{1}{\kappa_1\left(X_u^2 + Y_u^2\right)}$$

$$Y_d = Y_u e$$

$$X_d = X_u e$$

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

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

Although the model has proven quite adequate for photogrammetry and camera calibration problems, we can see that the above formulation suffers from several significant drawbacks when used in 3D vision scenarios where the forward projection must be computed. The

expressions are ill-formed when either the radial distortion parameter κ_1 or the distance from the image center $X_u^2 + Y_u^2$ are zero. In these cases e approaches unity, but in practical terms it must be computed as the difference of two large numbers, making it sensitive to truncation and roundoff errors. We expect that in many cases the radial distortion parameter will be very small, since many image sequences contain little or no radial distortion, and some feature points are likely to be located near the center of the image, so this will cause significant numerical problems. Additionally, computation of the distorted position is very complicated, and the expressions for the derivative of the projection function with respect to the motion and shape variables, required for Levenberg-Marquardt minimization, would be hideous.

4.3.2. "3D" Radial Distortion Model

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

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

The distorted image position of a point is

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

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

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

We can now solve for shape and motion under a radial distortion model using the same formulation used in the previous sections. The derivative of the projection function with

respect to any shape and motion variable υ is computed by applying the chain rule to the derivative of the perspective projection function.

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

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

The derivatives with respect to the camera parameters are

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

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

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

Note that the value of κ used in equation (133) alone does not give any indication of the

overall magnitude of radial distortion likely to be seen in an image produced by the lens unless the camera's field of view is also known. For example, if a 2 unit wide object is 10 units from the camera and it completely fills the field of view of the lens, then a value of $\kappa = 1$ will cause a point at the center of the left edge of the image to move ~1% closer to the image center. The value of P_{p_x} at the left edge of the image will be 0.1, so the distortion factor $1/\left(1 + \kappa\left(P_{p_x}^2 + P_{p_y}^2\right)\right)$ will be $1/1.01$ or about 1%. However, if a 2 unit wide object only half fills the field of view, then points at the left edge of the image will have $P_{p_x} = 0.2$, so the point's position will deform by $1/1.04$ or about 4%. The second lens will produce images with substantially more radial distortion, yet both of them have $\kappa = 1$. Therefore whenever we refer to a radial distortion parameter value we will give the value for the *normalized radial distortion parameter*, which is just κ times the square of the field of view in pixels. Using this definition, the amount of radial distortion visible in an image can be characterized by a single number. So for example, for any lens with a normalized radial distortion parameter value $\kappa = 1$, the center point of the left edge will be distorted by $1/(1 + 1(0.5)^2) = 0.8$ or a reduction of 20%.

4.3.3. Depth-shape Formulation

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

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

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

$$Y_u = \left(\frac{Y_d - o_y}{a}\right) e \qquad (136)$$

$$X_u = (X_d - o_x)\, e$$

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

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

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

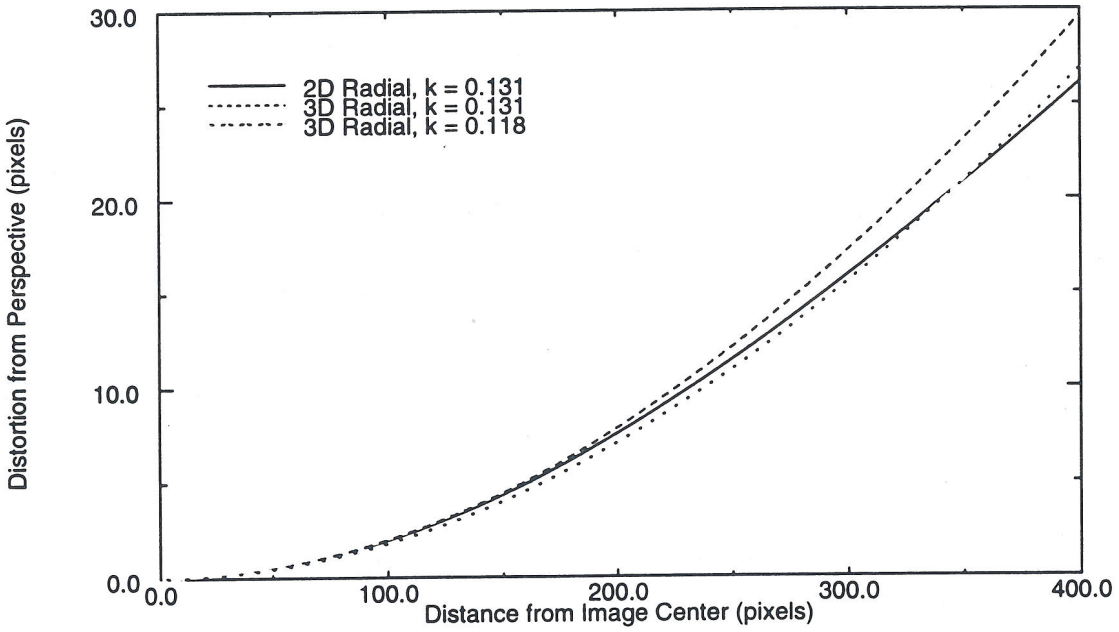


Figure 29. Comparison of distortion models.

4.4. Performance Analysis and Discussion

4.4.1. Analysis of Projective and Euclidean techniques

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

The initial motion used for all of these experiments consisted of assigning each frame the same camera orientation and position, i.e. assuming no camera motion at all. The initial

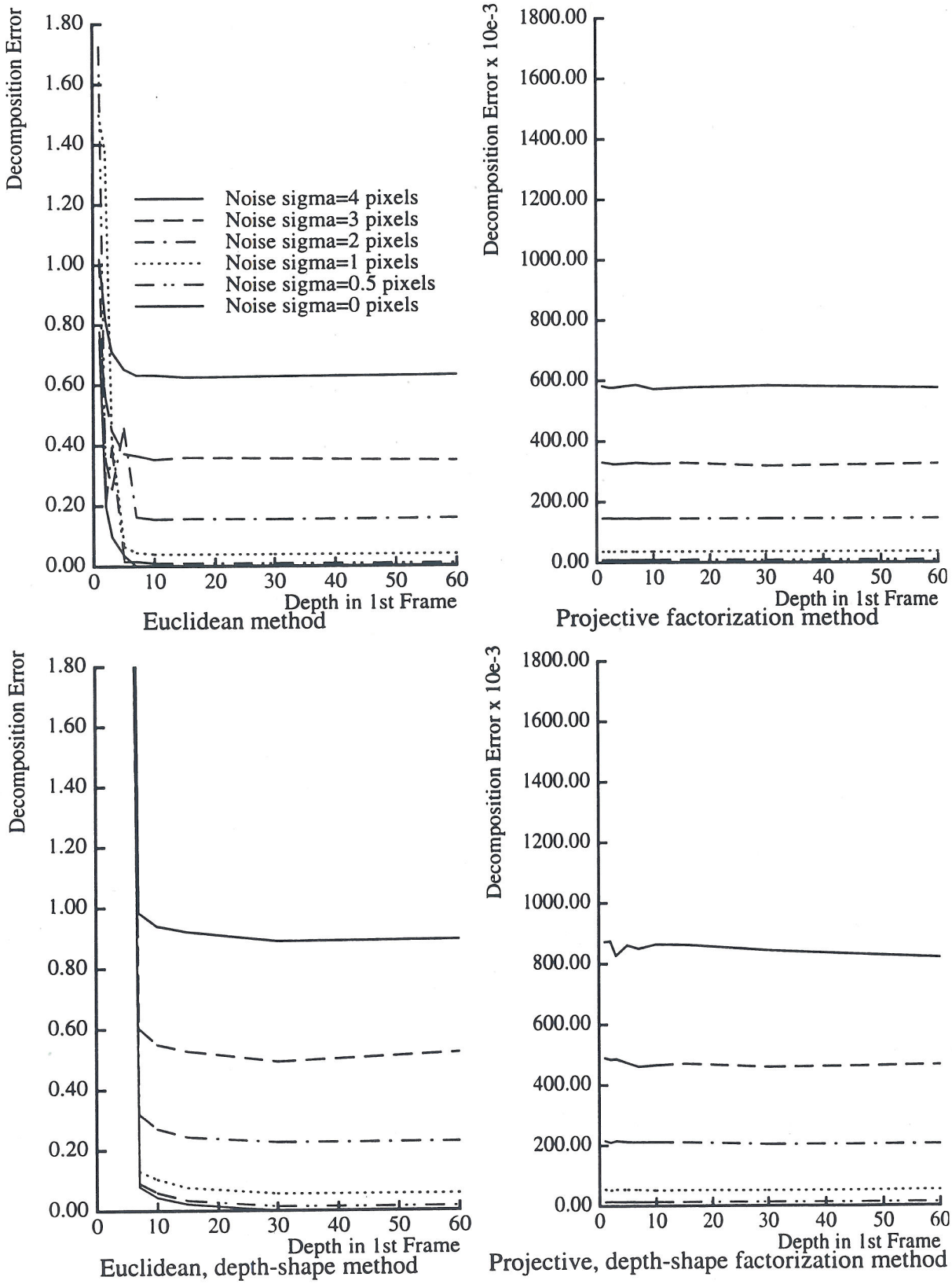


Figure 30. Total sum-of-squares error ε

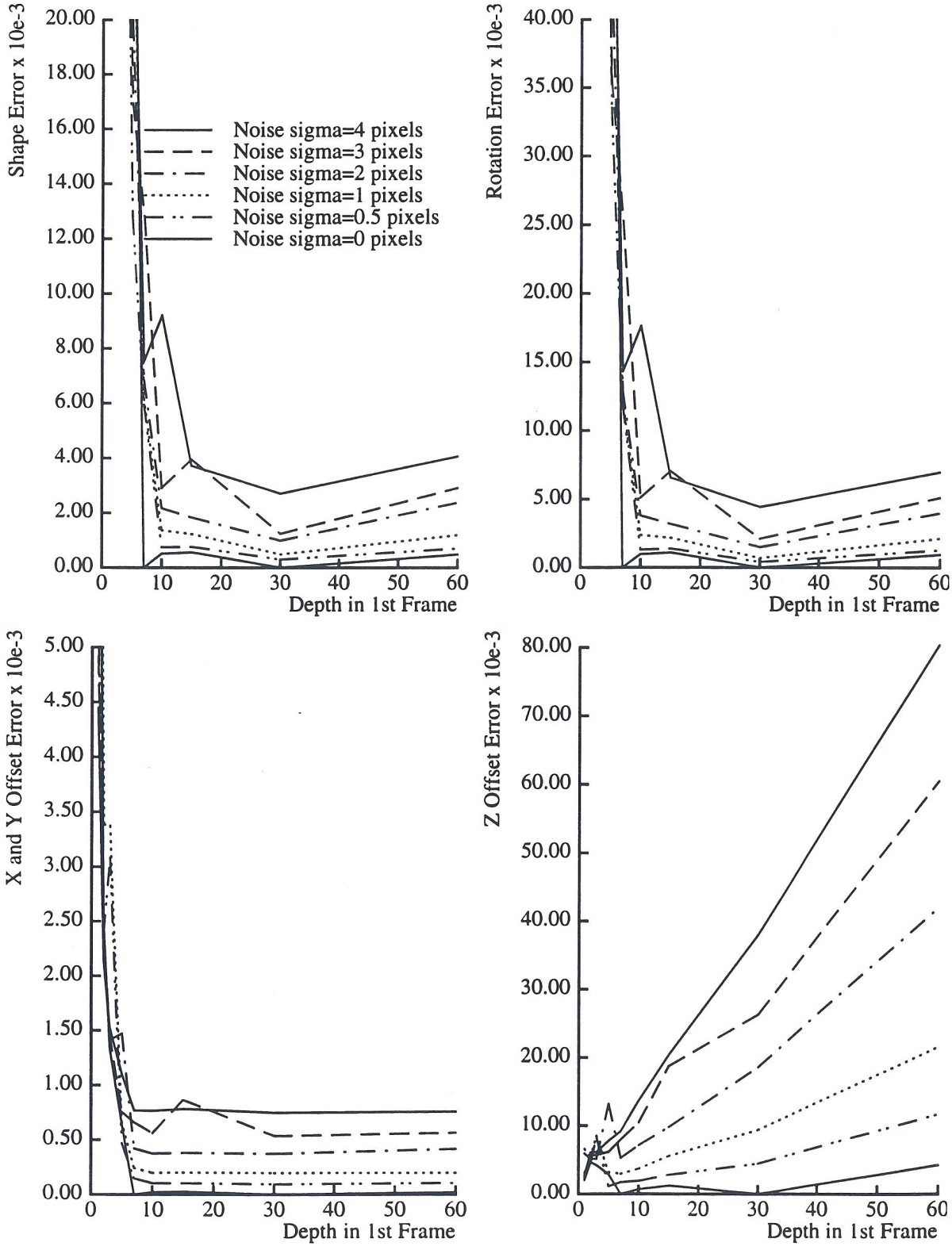


Figure 31. Shape and Motion Recovery Error for Euclidean Method

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

4.4.1.2. Analysis of Projective Normalization for Distant Objects

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

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

4.4.1.3. Analysis of Depth-shape Formulation

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

4.4.1.4. Analysis of Radial Distortion Method

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

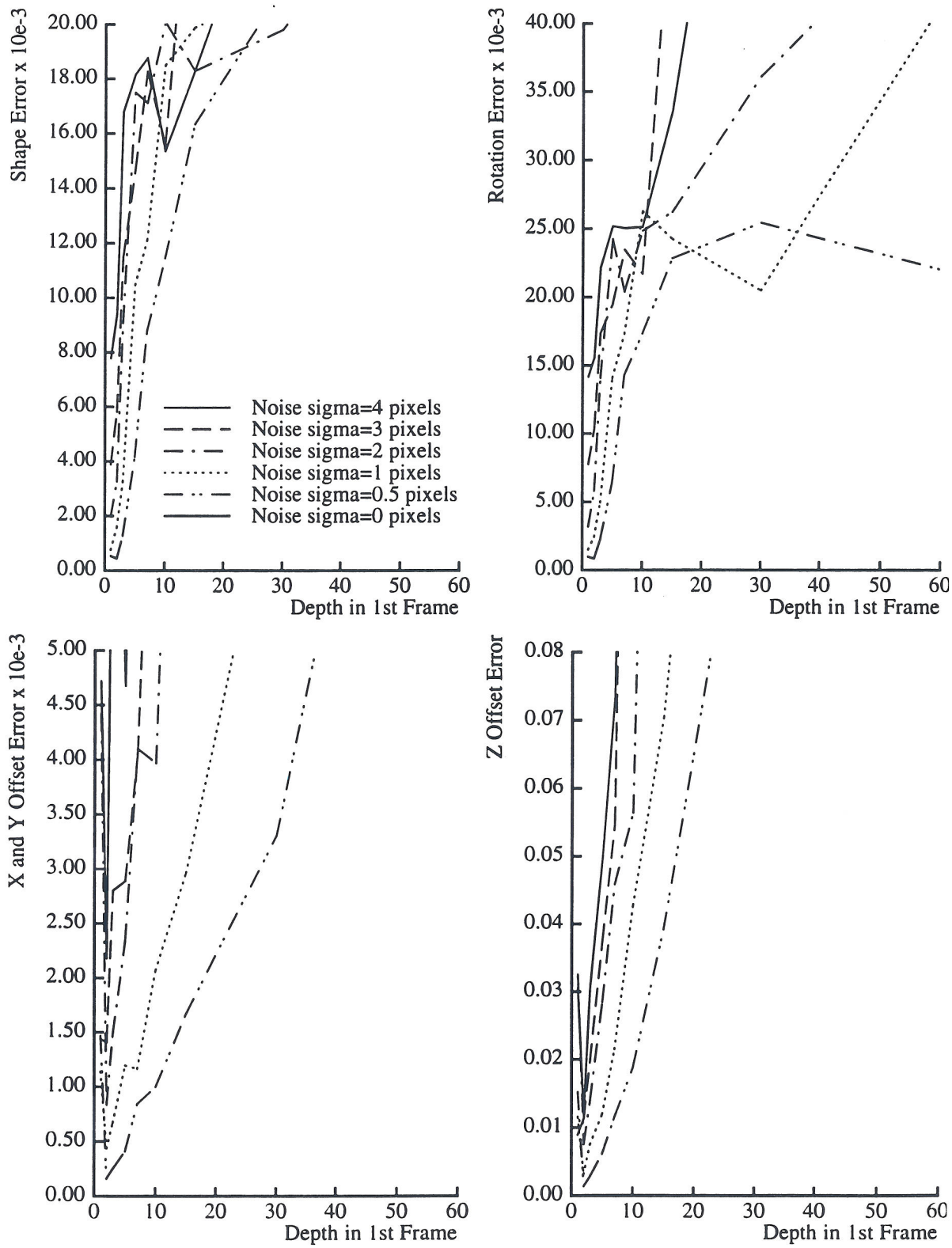


Figure 32. Projective Factorization Shape and Motion Recovery Error, $\sigma = 1$

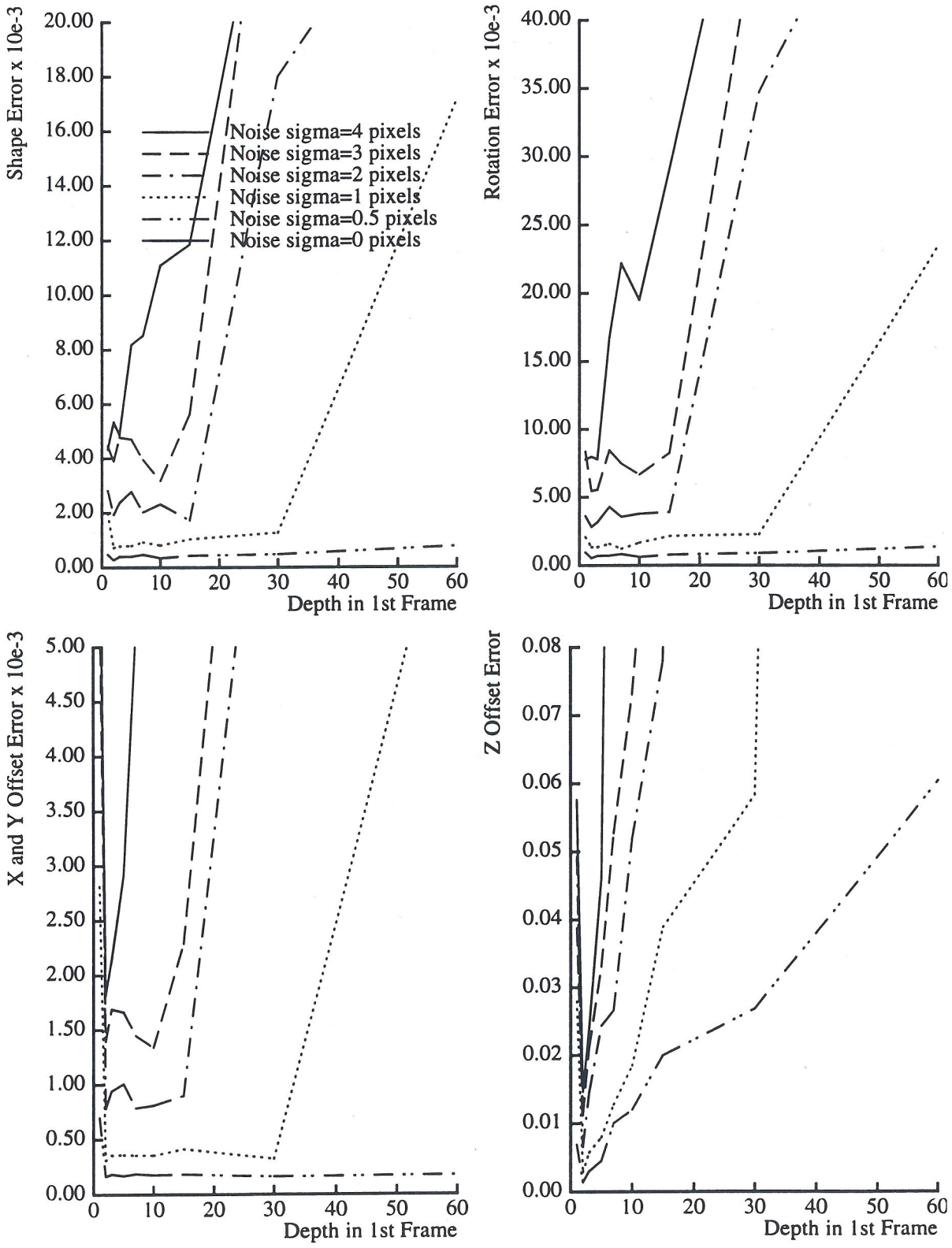


Figure 33. Projective Factorization Shape and Motion Recovery Error, Variable σ

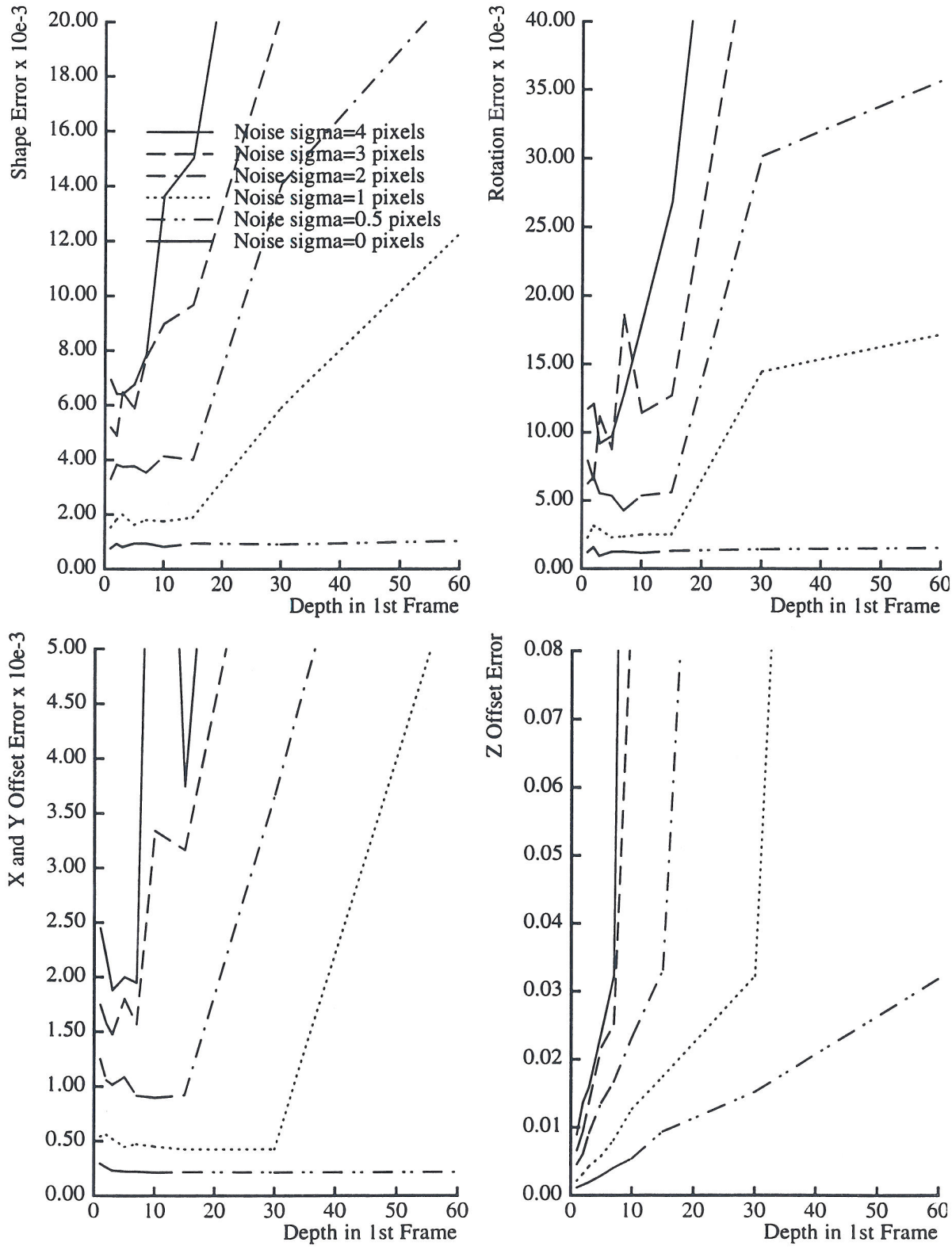


Figure 34. Projective Shape and Motion Recovery Error Variable σ, Depth-shape Formulation

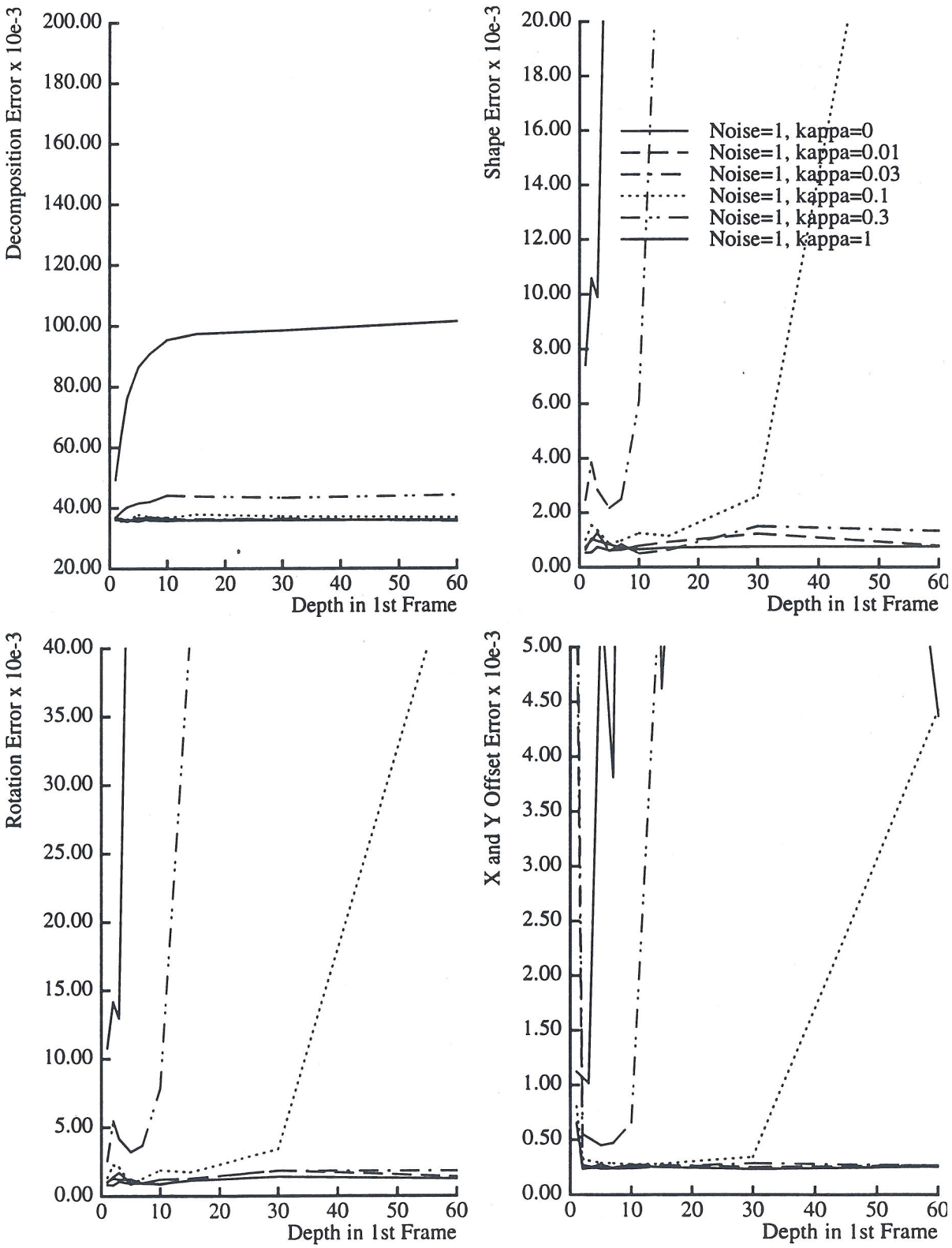


Figure 35. Non-radial Projective Factorization Method Applied to Radially Distorted Images

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

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

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

4.4.1.5. Analysis of Occlusion Handling

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

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

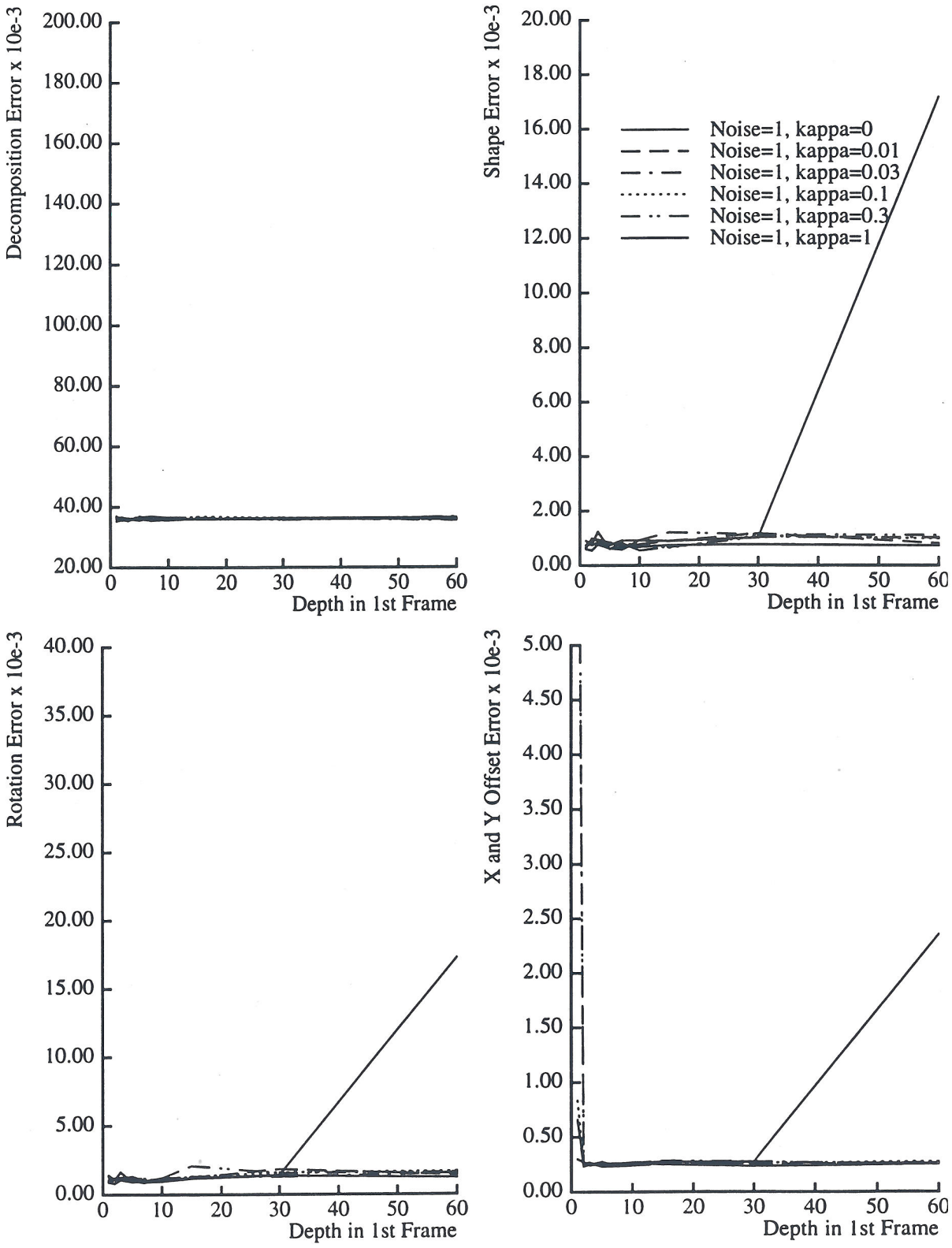


Figure 36. Projective Factorization with Variable Radial Distortion

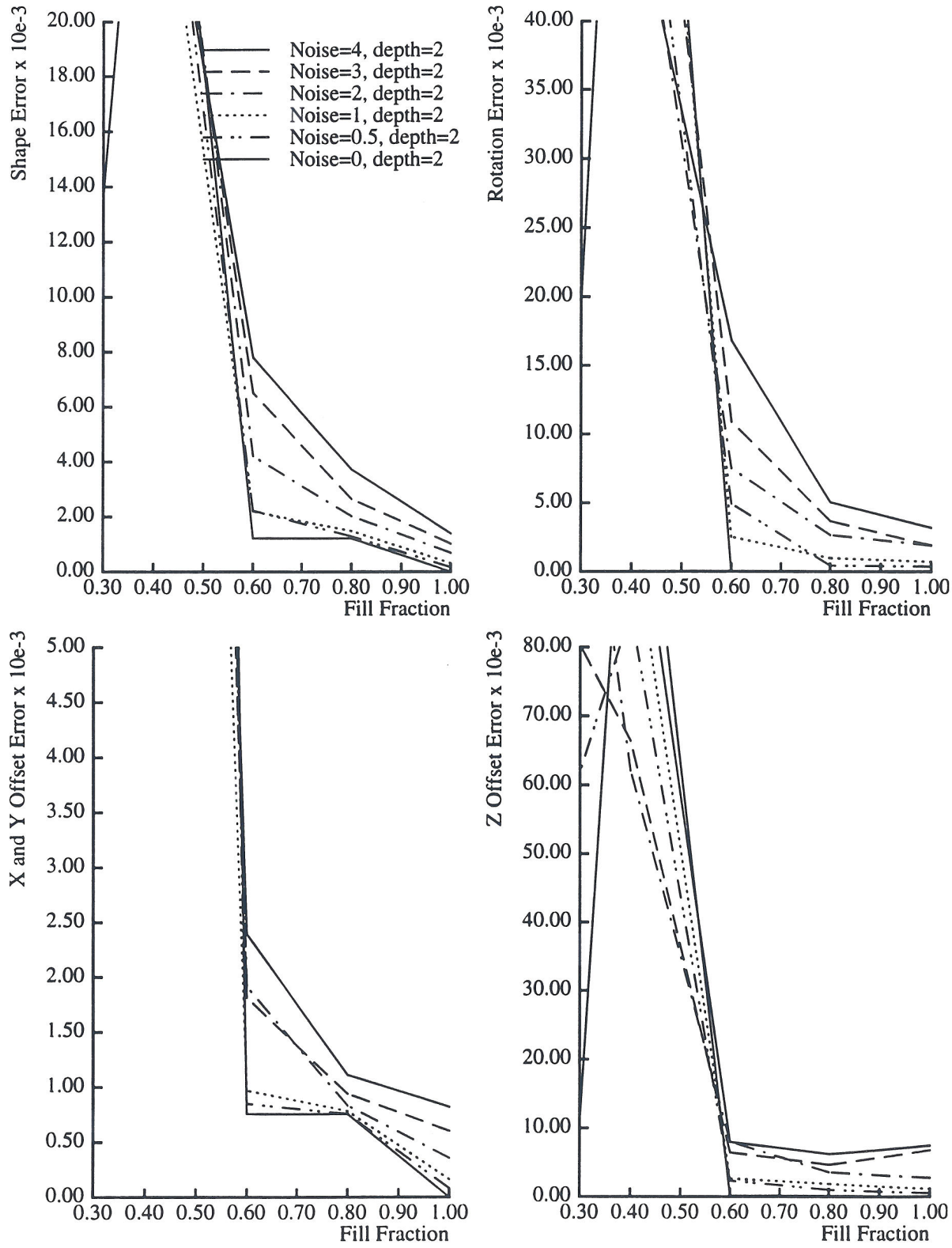


Figure 37. Projective Shape and Motion Recovery with Occlusion

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

4.4.1.6. Failure of Auto-Camera Calibration

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

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

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

4.4.2. Outdoor Building Example

A building at Carnegie Mellon University was imaged using a hand-held portable cam-

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

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

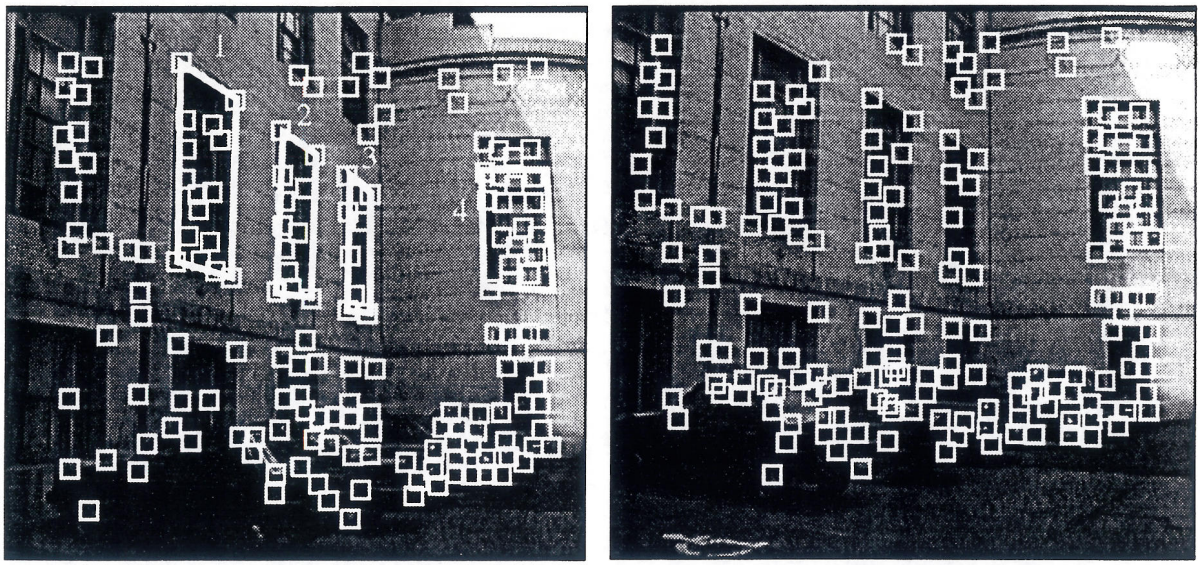
The projective factorization method was used to recover the shape and motion. The projective shape recovered from the factorization step is shown in Figure 39. This shape clearly contains both projective and affine distortions. For example, the object narrows towards the rear, and the wall is at approximately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

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

4.4.3. Indoor Kitchen Example

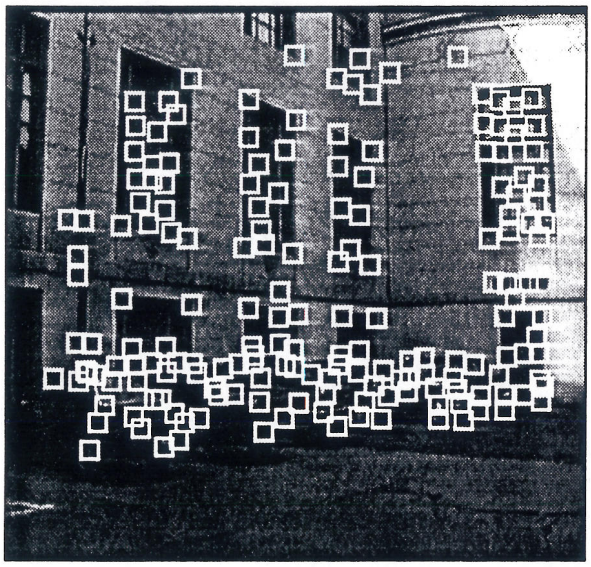
In this example, kitchen was imaged using the same hand-help portable camcorder. The camcorder had a variable zoom, which was set at the widest angle setting in order to view the entire kitchen from fairly close range. At this setting radial distortion effects, as can be seen by noticing the curved appearance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

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

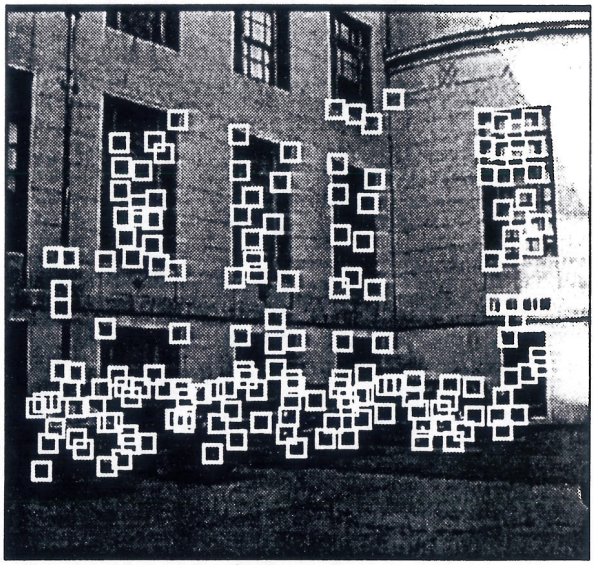


Frame 1

Frame 20



Frame 40



Frame58

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

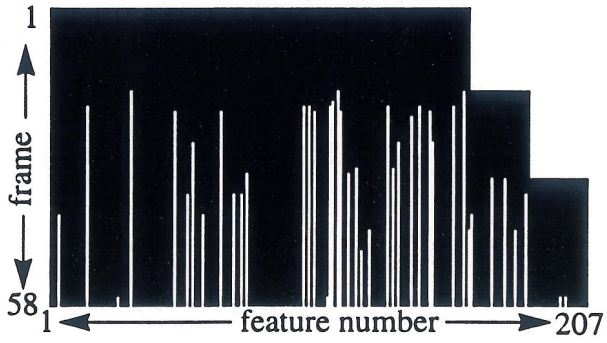
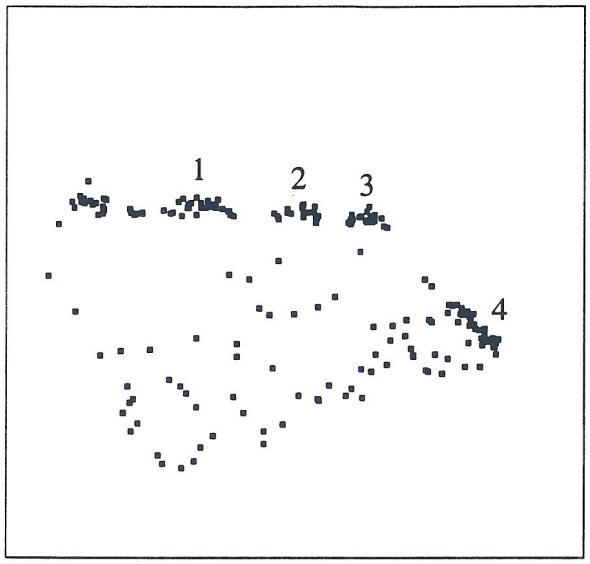
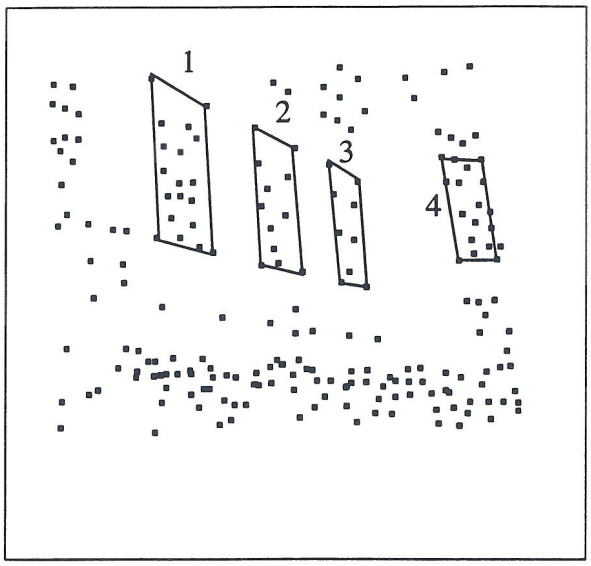


Figure 38. Hamburg Hall Sequence and Tracked Features

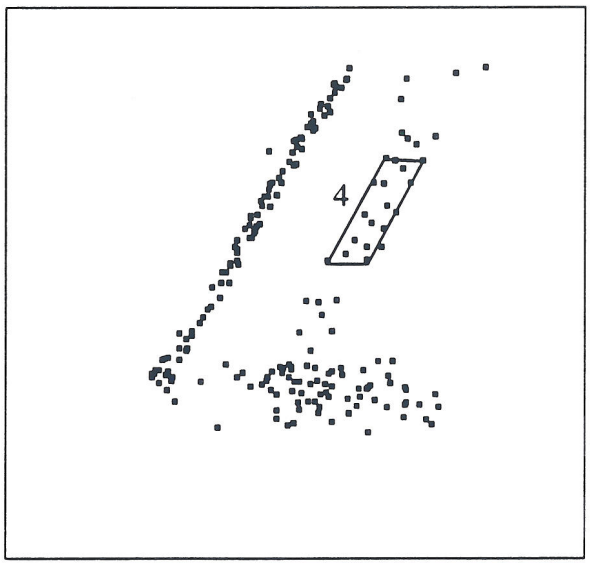
Four windows have been outlined and numbered manually for later reference.



Projective Shape - top view



Projective Shape - side view



Projective Shape - end view

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

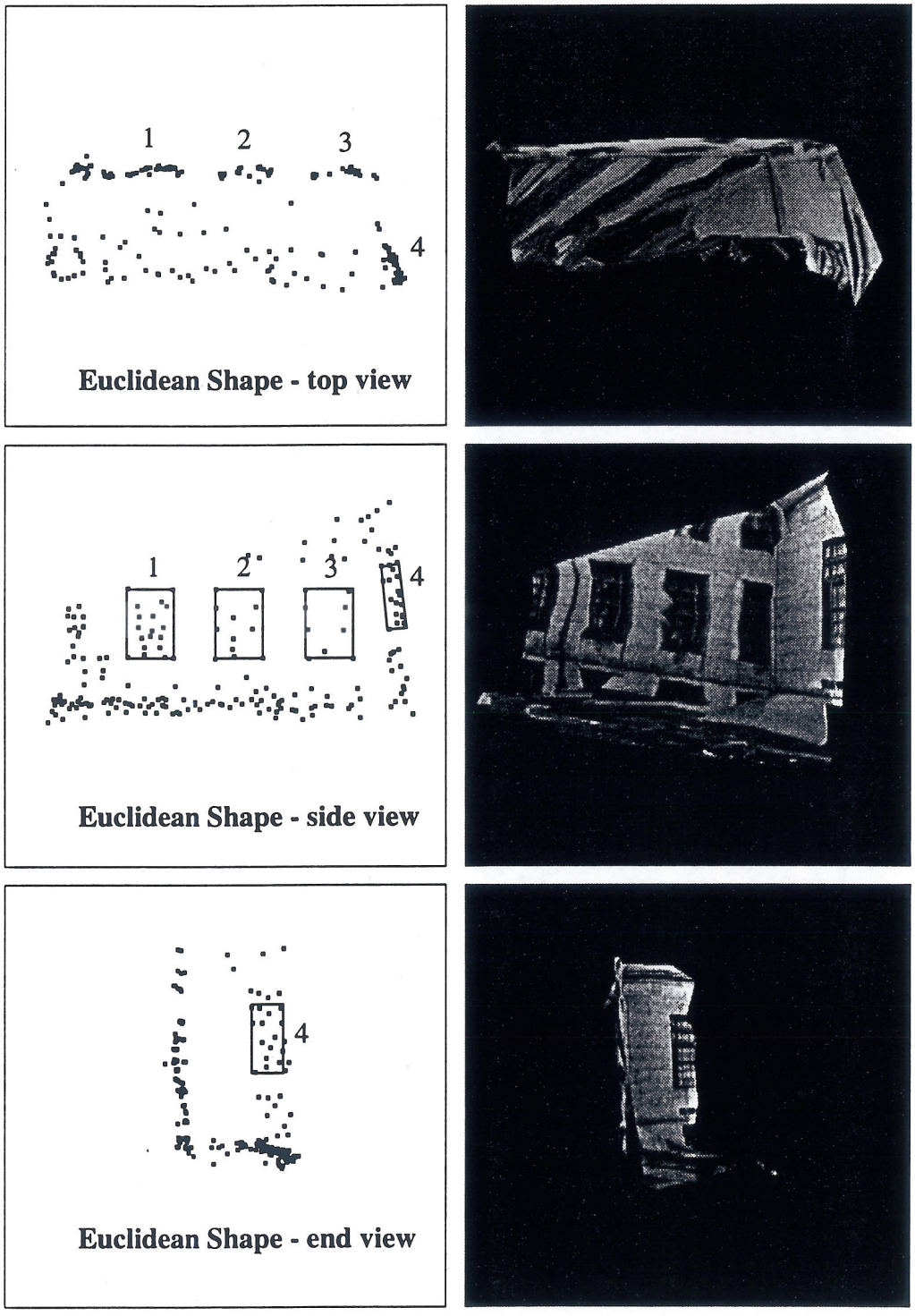
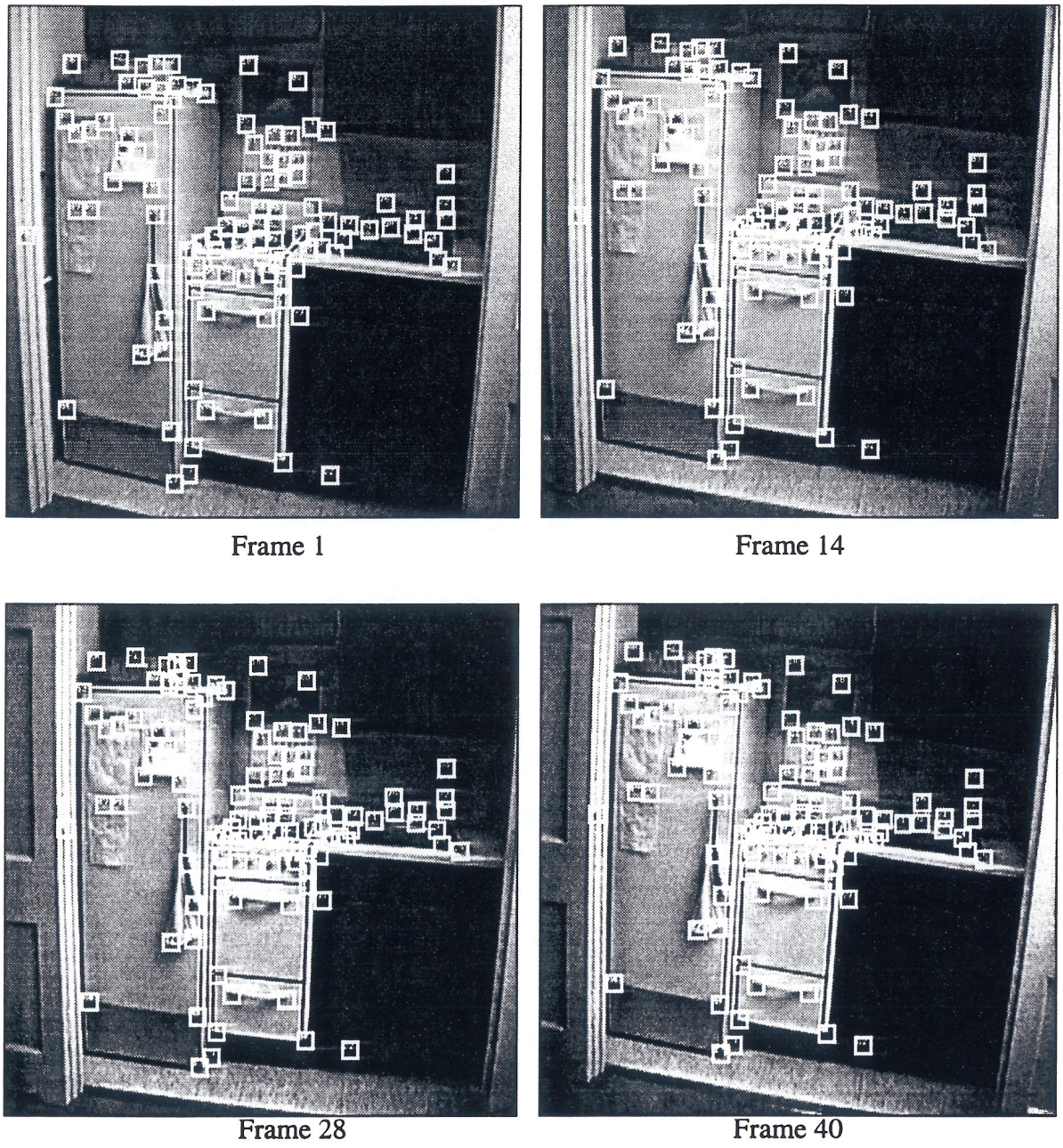


Figure 40. Euclidean Shape Recovered from Projective Factorization
The windows have been marked by hand using rectangles of equal size, to demonstrate that the recovered window shapes are equal-sized rectangles. The figures on the right show the results of mapping image texture onto the recovered shape,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



Frame 1 Frame 14

Frame 28 Frame 40

Figure 41. Kitchen Image Sequence and Feature Tracking Results

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

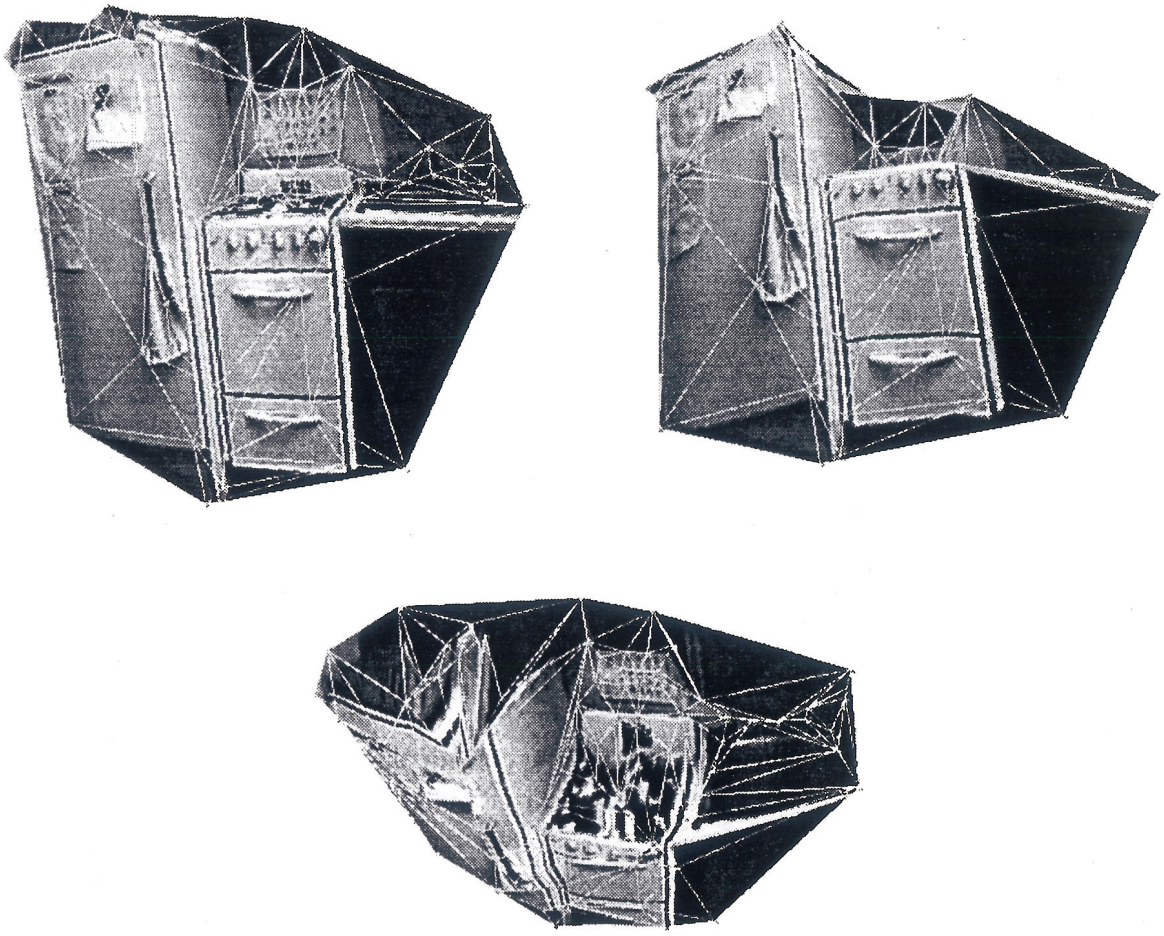


Figure 42. Reconstructed Kitchen Shape

4.5. Direct Shape and Motion Recovery without Tracking

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

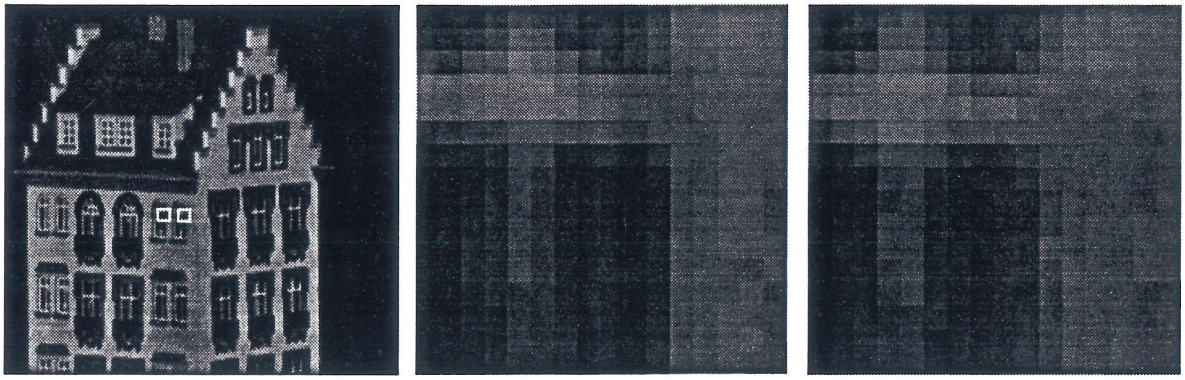


Figure 43. Similarity of Image Features

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

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

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

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

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

4.5.1. Least Squares Formulation

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

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

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

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

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

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

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

4.5.2. Levenberg Marquardt Solution

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

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

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

The α_{ij} and β_i entries still follow the pattern of equations (103) and (104) but are extended to sum the pixel errors over the region $R(F_p)$ as well. This involves summing $|R(F_p)|$ times as many terms as in the point-based method; for 15×15 windows, this is more than two orders of magnitude more computation. Indeed in the point-based methods, the majority of the computation was spent solving the linear system in α at each iteration, but using the

image-based method computing α becomes the primary computational burden. We can reduce the computational requirements considerably by precomputing five terms summations over each window. These values are independent of the particular variable υ and can be used to compute the entries of α and β for all variables for the given point or frame. The summations to compute α_{ij} and β_i are reduced to the same number of terms as for the point based method, and that additional computational load of over the point-based method is only the computation of these five terms for each feature point and each frame.

The β_i entries are computed using the following summation.

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

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

The α_{kj} entries are computed by summing

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

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

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

4.5.3. Multi-resolution Technique to Avoid Local Minima

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

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

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

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

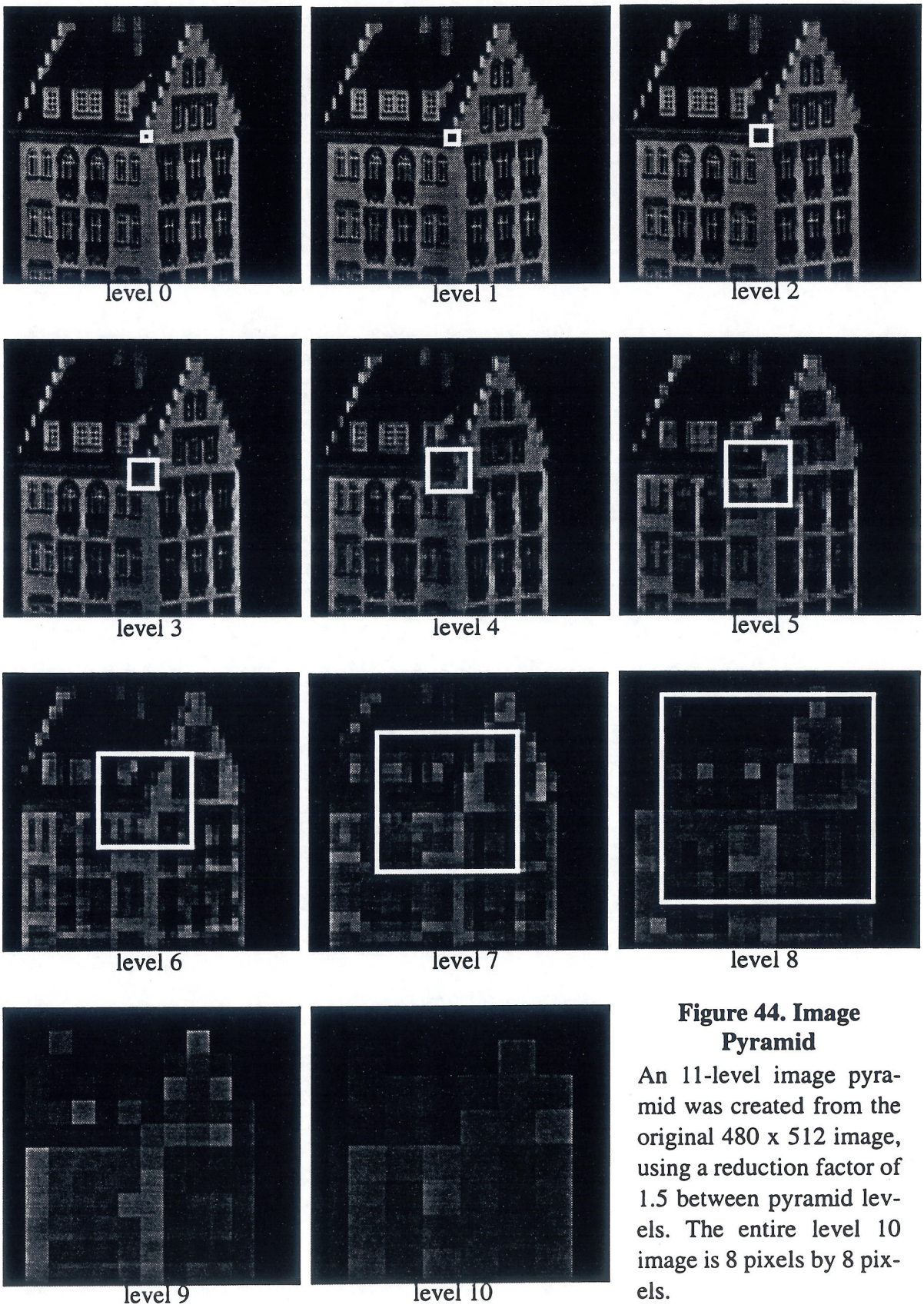


Figure 44. Image Pyramid

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

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

4.5.4. Extension to Affine Warping Model

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

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

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

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

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

4.5.5. Direct Shape and Motion Recovery Example

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

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

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

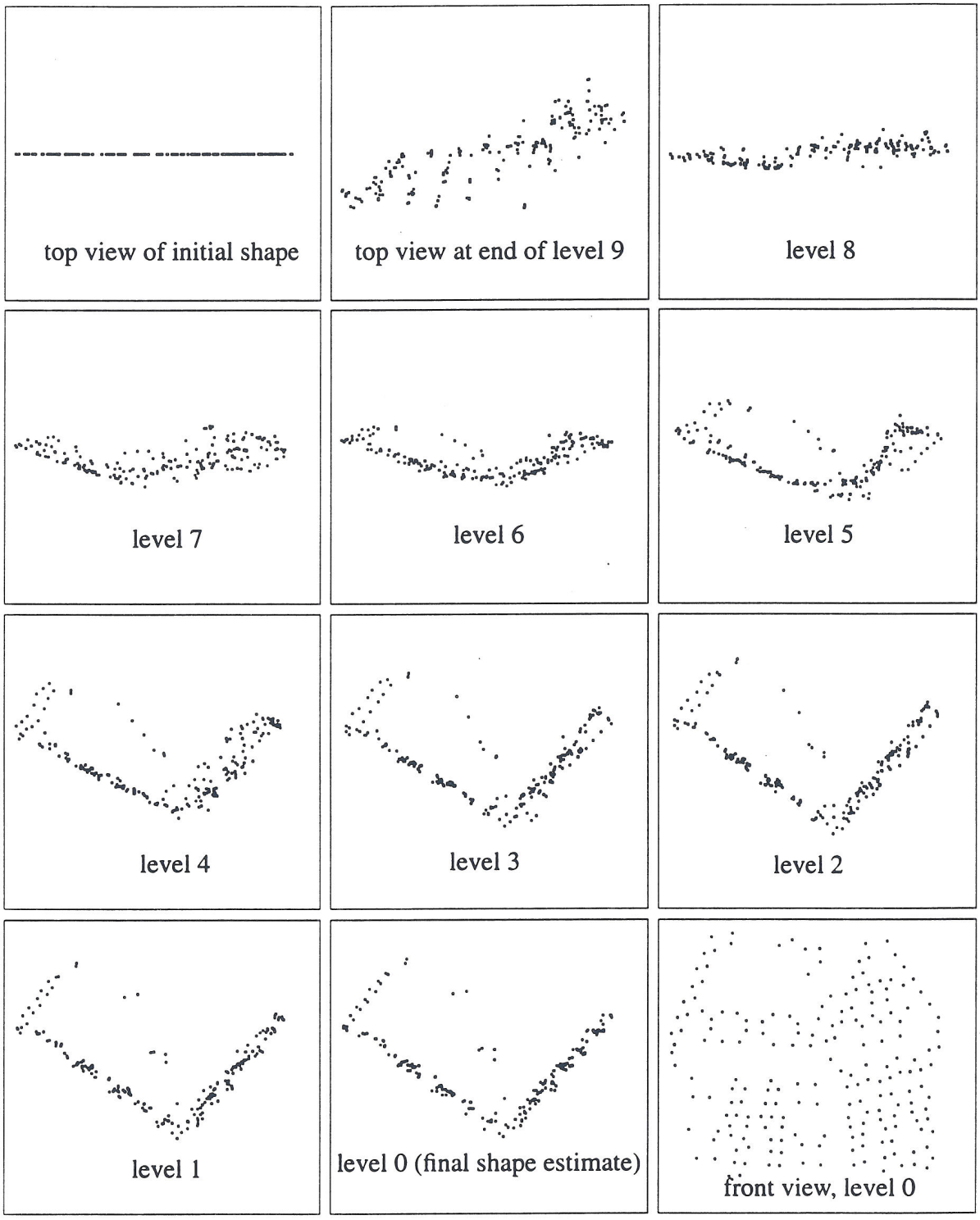


Figure 45. Shape estimates by image pyramid level

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

5. Conclusions

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

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

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

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

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

When a simple initial value is used, the Euclidean optimization method works well only in the same sorts of situations in which paraperspective also works well. By modeling perspective projection completely it avoids the need for a separate refinement step to remove perspective distortion, and is able to converge in a substantially smaller number of iterations

than the separate refinement method. The Euclidean optimization method can also be used to refine the results of either the projective or paraperspective factorization method. It can also accommodate radial distortion, as can the projective factorization method. The primary differences between paraperspective factorization, projective factorization, and Euclidean optimization are summarized in Table 4.

Table 4: Comparison of shape recovery techniques

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

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

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

5.1. Future work

5.1.1. Feature Tracking

When the camera is extremely unsteady or the camera operator pans back and forth across the field of view, the same feature points may enter and leave the field of view several times within a short period of time. The current feature tracker considers a point "lost" when it

leaves the field of view; after many features have been lost, the system searches for new features. Sometimes a point leaves the field of view for a single frame, and then reappears. The current tracker cannot recognize when a previous feature has re-entered the field of view, and therefore any point which leaves for even a frame is considered permanently lost. It may detect the feature as a new feature, and continue to track it for future frames, without recognizing it as the same as a previously seen feature. By not recognizing that the two are the same feature, significant information is lost.

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

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

5.1.2. Accommodating Long Sequences

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

An obvious solution is to separate the sequence into several smaller sequences which will

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

5.1.3. Further Incorporating Image Data

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

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

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

5.1.4. Camera calibration

The framework developed in this paper allows recovery of the camera focal length, aspect

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

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

5.1.5. Multiple cameras

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

pp. 242-248, October 1991.

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

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

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

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

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

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

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

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