A Sequential Factorization Method for Recovering Shape and Motion from Image Streams

Toshihiko Morita and Takeo Kanade

8 June 1994
CMU-CS-94-158

School of Computer Science

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

Abstract

We present a sequential factorization method for recovering the three-dimensional shape of an object and the motion of the camera from a sequence of images, using tracked features. The factorization method originally proposed by Tomasi and Kanade produces robust and accurate results incorporating the singular value decomposition. However, it is still difficult to apply the method to real-time applications since it is based on a batch-type operation and the cost of the singular value decomposition is large. We develop the factorization method into a sequential method by regarding the feature positions as a vector time series. The new method produces estimates of shape and motion at each frame. The singular value decomposition is replaced with an updating computation of only three dominant eigenvectors, which can be performed in $O(P^2)$ time, while the complete singular value decomposition requires $O(FP^2)$ operations for a $F \times P$ matrix. Also, the method is able to handle infinite sequences since it does not store any increasingly large matrices. Experiments using synthetic and real images illustrate that the method has nearly the same accuracy and robustness as the original method.

This research is sponsored by the Department of the Army, Army Research Office under Grant No. DAAH04-94-G-0006. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Army or the U.S. Government.

.

ACM Computing Reviews Keywords: 1.2.10 Vision and Scene Understanding, 1.2.9 Robotics, 1.4.6 Segmentation, 1.4.8 Scene Analysis, 1.4.1 Digitization

1. Introduction

Recovering both the 3D shape of an object and the motion of the camera simultaneously from a stream of images is an important task and has wide applicability in many tasks such as navigation and robot manipulation. Tomasi and Kanade[1] first developed a factorization method to recover shape and motion under an orthographic projection model, and obtained robust and accurate results. Poelman and Kanade[2] have extended the factorization method to scaled-orthographic projection and paraperspective projection. This method closely approximates perspective projection in most practical situations so that it can deal with image sequences which contain perspective distortions.

Although the factorization method is a useful technique, its applicability is so far limited to off-line computations for the following reasons. First, the method is based on a batch-type computation; that is, it recovers shape and motion after all the input images are given. Second, the singular value decomposition, which is the most important procedure in the method, requires $O(FP^2)$ operations for P features in F frames. Finally, it needs to store a large measurement matrix whose size increases with the number of frames. These drawbacks make it difficult to apply the factorization method to real-time applications.

This report presents a sequential factorization method that considers the input to the system as a vector time series of feature positions. The method produces estimates of shape and motion at each input frame. A covariance-like matrix is stored instead of feature positions, and its size remains constant as the number of frames increases. The singular value decomposition is replaced with a computation, updating only three dominant eigenvectors, which can be performed in $O(P^2)$ time. Consequently, the method becomes recursive.

We first briefly review the factorization method by Tomasi and Kanade. We then present our sequential factorization method in Section 3. The algorithm's performance is tested using synthetic data and real images in Section 4.

2. Theory of the Factorization Method: Review

2.1 Formalization

The input to the factorization method is a measurement matrix W, representing image positions of tracked features over multiple frames. Assuming that there are P features over F frames, and letting (x_{fp}, y_{fp}) be the image position of feature p at frame f, W is a $2F \times P$ matrix such that

$$W = \begin{bmatrix} x_{11} & \dots & x_{1P} \\ \vdots & & \vdots \\ x_{F1} & \dots & x_{FP} \\ y_{11} & \dots & y_{1P} \\ \vdots & & \vdots \\ y_{F1} & \dots & y_{FP} \end{bmatrix}. \tag{1}$$

Each column of W contains all the observations for a single point, while each row contains all the observed x-coordinates or y-coordinates for a single frame.

Suppose that the camera orientation at frame f is represented by orthonormal vectors $\boldsymbol{i}_f$, $\boldsymbol{j}_f$, and $\boldsymbol{k}_f$, where $\boldsymbol{i}_f$ corresponds to the x-axis of the image plane and $\boldsymbol{j}_f$ to the y-axis. The vectors $\boldsymbol{i}_f$ and $\boldsymbol{j}_f$ are collected over F frames into a motion matrix $M \in R^{2F \times 3}$ such that

$$M = \begin{bmatrix} \boldsymbol{i}_1^T \\ \vdots \\ \boldsymbol{i}_F^T \\ \boldsymbol{j}_1^T \\ \vdots \\ \boldsymbol{j}_F^T \end{bmatrix}. \tag{2}$$

Let $\boldsymbol{s}_p$ be the location of feature p in a fixed world coordinate system, whose origin is set at the center-of-mass of all the feature points. These vectors are then collected into a shape matrix $S \in R^{3 \times P}$ such that

$$S = \begin{bmatrix} \boldsymbol{s}_1 & \dots & \boldsymbol{s}_P \end{bmatrix}. \tag{3}$$

Note that

$$\sum_{p=1}^{P} \boldsymbol{s}_p = \boldsymbol{0}. \tag{4}$$

With this notation, the following equation holds by assuming an orthographic projection.

$$W = MS \tag{5}$$

Tomasi and Kanade[1] pointed out the simple fact that the rank of W is at most 3 since it is the product of the $2F \times 3$ motion matrix M and the $3 \times P$ shape matrix S. Based on this rank theory, they developed a factorization method that robustly recovers the matrices M and S from W.

2.2 Subspace Computation

The actual procedure of the factorization method consists of two steps. First, the measurement matrix is factorized into two matrices of rank 3 using the singular value decomposition. Assume, without loss of generality, that $2F \geq P$. By computing the singular value decomposition of $W \in R^{2F \times P}$, we can obtain orthogonal matrices $U \in R^{2F \times 3}$ and $V \in R^{P \times 3}$ such that

$$W = U\Sigma V^T, \tag{6}$$

where $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \sigma_3)$ and $\sigma_1 \geq \sigma_2 \geq \sigma_3 > 0$. In reality, the rank of W is not exactly 3, but approximately 3. U is made from the first three columns of the left singular matrix of W. Likewise, Σ consists of the first three singular values and V is made from the first three rows of the right singular matrix. By setting

$$\hat{M} = U \text{ and } \hat{S} = \Sigma V^T \tag{7}$$

we can factorize W into

$$W = \hat{M}\hat{S}, \tag{8}$$

where the product $\hat{M}\hat{S}$ is the best possible rank three approximation to W.

It is well known that the left singular vectors U span the column space of W while the right singular vectors V span its row space. The span of U, namely *motion space*, determines the motion, and the span of V, namely *shape space*, determines the shape. The rank theory claims that the dimension of each subspace is at most three, and the first step of the factorization method finds those subspaces in the high dimensional input spaces. Both spaces are said to be dual in the sense that one of them can be computed from the other. This observation helps us to further develop the sequential factorization method.

2.3 Metric Transformation

The decomposition of equation (8) is not completely unique: it is unique only up to an affine transformation. The second step of the method is necessary to find a 3×3 non-singular matrix A, which transforms $\hat{M}$ and $\hat{S}$ into the true solutions M and S as follows.

$$M = \hat{M}A \tag{9}$$

$$S = A^{-1}\hat{S} \tag{10}$$

Observing that rows $\boldsymbol{i}_f$ and $\boldsymbol{j}_f$ of M must satisfy the normalization constraints,

$$\boldsymbol{i}_f^T\boldsymbol{i}_f = \boldsymbol{j}_f^T\boldsymbol{j}_f = 1 \text{ and } \boldsymbol{i}_f^T\boldsymbol{j}_f = 0, \tag{11}$$

we obtain the system of $3F$ overdetermined equations such that

$$\begin{aligned} \hat{\boldsymbol{i}}_f^T L \hat{\boldsymbol{i}}_f &= 1 \\ \hat{\boldsymbol{j}}_f^T L \hat{\boldsymbol{j}}_f &= 1 \\ \hat{\boldsymbol{i}}_f^T L \hat{\boldsymbol{j}}_f &= 0 \end{aligned} \tag{12}$$

where $L \in R^{3 \times 3}$ is a symmetric matrix

$$L = A^T A \tag{13}$$

and, $\hat{\boldsymbol{i}}_f$ and $\hat{\boldsymbol{j}}_f$ are the rows of $\hat{M}$. By denoting $\boldsymbol{i}_f^T = [i_{f1}, i_{f2}, i_{f3}]$, $\boldsymbol{j}_f^T = [j_{f1}, j_{f2}, j_{f3}]$, and

$$L = \begin{bmatrix} l_1 & l_2 & l_3 \\ l_2 & l_4 & l_5 \\ l_3 & l_5 & l_6 \end{bmatrix}, \tag{14}$$

the system (12) can be rewritten as

$$G\boldsymbol{l} = \boldsymbol{c}, \tag{15}$$

where $G \in R^{3F \times 6}$, $\boldsymbol{l} \in R^6$, and $\boldsymbol{c} \in R^{3F}$ are defined by

$$G = \begin{bmatrix} \mathbf{g}^T(\mathbf{i}_1, \mathbf{i}_1) \\ \vdots \\ \mathbf{g}^T(\mathbf{i}_F, \mathbf{i}_F) \\ \mathbf{g}^T(\mathbf{j}_1, \mathbf{j}_1) \\ \vdots \\ \mathbf{g}^T(\mathbf{j}_F, \mathbf{j}_F) \\ \mathbf{g}^T(\mathbf{i}_1, \mathbf{j}_1) \\ \vdots \\ \mathbf{g}^T(\mathbf{i}_F, \mathbf{j}_F) \end{bmatrix}, \ \mathbf{l} = \begin{bmatrix} l_1 \\ \vdots \\ l_6 \end{bmatrix}, \ \mathbf{c} = \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}1\\1\\1\\1\\1\\1\end{matrix}}\right\} 2F \\ \left.\vphantom{\begin{matrix}1\\1\\1\end{matrix}}\right\} F \end{matrix}, \tag{16}$$

and

$$\mathbf{g}^T(\mathbf{a}_f, \mathbf{b}_f) = \begin{bmatrix} a_{f1}b_{f1} & 2a_{f1}b_{f2} & 2a_{f1}b_{f3} & a_{f2}b_{f2} & 2a_{f2}b_{f3} & a_{f3}b_{f3} \end{bmatrix}. \tag{17}$$

The simplest solution of the system is given by the pseudo-inverse method such that

$$\mathbf{l} = (G^T G)^{-1} G^T \mathbf{c}. \tag{18}$$

The vector $\mathbf{l}$ determines the symmetric matrix L, whose eigendecomposition gives A. As a result, the motion M and the shape S are derived according to equations (9) and (10).

The matrix A is an affine transform which transforms $\hat{M}$ into M in the *motion space*, while the matrix A^{-1} transforms $\hat{S}$ into S in the *shape space*. Obtaining this transform is the main purpose of the second step of the factorization method, which we call *metric transformation*.

3. A Sequential Factorization Method

3.1 Overview

In the original factorization method, there was one measurement matrix W containing tracked feature positions throughout the image sequence. After all the input images are given and the feature positions are collected into the matrix W, the motion and shape are then computed. In real-time applications, however, it is not feasible to use this batch-type scheme. It is more desirable to obtain an estimate at each moment sequentially. The input to the system must be viewed as a vector time series. At frame f, two vectors containing feature positions such that

$$\boldsymbol{x}_f^T = [x_{f1}, x_{f2}, \ldots, x_{fP}] \text{ and } \boldsymbol{y}_f^T = [y_{f1}, y_{f2}, \ldots, y_{fP}] \tag{19}$$

are given. Immediately after receiving these vectors, the system must compute the estimates of the camera coordinates $\boldsymbol{i}_f$, $\boldsymbol{j}_f$ and the shape S_f at that frame. At the next frame, new samples $\boldsymbol{x}_{f+1}$ and $\boldsymbol{y}_{f+1}$ arrive and new camera coordinates $\boldsymbol{i}_{f+1}$ and $\boldsymbol{j}_{f+1}$ are to be computed as well as an updated shape estimate S_{f+1}.

The key to developing such a sequential method is to observe that the shape does not change over time. The *shape space* is stationary, and thus, it should be possible to derive S_f from S_{f-1} without performing expensive computations.

More specifically, we store the feature vectors $\boldsymbol{x}_f$ and $\boldsymbol{y}_f$ in a covariance-type matrix $Z_f \in R^{P \times P}$ defined recursively by

$$Z_f = Z_{f-1} + \boldsymbol{x}_f \boldsymbol{x}_f^T + \boldsymbol{y}_f \boldsymbol{y}_f^T. \tag{20}$$

As shown later, the rank of Z_f is at most three and its three dominant eigenvectors Q_f span the *shape space*. Once Q_f is obtained, the camera coordinates at frame f can be computed simply by multiplying the feature vectors and the eigenvectors as follows.

$$\hat{\boldsymbol{i}}_f^T = \boldsymbol{x}_f^T Q_f, \qquad \hat{\boldsymbol{j}}_f^T = \boldsymbol{y}_f^T Q_f \tag{21}$$

This framework makes it possible to estimate camera coordinates immediately after receiving feature vectors at each frame. All information obtained up to the frame is accumulated in Q_f and used to produce the estimates at that frame.

In equation (20), the size of Z_f is fixed to $P \times P$, which only depends on the number of feature points. Therefore, the algorithm does not need to store any matrices whose sizes increase over time.

The computational effort in the original factorization method is dominated by the cost of the singular value decomposition. In the framework above, we need to compute eigenvectors of Z_f. Note that, however, we only need the first three dominant eigenvectors. Fortunately, several methods exist to compute only the dominant eigenvectors with much less computation necessary to compute all the eigenvectors. Before describing the details of our algorithm, we briefly review these techniques in the following section.

3.2 Iterative Eigenvector Computation

Among the existing methods which can compute dominant eigenvectors of a square matrix, we introduce two methods, the power method and orthogonal iteration[3]. The power method is the simplest, which computes the most dominant eigenvector, i.e., an eigenvector associated with the largest eigenvalue. It provides the starting point for most other techniques, and is easy to understand. The method of orthogonal iteration, which we adopt in our method, is able to compute several dominant eigenvectors.

3.2.1 Power Method

Assume that we want to compute the most dominant eigenvectors of an $n \times n$ matrix B. Given a unit 2-norm $\boldsymbol{q}^{(0)} \in R^n$, the power method iteratively computes a sequence of vectors $\boldsymbol{q}^{(k)}$:

for $k = 1, 2, \ldots$

$$\boldsymbol{y}^{(k)} = B\boldsymbol{q}^{(k-1)}$$

$$\boldsymbol{q}^{(k)} = \boldsymbol{y}^{(k)} / \|\boldsymbol{y}^{(k)}\|_2$$

end

The second step of the iteration is simply a normalization that prevents $\boldsymbol{q}^{(k)}$ from becoming very large or very small. The vectors $\boldsymbol{q}^{(k)}$ generated by the iteration converge to the most dominant eigenvector of B. To examine the convergence property of the power method, suppose that B is diagonalizable. That is, $X^{-1}BX = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ with an orthogonal matrix $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, and $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$. If

$$\boldsymbol{q}^{(0)} = b_1\boldsymbol{x}_1 + b_2\boldsymbol{x}_2 + \ldots + b_n\boldsymbol{x}_n \tag{22}$$

and $b_1 \neq 0$, then it follows that

$$\boldsymbol{q}^{(k)} = cB^k\boldsymbol{q}^{(0)} = c\left(\sum_{j=1}^{n} b_j\lambda_j^k\boldsymbol{x}_j\right) = cb_1\lambda_1^k\left(\boldsymbol{x}_1 + \sum_{j=2}^{n}\frac{b_j}{b_1}\left(\frac{\lambda_j}{\lambda_1}\right)^k\boldsymbol{x}_j\right) \tag{23}$$

where c is a constant. Since $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$, equation (23) shows that the vectors $\boldsymbol{q}^{(k)}$ point more and more accurately toward the direction of the dominant eigenvector $\boldsymbol{x}_1$, and the convergence factor is the ratio $r = |\lambda_2/\lambda_1|$.

3.2.2 Orthogonal Iteration

A straightforward generalization of the power method can be used to compute several dominant eigenvectors of a symmetric matrix. Assume that we want to compute p dominant eigenvectors of a symmetric matrix $B \in R^{n \times n}$, where $1 \le p \le n$. Starting with an $n \times p$ matrix Q_0 with orthonormal columns, the method of orthogonal iteration generates a sequence of matrices $Q_k \in R^{n \times p}$:

for $k = 1, 2, \ldots$

$$Y_k = BQ_{k-1}$$

$$Q_k R_k = Y_k \qquad \text{(QR factorization)}$$

end

The second step of the above iteration is the QR factorization of Y_k, where Q_k is an orthogonal matrix and R_k is an upper triangular matrix. The QR factorization can be achieved by the Gram-Schmidt process. This step is viewed as a normalization process that is similar to the normalization used in the power method.

Suppose that $X^T BX = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ is the eigendecomposition of B with an orthogonal matrix $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, and $|\lambda_1| > |\lambda_2| \ge \ldots \ge |\lambda_n|$. It has been shown in [3] that the subspace $\mathrm{range}(Q_k)$ generated by the iteration converges to $\mathrm{span}\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p\}$ at a rate proportional to $|\lambda_{p+1}/\lambda_p|$, i.e.,

$$\mathrm{dist}(\mathrm{range}(Q_k), \mathrm{range}(X_p)) \le c\left|\frac{\lambda_{p+1}}{\lambda_p}\right|^k, \tag{24}$$

where $X_p = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p]$ and c is a constant. The function *dist* represents the subspace distance defined by

$$\mathrm{dist}(\mathrm{range}(Q_k), \mathrm{range}(X_p)) = \left\| Q_k Q_k^T - X_p X_p^T \right\|_2 \tag{25}$$

The method offers an attractive alternative to the singular value decomposition in situations where B is a large matrix and a few of its largest eigenvalues are needed. In our case, these conditions clearly hold. In addition, the rank theory of the factorization method[1] guarantees that the ratio $|\lambda_4/\lambda_3|$ is very small, and as a result, we should achieve fast convergence for computing the first three eigenvectors.

3.3 Sequential Factorization Algorithm

As in the original method, the sequential factorization method consists of two steps, sequential *shape space* computation and sequential metric transformation.

3.3.1 A Sequential Shape Space Computation

In the sequential factorization method, we consider the feature vectors, $\boldsymbol{x}_f^T$ and $\boldsymbol{y}_f^T$, as a vector time series. Let us denote the measurement matrix in the original factorization method at frame f by W_f. Then, it grows in the following manner:

$$W_1 = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{y}_1^T \end{bmatrix}, \qquad W_2 = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \boldsymbol{y}_1^T \\ \boldsymbol{y}_2^T \end{bmatrix}, \qquad \dots, \qquad W_f = \begin{bmatrix} \boldsymbol{x}_1^T \\ : \\ \boldsymbol{x}_f^T \\ \boldsymbol{y}_1^T \\ : \\ \boldsymbol{y}_f^T \end{bmatrix}, \qquad \dots \tag{26}$$

Now let us define a matrix time series $Z_f \in R^{P \times P}$ by

$$Z_f = Z_{f-1} + \boldsymbol{x}_f \boldsymbol{x}_f^T + \boldsymbol{y}_f \boldsymbol{y}_f^T. \tag{27}$$

From the definition, it follows that

$$Z_f = W_f^T W_f. \tag{28}$$

Since the rank of W_f is at most three, the rank of Z_f is also at most three. If

$$W_f = U_f \Sigma_f V_f^T \tag{29}$$

is the singular value decomposition of W_f, where $U_f \in R^{2f \times 3}$ and $V_f \in R^{P \times 3}$ are orthogonal matrices, and $\Sigma_f = \mathrm{diag}(\sigma_{f,1}, \sigma_{f,2}, \sigma_{f,3})$, then

$$Z_f = (U_f \Sigma_f V_f^T)^T U_f \Sigma_f V_f^T = V_f \Sigma_f^2 V_f^T. \tag{30}$$

This means the eigenvectors of Z_f are equivalent to the right singular vectors V_f of W_f. Hence, it is possible to obtain the *shape space* by computing the eigenvectors of Z_f.

To compute V_f, we combine orthogonal iteration with updating by equation (27). Given a $P \times 3$ matrix Q_0 with orthonormal columns and a null matrix $Z_0 \in R^{P \times P}$, the following algorithm generates a sequence of matrices $Q_f \in R^{P \times 3}$:

[Algorithm (1)] ***for*** *f = 1, 2, ...*

(1) $Z_f = Z_{f-1} + \boldsymbol{x}_f \boldsymbol{x}_f^T + \boldsymbol{y}_f \boldsymbol{y}_f^T$

(2) $Y = Z_f Q_{f-1}$

(3) $Q_f R = Y$ (QR factorization)

end

The index f corresponds to the frame number and each iteration is performed frame by frame. The matrix Q_f generated by the algorithm is expected to converge to the eigenvectors V_f of Z_f. While the original orthogonal iteration works with a fixed matrix, the above algorithm works with the matrix Z_f, which varies from iteration to iteration incorporating new features. In other words, the sequential factorization method folds the update of Z_f into the orthogonal iteration. If the $\text{range}(V_f)$ randomly changes over time, no convergence is expected to appear. However, it can be shown that

$$\text{range}(V_f) = \text{range}(W_f^T) = \text{range}(S^T)\text{, for all } f. \tag{31}$$

Therefore, $\text{range}(V_f)$ is stationary and $\text{range}(Q_f)$ converges to $\text{range}(V_f)$ as in the orthogonal iteration. Even when noise exists, if the noise is uncorrelated or the noise space is orthogonal to the signal space $\text{range}(V_f)$, then $\text{range}(V_f)$ is still equal to $\text{range}(S^T)$ and the convergence can be shown. The following convergence rate of the algorithm is deduced from the convergence rate of the orthogonal iteration.

$$\text{dist}(\text{range}(Q_f), \text{range}(V_f)) \le c \prod_{k=1}^{f} \left| \frac{\sigma_{k,4}}{\sigma_{k,3}} \right| \tag{32}$$

3.3.2 Stationary Basis for the Shape Space

Algorithm (1) presented in the previous section produces the matrix Q_f, which converges to the matrix V_f that spans the *shape space*. The true shape and motion are determined from the *shape space* by a metric transformation. It is not straightforward at this point, however, to apply the metric transformation sequentially. The problem is that, even though $\text{range}(V_f)$ is stationary, the matrix V_f itself changes as the number of frames increases. This is due to the nature of singular vectors. They are the basis for the row and column subspaces of a matrix, and the singular value decomposition chooses them in a special way. They are more than just orthonormal. As a result, they rotate in the 3D subspace $\text{range}(V_f)$. Recall that the matrix A obtained in metric transformation (9) is a transform from $\hat{M}_f$ (or U_f) to M_f in the subspace $\text{range}(\hat{M}_f)$. Since V_f changes at each frame, U_f also changes. Consequently, the matrix A also changes frame by frame.

For clarity, let us denote an A matrix at frame f as A_f. The fact that A_f changes at each frame makes it difficult to estimate A_f iteratively and efficiently. Thus we need to add an additional process to obtain stationary basis for the *shape space* to update matrix A_f.

Let us define a projection matrix $H_f \in R^{P \times P}$ onto the $\text{range}(Q_f)$ by

$$H_f = Q_f Q_f^T, \tag{33}$$

where Q_f is the output from Algorithm (1). Needless to say, the rank of H_f is at most three. Since $\text{range}(Q_f)$ (=$\text{range}(\hat{M}_f)$) is stationary, the projection matrix H_f must be stationary.

It is thus possible to obtain the stationary basis for the *shape space* by replacing Q_f with the eigenvectors of H_f.

An iterative method similar to Algorithm (1) can be used to reduce the amount of computation. Given a $P \times 3$ matrix $\overline{Q_0}$ with orthonormal columns, the iterative method below generates a matrix $\overline{Q_f} \in R^{P \times 3}$, which provides the stationary basis for the *shape space*.

[Algorithm (2)] ***for*** f = 1, 2, ...

$$H_f = Q_f Q_f^T$$

$$Y = H_f \overline{Q}_{f-1}$$

$$\overline{Q}_f R = Y \qquad \text{(QR factorization)}$$

end

3.3.3 Sequential Metric Transformation

In the previous section, we derived the *shape space* in terms of $\overline{Q_f}$. Once $\overline{Q_f}$ is obtained, it is possible to compute camera coordinates $\hat{\boldsymbol{i}}_f$ and $\hat{\boldsymbol{j}}_f$ as

$$\hat{\boldsymbol{i}}_f^T = \boldsymbol{x}_f^T \overline{Q}_f, \quad \hat{\boldsymbol{j}}_f^T = \boldsymbol{y}_f^T \overline{Q_f} \tag{34}$$

These coordinates are used to solve the overdetermined equations (12) and the true camera coordinates are recovered in the same way as in the original method. Doing so, however, requires storing all the coordinates $\hat{\boldsymbol{i}}_f$ and $\hat{\boldsymbol{j}}_f$, the number of which may be very large. Instead, we use the following sequential algorithm.

[Algorithm (3)] ***for*** *f = 1, 2, ...*

$$\hat{\boldsymbol{i}}_f^T = \boldsymbol{x}_f^T \overline{Q}_f, \quad \hat{\boldsymbol{j}}_f^T = \boldsymbol{y}_f^T \overline{Q_f}$$

$$D_f = D_{f-1} + \boldsymbol{g}(\hat{\boldsymbol{i}}_f, \hat{\boldsymbol{i}}_f)\,\boldsymbol{g}^T(\hat{\boldsymbol{i}}_f, \hat{\boldsymbol{i}}_f) + \boldsymbol{g}(\hat{\boldsymbol{j}}_f, \hat{\boldsymbol{j}}_f)\,\boldsymbol{g}^T(\hat{\boldsymbol{j}}_f, \hat{\boldsymbol{j}}_f) + \boldsymbol{g}(\hat{\boldsymbol{i}}_f, \hat{\boldsymbol{j}}_f)\,\boldsymbol{g}^T(\hat{\boldsymbol{i}}_f, \hat{\boldsymbol{j}}_f)$$

$$E_f = E_{f-1} + \boldsymbol{g}(\hat{\boldsymbol{i}}_f, \hat{\boldsymbol{i}}_f) + \boldsymbol{g}(\hat{\boldsymbol{j}}_f, \hat{\boldsymbol{j}}_f)$$

end

Let G_f and $\boldsymbol{c}_f$ be the matrices G and $\boldsymbol{c}$ at frame f, where G and $\boldsymbol{c}$ are defined in Section 2.3 From the definition, it follows that

$$D_f = G_f^T G_f \tag{35}$$

$$E_f = G_f^T \boldsymbol{c}_f. \tag{36}$$

Assigning equations (35) and (36) to equation (18), we have

$$\boldsymbol{l}_f = D_f^{-1} E_f \tag{37}$$

which gives the symmetric matrix L_f. The eigendecomposition of L_f yields the affine transform A_f and, as a result, the camera coordinates and the shape are obtained as follows:

$$\boldsymbol{i}_f^T = \hat{\boldsymbol{i}}_f^T A_f \qquad \boldsymbol{j}_f^T = \hat{\boldsymbol{j}}_f^T A_f \tag{38}$$

$$S_f = A_f^{-1} \bar{Q}_f \tag{39}$$

Algorithm (3) followed by equations (37), (38), and (39) completes the sequential method. The size of matrices D_f and E_f are fixed to 6×6 and 6×1, and the method does not store any matrices that grow, even in the sequential metric transformation.

4. Experiments

4.1 Synthetic Data

In this section we compare the accuracy of our sequential factorization method with that of the original factorization method. Since both methods are essentially based on the rank theory, we do not expect any difference in the results. Our purpose here is to confirm that the sequential method has the same accuracy of shape and motion recovery as the original method.

4.1.1 Data Generation

The object in this experiment consists of 100 random feature points. The sequences are created using a perspective projection of those points. The image coordinates of each point are perturbed by adding Gaussian noise, which we assume to simulate tracking error and image noise. The standard deviation of the Gaussian noise is set to two pixels of a 512×512 pixel image. The distance of the object center from the camera is fixed to ten times the object size. The focal length is chosen so that the projection of the object covers the whole 512×512 image. The camera is rotated as shown in Figure 1, while the object is translated to keep its image at the image center. Quantization errors are not added since we assume that we are able to track features with a subpixel resolution.

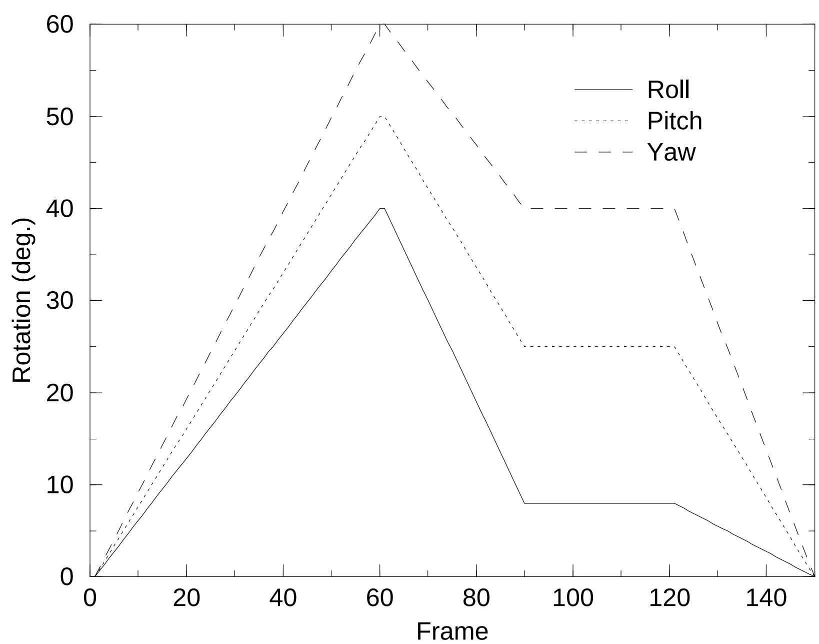

Figure 1 True camera motion

The camera roll, pitch, and yaw are varied as shown in this figure. The sequence consists of 150 frames.

When discussing the accuracy of the sequential method, one needs to consider its dynamic property regarding the 3D recovery. The accuracy of the recovery at a particular frame by the sequential method depends on the total amount of motion up to that time, since the recovery is made only from the information obtained up to that time. At the beginning of an image sequence, for example, the motion is generally small, so high accuracy can not be expected. The accuracy generally improves as the motion becomes larger. The original method does not have this dynamic property, since it is based on a batch-type scheme and uses all the information throughout the sequence.

In order to compare both methods under the same conditions, we perform the following computations beforehand. First, we form a submatrix W_f, which only contains the feature positions up to frame f. The original factorization is applied to the submatrix, then the results are kept as solutions at frame f. They are the best estimates given by the original method. Repeating this process for each frame, we derive the best estimates, with which our results are compared.

4.1.2 Accuracy of the Sequential Shape Space Computation

We first discuss the convergence property of the sequential *shape space* computation. The sequential factorization method starts with Algorithm (1) in Section 3.3.1, iteratively generating the matrix Q_f which is an estimate for the true *shape space* S^T. Let us represent the estimation error with respect to the true *shape space* by

$$E_s = \mathrm{dist}\,(\mathrm{span}\,(Q_f), \mathrm{span}\,(S^T)) \tag{40}$$

Recall that the function *dist* provides a notion of difference between two spaces. On the other hand, the original method produces the best estimate for the *shape space* by computing the right singular vectors V_f of the submatrix W_f, and its error with respect to the true *shape space* is also represented by

$$E_o = \mathrm{dist}\,(\mathrm{span}\,(V_f), \mathrm{span}\,(S^T)) \tag{41}$$

Comparing both errors, Figure 2 shows that they are almost identical. That is, the errors given by the sequential method are almost equal to those given by the original method.

At the beginning of the sequence, the amount of motion is small and both errors are relatively large. The ratio of the 4th to 3rd singular values, shown in Figure 3, also indicates that it is difficult to achieve good accuracy at the beginning. Both errors, however, quickly become smaller as the camera motion becomes larger. After about the 20th frame, constant errors of 3×10^{-2} are observed in this experiment.

The solutions given by the two methods are so close that the graphs are completely overlapped. Thus, we also plot their difference defined by

$$\Delta E = \mathrm{dist}\,(\mathrm{span}\,(Q_f), \mathrm{span}\,(V_f)) \tag{42}$$

in Figure 4. Although ΔE is relatively large at the beginning, it quickly becomes

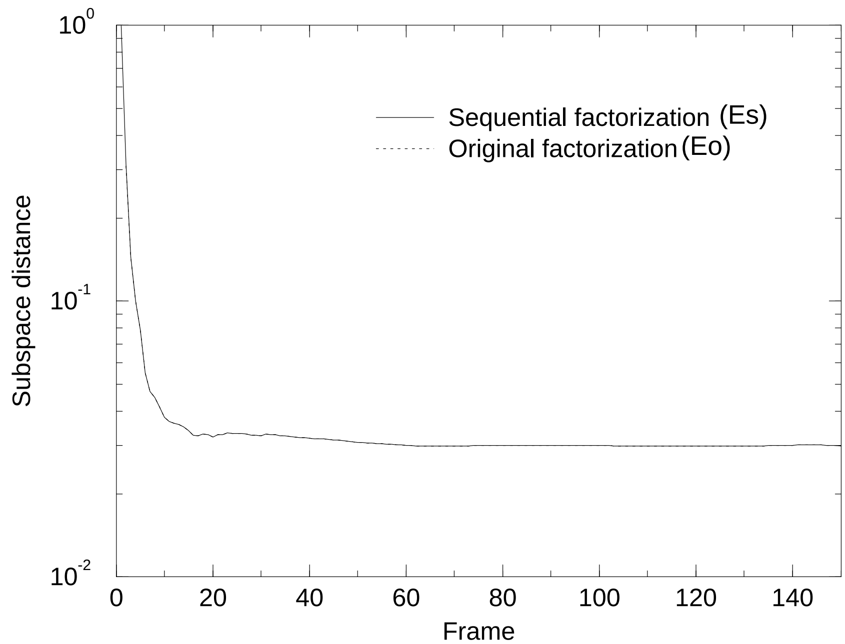

Figure 2 Shape space errors

Shape space estimation errors by the sequential method (solid line) and the original method (dashed line) with respect to the true shape space. The errors are defined by subspace distance and plotted logarithmically.

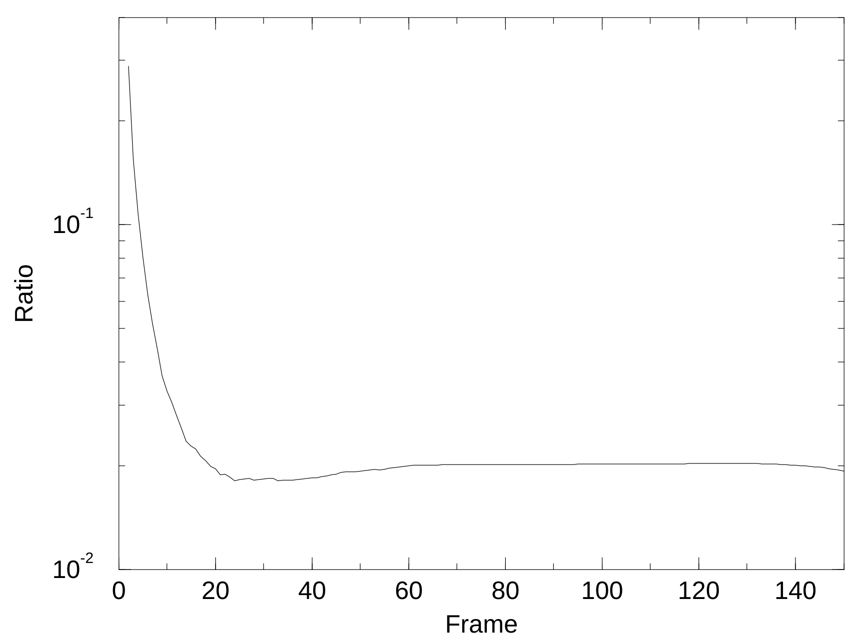

Figure 3 Singular value ratio

The ratio of the 4th to 3rd singular values, that is σ_4/σ_3.

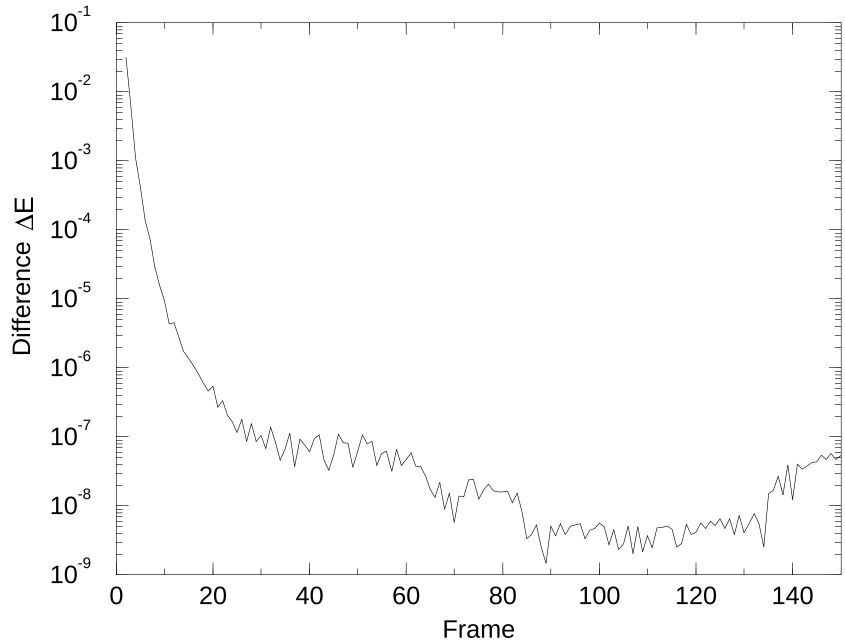

Figure 4 Difference of shape space errors

The difference of the estimates by the sequential and original methods, versus the frame number. The difference is plotted logarithmically.

very small. In fact, after about the 30th frame, ΔE is less than 1×10^{-7}, while E_s and E_o are both 3×10^{-2}.

4.1.3 Accuracy of the Motion and Shape Recovery

The three plots of Figure 5 show errors in roll, pitch, and yaw in the recovered motion: the solid lines correspond to the sequential method, the dotted lines to the original method. The difference in motion errors between the original and sequential methods is quite small.

Both results are unstable for a short period at the beginning of the sequence. After that, they show two kinds of errors: random and structural. Random errors are due to Gaussian noise added to the feature positions. Structural errors are due to perspective distortion, and relate to the motion patterns. The structural errors show a negative peak at about the 60th frame and are almost constant between the 90th and 120th frames. Note the pattern corresponds to the motion pattern shown in Figure 1.

Of course, these intrinsic errors cannot be eliminated in the sequential method. The point to observe is that the differences between the two solutions are sufficiently smaller than the intrinsic errors.

Shape errors which are compared in Figure 6 also indicate the same results. Again, the differences between the two methods are quite small compared to the intrinsic errors which the original method possesses. Note that no Gaussian noise appears in the shape errors since they are averaged over all the feature points.

We conclude from these results that the sequential method is nearly as accurate as the original method except that some extra frames are required to converge.

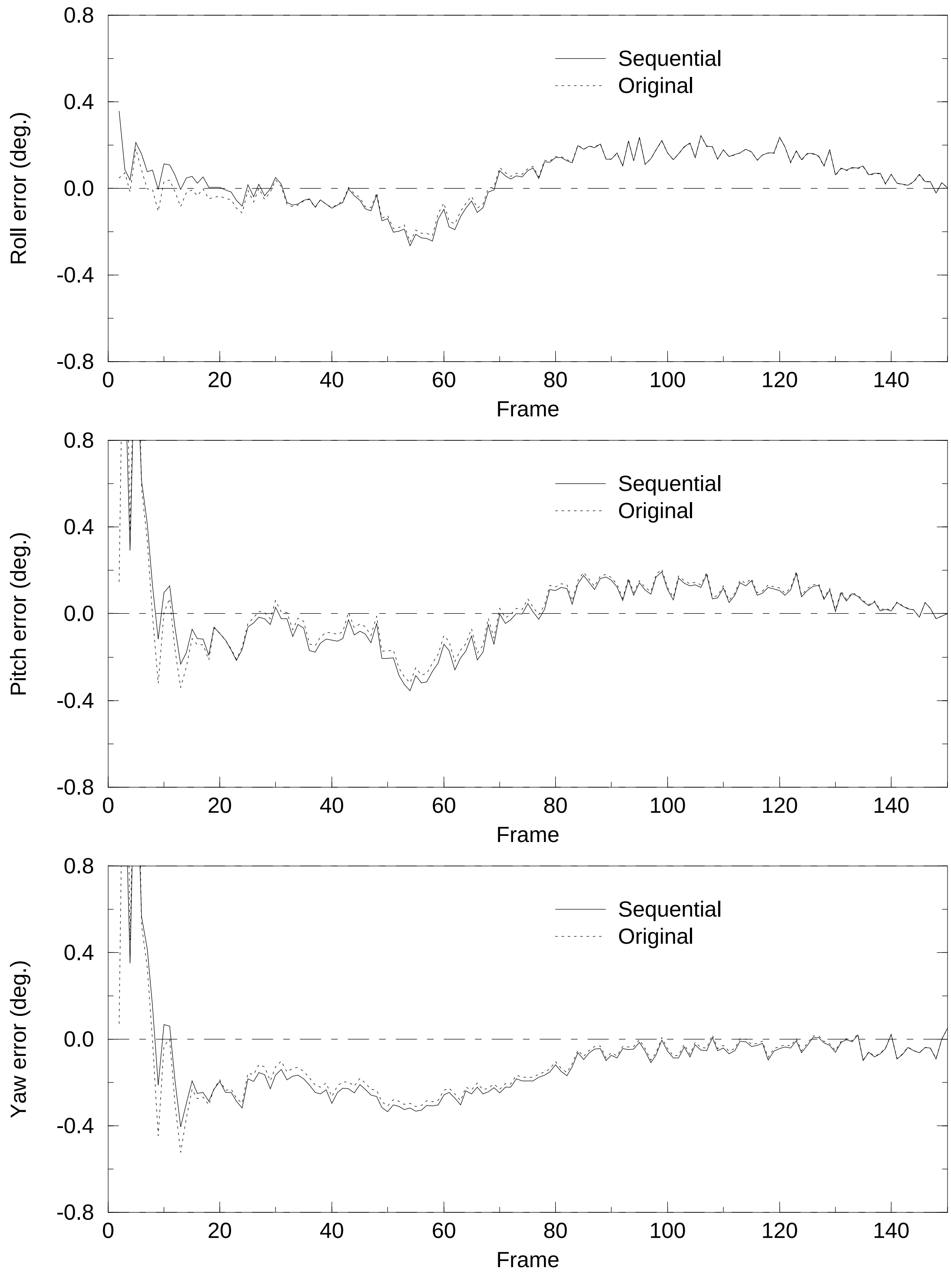

Figure 5 Motion errors

Errors of recovered camera roll (top), pitch (middle), and yaw (bottom). The errors given by the sequential method are plotted with solid lines, while the errors given by the original method are plotted with dotted lines.

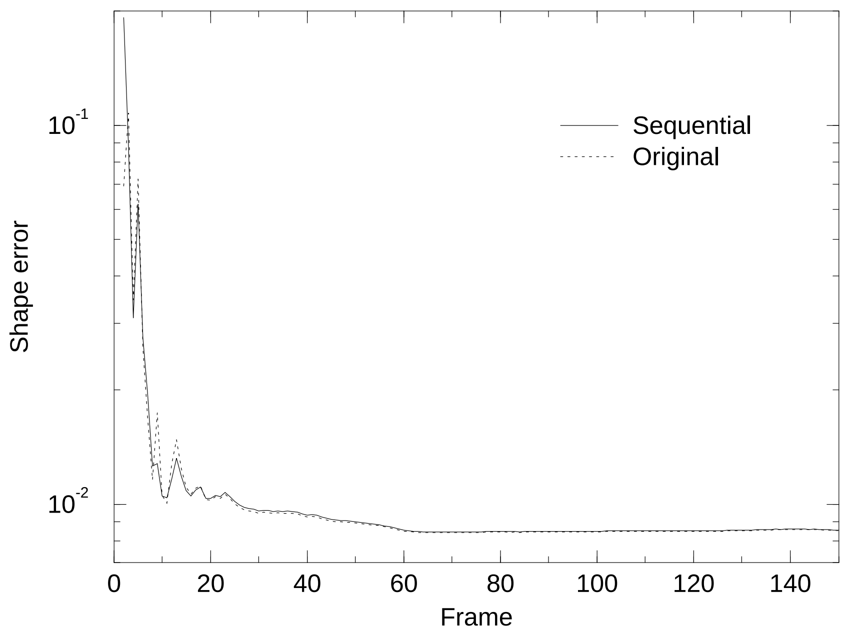

Figure 6 Shape error

This figure compares the shape errors given by the two method. The errors given by the sequential method are plotted with solid lines, while the errors given by the original method are plotted with dotted lines. The errors are computed as the root-mean-square errors of the recovered shape with respect to the true shape, at each frame.

4.2 Real Images

Experiments were performed on two sets of real images. The first set is an image sequence of a satellite rotating in space. Another experiment uses a long video recording (764 images) of a house taken with a hand-held camera. These experiments demonstrate the applicability of the sequential factorization method in real situations. In both experiments, features are selected and tracked using the method presented by Tomasi and Kanade[1].

4.2.1 Satellite Images

Figure 7 shows an image of the satellite with selected features indicated by small squares. The image sequence was digitized from a video recording[4] actually taken by a space shuttle astronaut. The feature tracker automatically selected and tracked 32 features throughout the sequence of 101 images. Of these, five features on the astronaut maneuvering around the satellite were manually eliminated because they had a different motion. Thus, the remaining 27 features were processed. Figure 8 shows the recovered motion in terms of roll, pitch, and yaw. The side view of the recovered shape is displayed in Figure 9, where the features on the solar panel are marked with opaque squares and others with filled squares. No ground-truth is available for the shape or the motion in this experiment. Yet, it appears that the solutions are satisfactory, since the features on the solar panel almost lie in a single line in the side view.

4.2.2 House Images

Figure 10 shows the first image of the sequence used in the second experiment. Using a hand-held camera, one of the authors took this sequence while walking. It consists of 764 images which correspond to about 25 seconds. The feature tracker detected and tracked 62 features. The recovered motion and shape are shown in Figures 11 and 12. It is clearly seen that the shape is qualitatively correct. It is also reasonable to observe that only the camera yaw is increasing, because the camera is moving parallel to the ground. In addition, note that the computed roll motion reveals the pace of the recorder's steps, which is about 1 step per second.

Further evaluation of accuracy in these experiments is difficult. However, this qualitative analysis of the results with real images, and quantitative analysis of the results with synthetic data essentially shows that the sequential method works as well with real images as the original batch method.

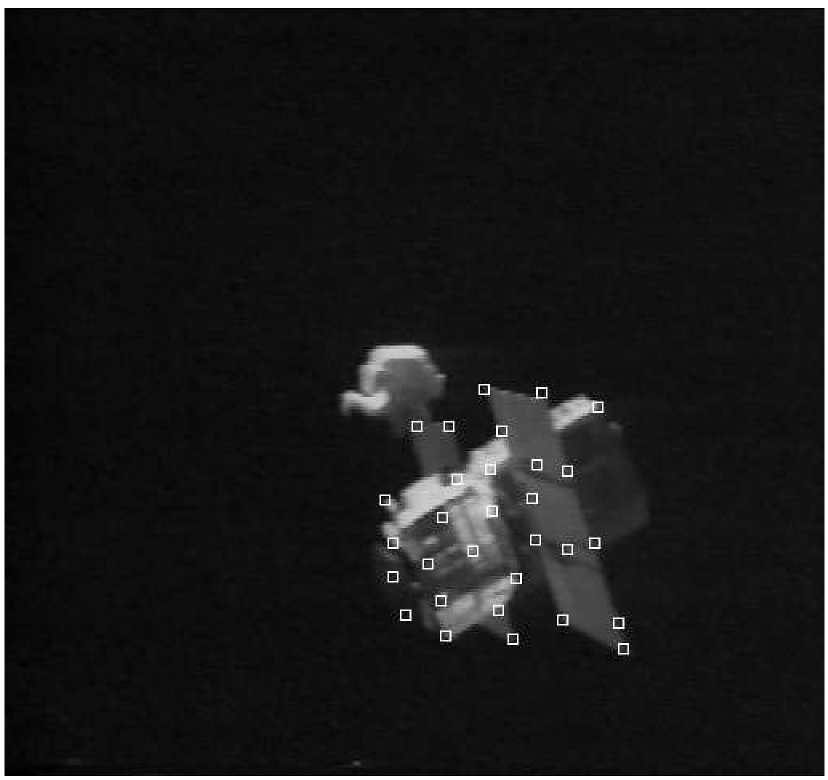

Figure 7 An image of a satellite

The first frame of the satellite image sequence. The superimposed squares indicate the selected features.

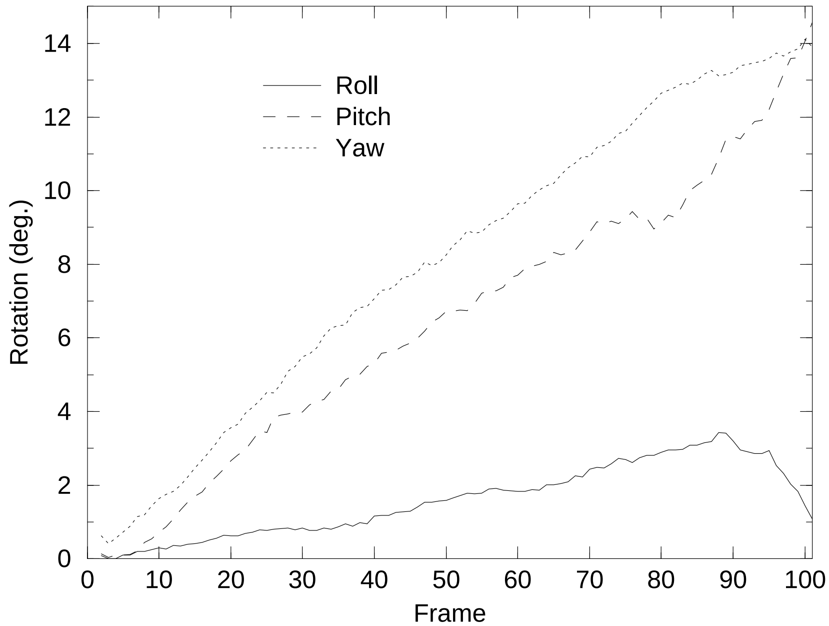

Figure 8 Recovered motion of satellite

Recovered camera roll (solid line), pitch (dashed line), and yaw (dotted line) for the satellite image sequence.

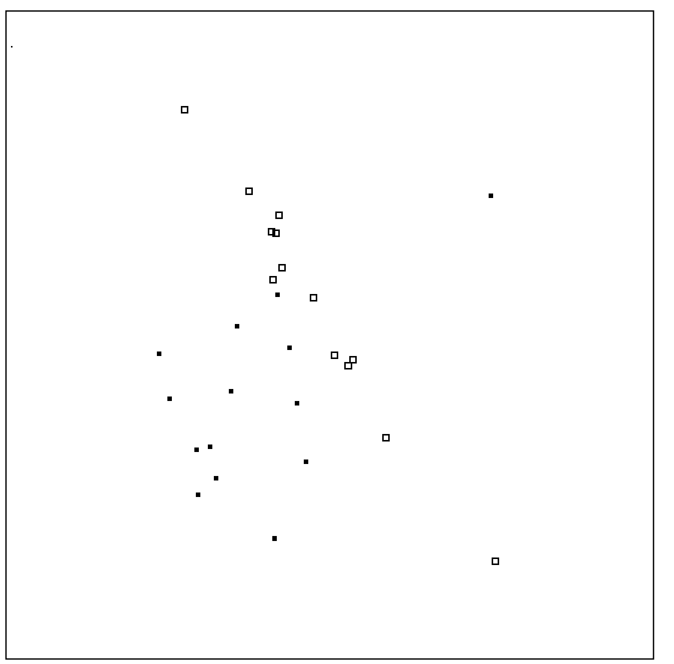

Figure 9 Side view of the recovered shape

A side view of the recovered shape of the satellite. The features on the solar panel are shown with opaque squares and others with filled squares. Notice that the features on the solar panel correctly lie in a single plane.

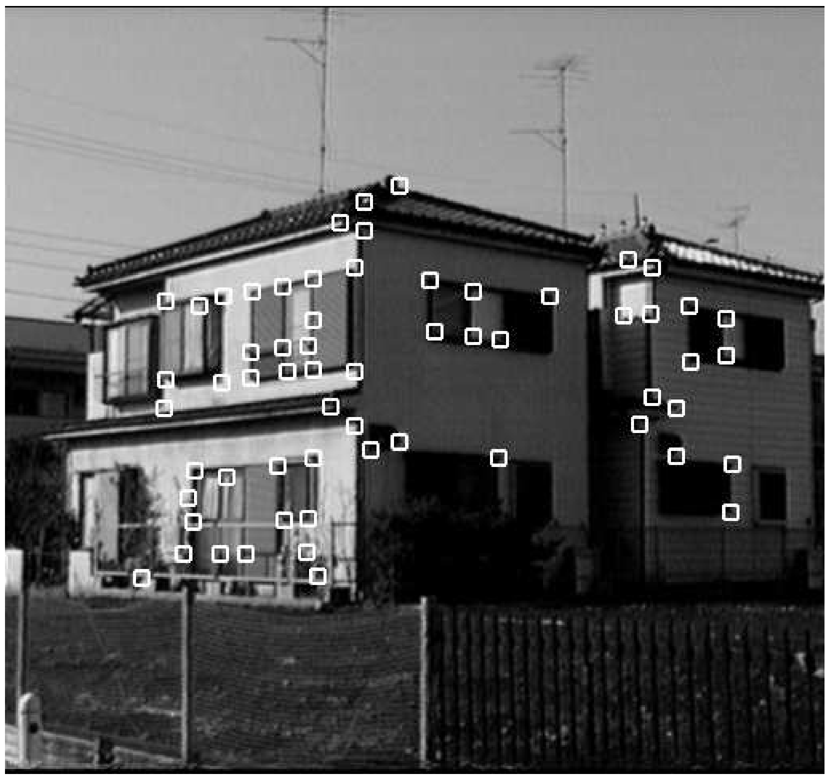

Figure 10 An image of a house

The first frame of the house image sequence. The superimposed squares indicate the selected features.

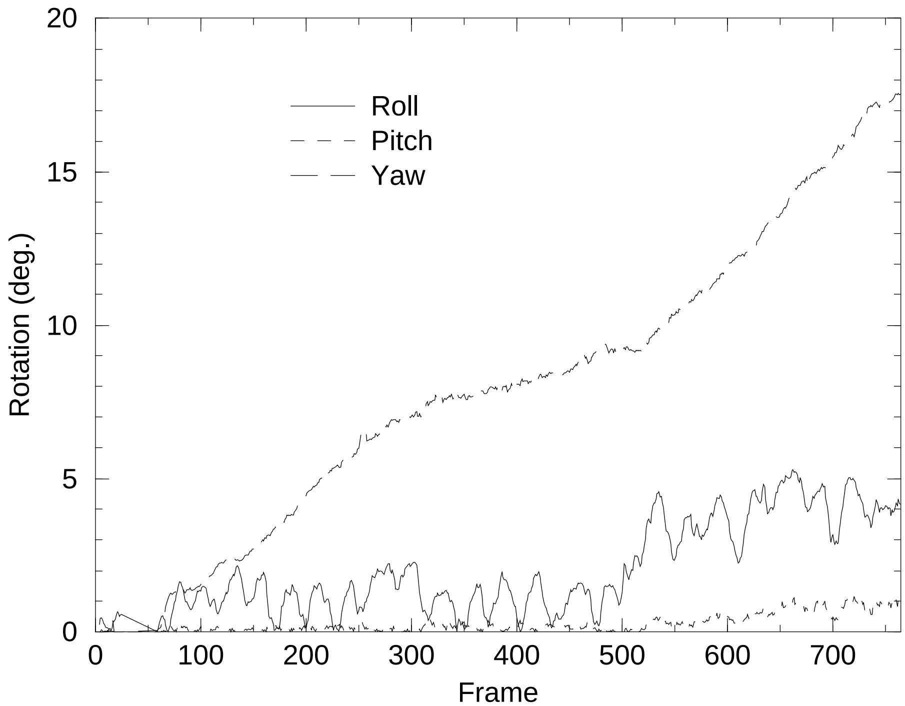

Figure 11 Recovered motion of house

Recovered camera roll (solid line), pitch (dashed line), and yaw (dotted line) for the house image sequence.

Figure 12 Top view of the recovered shape

A view of the recovered shape of the house from above. The features on the two side walls are correctly recovered.

4.3 Computational Time

Finally, we compare the processing time of the sequential method with the original method. The computational complexity of the original method is dominated by the cost of the singular value decomposition, which needs $14FP^2 + 11P^3/3$ computations for a $2F \times P$ measurement matrix with $2F \geq P$ [5]. Note that F corresponds to the number of frames and P to the number of features. On the other hand, the complexity of the sequential method is $22P^2 + 44P$ for computing dominant eigenvectors, plus $4P^2$ for updating the Zmatrix. Computing the solution for frame F, therefore, takes only $O(P^2)$ using the sequential method, while the original method would require $O(FP^2)$ operations.

Figure 13 shows the actual processing time of the sequential method on a Sun4/10 compared together with that of the original method. The number of features varied from 10 to 500, while the number of frames was fixed at 120. The processing time for selecting and tracking features was not included. The singular value decomposition of the original method is based on a routine found in [6]. The results sufficiently agree with our analysis above. In addition, when the number of features is less than 40, the sequential method is possible to run within 1/30 ms, which means video-rate processing on a Sun4/10.

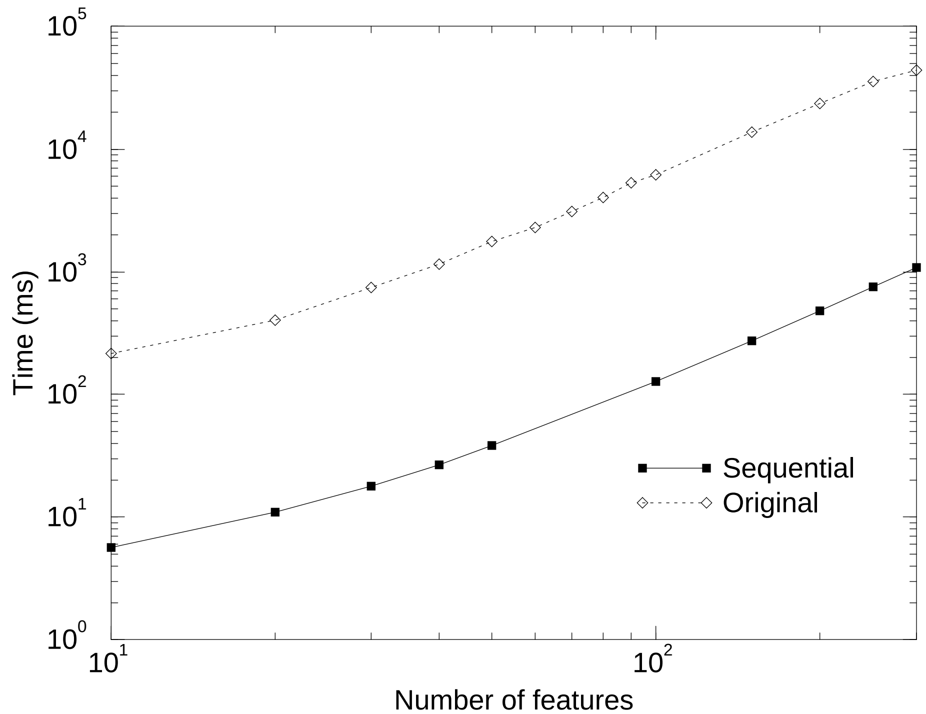

Figure 13 Processing time

The processing time of the sequential method on a Sun4/10 (solid line) compared with that of the original method (dotted line), as a function of the number of features which is varied from 10 to 500. The number of frames is fixed at 120.

5. Conclusions

We have presented the sequential factorization method, which provides estimates of shape and motion at each frame from a sequence of images. The method produces as accurate and robust results as the original method, while significantly reducing the computational complexity. The reduction in complexity is important for applying the factorization method to real-time applications. Furthermore, the method does not require storing any growing matrices so that its implementation in VLSI or DSP is feasible.

Faster convergence in the *shape space* computation could be achieved using more sophisticated algorithms such as the orthogonal iteration with Ritz acceleration[3] instead of the basic orthogonal iteration. Also, it is possible to use scaled orthographic projection or paraperspective projection[2] to improve the accuracy of the sequential factorization method.

Acknowledgments

The authors wish to thank Conrad J. Poelman and Richard Madison for their helpful comments.

References

[1] Carlo Tomasi and Takeo Kanade, *Shape and motion from image streams: a factorization method*, Technical Report CMU-CS-91-172, Carnegie Mellon University, 1991. Later published as Carlo Tomasi and Takeo Kanade, "Shape and Motion from Image Streams Under Orthography: A Factorization Method", *International Journal of Computer Vision*, Vol. 9, No. 2, pp. 137-154, November 1992.

[2] Conrad J. Poelman and Takeo Kanade, *A paraperspective factorization method for shape and motion recovery*, Technical Report CMU-CS-92-208, Carnegie Mellon University, 1992. Revised and superceded as Technical Report CMU-CS-93-219, Carnegie Mellon University, 1993. Part of the latter technical report is presented at and included in the Proceedings of the Third European Conference on Computer Vision, Vol I, pp. 97-110, Stockholm, Sweden, May 1994.

[3] Gene H. Golub and Charles F. Van Loan, *Matrix computations, second edition*, The Johns Hopkins University Press, 1989.

[4] *Satellite rescue in space: highlights of shuttle flights 41C & 51A*, #V41, The Holiday Video Library, Finley-Holiday Film Corporation.

[5] Pierre Comon and Gene H. Golub, *Tracking a few extreme singular values and vectors in signal processing*, Proceedings of the IEEE, Vol. 78, no. 8, pp. 1327-1343, 1990.

[6] 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.