Correspondence-free Synchronization and Reconstruction in a Non-rigid Scene

Lior Wolf and Assaf Zomet
School of Computer Science and Engineering,
The Hebrew University,
Jerusalem 91904, Israel
e-mail: {lwolf,zomet}@cs.huji.ac.il

Abstract

3D reconstruction of a dynamic non-rigid scene from features in two cameras usually requires synchronization and correspondences between the cameras. These may be hard to achieve due to occlusions, wide base-line, different zoom scales, etc. In this work we present an algorithm for reconstructing a non-rigid scene from sequences acquired by two uncalibrated non-synchronized fixed orthographic cameras.

We assume that different points might be tracked in the two sequences. The only constraint used to relate the two cameras is that every 3D point tracked in one sequence can be described as a linear combination of some of the 3D points tracked in the other sequence. Such constraint is useful, for example, for articulated objects. We may track some points on an arm in the first sequence, and some other points on the same arm in the second sequence. On the other extreme, this model can be used for generally moving points tracked in both sequences without knowing the correct permutation. In between, this model can cover non-rigid bodies, with local rigidity constraints.

We present linear algorithms for synchronizing the two sequences and reconstructing the 3D points tracked in both views. Outlier points are automatically detected and discarded. The algorithm can handle both 3D objects and planar objects in a unified framework, therefore avoiding numerical problems existing in other methods.
1 Introduction

Many traditional algorithms for reconstructing a 3D scene from two or more cameras require the establishment of correspondences between the images. This becomes challenging in some cases, for example when the cameras have different zoom factors, or large vergence (wide-baseline stereo) [11, 10]. Using a moving video camera rather than a set of static cameras helps in overcoming some of the correspondence problems, but may decrease the stability and accuracy of the reconstruction. Moreover, reconstruction from a moving camera becomes harder if not impossible when the scene is not rigid.

In this paper we present a linear algorithm for synchronizing two fixed orthographic cameras viewing a non-rigid scene, and reconstructing that scene. We track feature points in both sequences, with no correspondence available between the sequences. Moreover, The points tracked by the two cameras may be different. Instead of using correspondence we use a weaker assumption: every 3D point tracked in the second sequence can be described as a linear combination of some of the 3D points tracked in the first sequence. The coefficients of this linear combination are unknown but fixed throughout the sequence. Since the cameras view the same scene, this assumption is reasonable. For example, when a point tracked in the second camera belongs to some rigid region of the scene, it can be expressed as a linear combination of some other points on the same region, tracked in the first camera.

Our non-rigidity concept is quite strong. We set no explicit limitation on the motion of each specific point. The linear combination with fixed coefficients assumption suggests that there exists a local rigid structure or that some of the points, which are allowed to move freely, are tracked in both sequences (although the correspondence between the sequences is unknown to us).

This paper is divided into two main parts. In the first part we show how to synchronize two video sequences. In the second part we show how to reconstruct the 3D scene.

1.1 Previous work

In [4] several sequences without correspondences between them were aligned in space and time. In [13] a similar setting was used for synchronization and self-calibration of a stereo rig. In both settings the motion matrices of each camera were computed independently, which is problematic for dynamic scenes. After computing the motion matrices they were combined to express the
inter-camera rigidity.

A bulk of related work are the class-based approaches [12, 9, 3, 2]. They assume that the points in 3D can be expressed as a linear combination of a small morphological model basis. This basis can either be provided as a learned prior [9, 12], or computed by the algorithm [3, 2]. In this work, rather than expressing all the points as a linear combination of some model, we set some of the points as linear combinations of other points, thus expressing local rigidity of subsets of the features.

However, the assumptions of the class-based approach and of our presented work can be expressed in a similar way: the matrix containing the positions of the 2D tracked points in different times can be expressed as a product of two matrices. Therefore, a single-camera factorization algorithm designed for class based setting [3] can be adapted for reconstruction in the proposed setting.

The method proposed in this paper uses two cameras and can be seen as a compromise between having the prior 3D model given as input to the algorithm [12, 9] and computing the model from the data using factorization [3, 2]. Our approach has two main advantages over the factorization methods. First, in factorization methods there is an ambiguity expressed in an unknown invertible matrix between the two factors. Second, the proposed method is robust to outliers as it includes a way to discard outliers by testing each feature separately. Using factorization on data with outliers degrades the quality of the results. We elaborate more on this point in Section 5.

A solution for 3D reconstruction of \( k \) moving bodies from a sequence of orthographic views was presented in [5] and later on in [8] using factorization. These algorithms exploit the fact that there are more measurements arising from each one of the objects than the minimal number required to span this motion. This redundancy is used to identify the motions. In this work the rigidity relations between points is weaker, allowing many different motions of small bodies with different dimensions. For example, rigidity constraints of lines and planes can be exploited to express non-rigid bodies, e.g. trees and clothes. In our experiments we compared our method to [5], and showed the advantage of using two cameras in the \( k \) body setting.

2 Formal statement

We use the affine camera model. A \( 2 \times 4 \) projection matrix projects 3D points tracked over time \( 1 \leq t \leq T \) onto an image. The tracked 3D points \( P_j^{(t)} \) \( (j = 1...n) \) are given as 3-vectors
(X Y Z)\top$, and their projection is given as:

\[
\begin{pmatrix}
    x_i^{(t)} \\
    y_i^{(t)}
\end{pmatrix} = 
\begin{pmatrix}
    a_1 & a_2 & a_3 \\
    b_1 & b_2 & b_3
\end{pmatrix} P_i^{(t)} +
\begin{pmatrix}
    a_4 \\
    b_4
\end{pmatrix}
\]

Instead of looking directly at the point measurements \((x, y)^T\), it would be more convenient to look at the flow from some reference frame. (We use the frame where \(t=0\) as our reference frame). The flow is given by:

\[
\begin{pmatrix}
    u_i^{(t)} \\
    v_i^{(t)}
\end{pmatrix} = 
\begin{pmatrix}
    a_1 & a_2 & a_3 \\
    b_1 & b_2 & b_3
\end{pmatrix} (P_i^{(t)} - P_i^{(0)}) = 
\begin{pmatrix}
    a_1 & a_2 & a_3 \\
    b_1 & b_2 & b_3
\end{pmatrix} dP_i^{(t)}
\]

where \(dP_i^{(t)} = P_i^{(t)} - P_i^{(0)}\) is the motion of point \(i\) from time \(t = 0\) to the time of frame \(t\).

Using the coordinate system of the first camera as our 3D world coordinate system, the projections onto the two cameras are given by:

\[
\begin{pmatrix}
    u_i^{(t)} \\
    v_i^{(t)}
\end{pmatrix} = 
\begin{pmatrix}
    1 & 0 & 0 \\
    0 & 1 & 0
\end{pmatrix} dP_i^{(t)},
\begin{pmatrix}
    \hat{u}_i^{(t)} \\
    \hat{v}_i^{(t)}
\end{pmatrix} = 
\begin{pmatrix}
    \alpha_1 & \alpha_2 & \alpha_3 \\
    \beta_1 & \beta_2 & \beta_3
\end{pmatrix} \hat{d}P_i^{(t)}
\]

The constraint we use in this work is that every 3D point tracked in the second sequence \(\hat{P}_i^{(t)}, 1 \leq i \leq n\) can be expressed as a linear combination of the points tracked in the first sequence:

\[
\hat{P}_i^{(t)} = (P_1^{(t)} \quad P_2^{(t)} \quad \ldots \quad P_n^{(t)}) Q_i
\]

Where \(Q_i\) is the vector specifying the coefficients of the linear combination, such that \(\Sigma_j Q_i(j) = 1\).

Note that this linear combination applies also to the 3D displacements \(d\hat{P}_i^{(t)}\) and therefore:

\[
\begin{pmatrix}
    \hat{\alpha}_1 & \hat{\alpha}_2 & \hat{\alpha}_3 \\
    \hat{\beta}_1 & \hat{\beta}_2 & \hat{\beta}_3
\end{pmatrix} = 
\begin{pmatrix}
    u_1^{(t)} & u_2^{(t)} & \ldots & u_n^{(t)} \\
    v_1^{(t)} & v_2^{(t)} & \ldots & v_n^{(t)} \\
    dZ_1^{(t)} & dZ_2^{(t)} & \ldots & dZ_n^{(t)}
\end{pmatrix} Q_i
\]
Where $dZ_i^{(t)}$ are the displacements along the Z axis of the points tracked in the first sequence and the last equality is derived by substituting the projection equation of the first camera.

Reorganizing the terms we get:

\[
\begin{pmatrix}
\hat{u}_i^{(t)} \\
\hat{v}_i^{(t)}
\end{pmatrix}
= \begin{pmatrix}
Q_i(1) \hat{a}_1 & Q_i(1) \hat{a}_2 & Q_i(1) \hat{a}_3 & Q_i(2) \hat{a}_1 & Q_i(2) \hat{a}_2 & \ldots & Q_i(n) \hat{a}_3 \\
Q_i(1) \hat{b}_1 & Q_i(1) \hat{b}_2 & Q_i(1) \hat{b}_3 & Q_i(2) \hat{b}_1 & Q_i(2) \hat{b}_2 & \ldots & Q_i(n) \hat{b}_3
\end{pmatrix}
\begin{pmatrix}
u_1^{(t)} \\
v_1^{(t)} \\
Z_1^{(t)} \\
\vdots \\
u_n^{(t)} \\
v_n^{(t)} \\
Z_n^{(t)}
\end{pmatrix}
\] (2)

Let $A_i = (Q_i(1) \hat{a}_1, Q_i(1) \hat{a}_2, Q_i(1) \hat{a}_3, Q_i(2) \hat{a}_1, Q_i(2) \hat{a}_2, \ldots, Q_i(n) \hat{a}_3)$, and $B_i$ be the similar expression involving $\hat{b}_1, \hat{b}_2$ and $\hat{b}_3$. Let $\hat{M}$ be the matrix whose rows are $A_1, \ldots, A_m, B_1, \ldots, B_m$; and let $C$ be the matrix whose columns are $(u_1^{(t)}, v_1^{(t)}, Z_1^{(t)}, \ldots, u_n^{(t)}, v_n^{(t)}, Z_n^{(t)})^T$ for $1 \leq t \leq T$.

Note that $\hat{M}$ is point-dependent, but does not vary with time, and that the matrix $C$ is time-dependent, but does not depend on points in the second sequence.

Let $\hat{U}$ and $\hat{V}$ be the matrices containing the flow in the second sequences:

\[
\hat{U} = \begin{bmatrix}
\hat{u}_1^{(1)} & \ldots & \hat{u}_m^{(1)} \\
\vdots & \ddots & \vdots \\
\hat{u}_1^{(T)} & \ldots & \hat{u}_m^{(T)}
\end{bmatrix}, \quad \hat{V} = \begin{bmatrix}
\hat{v}_1^{(1)} & \ldots & \hat{v}_m^{(1)} \\
\vdots & \ddots & \vdots \\
\hat{v}_1^{(T)} & \ldots & \hat{v}_m^{(T)}
\end{bmatrix}
\]

Stacking all of the equations of the form of eq. 2, $1 \leq i \leq m$ we get:

\[
\begin{pmatrix}
\hat{U}^T \\
\hat{V}^T
\end{pmatrix}_{2m \times T} = \hat{M}_{2m \times 3n} C_{3n \times T}
\] (3)

A similar expression can be derived for the first camera as well:

\[
\begin{pmatrix}
U^T \\
V^T
\end{pmatrix}_{2n \times T} = M_{2n \times 3n} C_{3n \times T}
\] (4)
Where $U, V$ contain the flow in the first sequence, and $M$ is the diagonal $3n \times 3n$ matrix after dropping every third row.

3 Synchronization and Action Indexing

Given two sequences of a dynamic scene in the above setting, we first synchronize them. Given this synchronization, we later describe how to compute the structure of the scene across time.

The ability to synchronize two sequences describing the same action can be used for other dynamic-scene applications such as action indexing. Two matching actions should have a minimal synchronization error. We construct a single matrix by stacking the flow in both sequences side by side: $E = (U, V, \hat{U}, \hat{V})$. This is a matrix of size $T \times (2n + 2m)$ and since we consider long sequences we would expect its rank to be $(2n + 2m)$. However, this holds only when there is no relation between the sequences, i.e when they are not synchronized. Combining equations (3) and (4) we get:

$$E^T = \begin{pmatrix} M \\ \hat{M} \end{pmatrix}_{(2m+2m) \times 3n} C_{3n \times T}$$

(5)

therefore the rank of $E$ is bounded by $3n$.

This upper bound on the rank is usually not tight, depending on the rigidity of the scene. For example, when the scene contains $k$ rigid bodies the rank is bounded by $3k$. However, in this case we expect the rank of the the matrix $E$ to decrease in the synchronized case at least as much as it decreases in the unsynchronized case.

Outliers (points which do not satisfy our assumptions even remotely) pose a different kind of problem for this bound - they might increase the rank of the matrix $E$. However, for outliers we would expect the rank of the matrix $E$ to increase by the same amount both for the synchronized case and for the unsynchronized case. Therefore in both cases (partially rigid scenes and outliers) we would expect the rank in the synchronized case to be lower than the rank in the unsynchronized case.

In practice, however, synchronizing by examining the rank of a matrix is problematic since the measurement based matrices will usually be of full rank due to noise. In order to deal with this problem we propose two solutions - the first one is by examining the effective rank of the measurements, and the second is by considering principle angles between the measurements of
both sequences.

3.1 Synchronization using effective ranks

The first method does not consider \( n \), the number of points tracked in the first camera, for the rank test, but instead uses a heuristic to define \( \tilde{n} \), the effective rank of \((U, V)\). We compute the singular values \( s_1, ..., s_{n_1} \) of \((U, V)\), and set \( \tilde{n} = \min_j \{ \sum_{k=1}^{2j} s_k > U \} \) for some threshold \( U \) (we used \( U = 0.99 \sum_{k=1}^{n_1} s_k \)). This is equivalent to finding the minimal rank of a matrix in a given error bound under Frobenius norm.

We choose the synchronization for which the algebraic measure \( g(E) \) defined below is minimized. The minimization is carried out over all possible synchronizations where each possible synchronization provides a different matrix \( E \). Let \( e_1, ..., e_{n_2} \) be all the singular values of \( E \). Then \( g \) is defined by:

\[
g(E) = \sum_{k=3\tilde{n}+1}^{n_2} e_k
\]

\( g \) measures the amount of energy in the matrix beyond the expected rank bound. It is expected that if the rank bound on \( E \) is violated by noise, this measure would be low, otherwise, if it is broken by non-synchronized data, this measure should be high.

3.2 Synchronization using principle angles

A second method for synchronization is to exploit the constraint that for synchronized sequences the linear subspaces spanned by the columns of \([U, V]\) and of \([U', V']\) intersect. The rank of the intersection is \( 2n \).

A stable way of measuring the amount of intersection between two linear subspaces is by using the principle angles [6]. Let \( U_A, U_B \) represent two linear subspaces of \( R^n \). The principal angles \( 0 \leq \theta_1 \leq ... \leq \theta_k \leq (\pi/2) \) between the two subspaces are uniquely defined by:

\[
\cos(\theta_k) = \max_{u \in U_A} \max_{v \in U_B} u^T v
\]

subject to:

\[
u^T u = v^T v = 1, \quad u_i = 0, \quad v_i = 0, \quad i = 1, ..., k - 1
\]

A convenient way to compute the principal angles is via the QR factorization, as follows. Let
\[ A = Q_AR_A \text{ and } B = Q_BR_B \] where \( Q \) is an orthonormal basis of the respective subspace and \( R \) is an upper-diagonal \( k \times k \) matrix with the Gram-Schmidt coefficients representing the columns of the original matrix in the new orthonormal basis. The singular values \( \sigma_1, ..., \sigma_k \) of the matrix \( Q_A^TQ_B \) are the principal angles \( \cos(\theta_i) = \sigma_i \).

In practice, due to noise the linear subspaces are inflated, so we use SVD instead of QR decomposition in order to get the orthogonal basis of the column spaces, but take only the leading vectors of these basis using the effective rank method described above.

As mentioned above, the linear subspaces spanned by the columns of \([U, V]\) and of \([U', V']\) intersect and the rank of the intersection is \(2n\). Hence, the first \(2n\) principle angles between these column spaces vanish. In practice we are effected by noise and would expect those first principle angles to be close to zero. In order to achieve synchronization we consider a function of the first few principle angles (e.g. their average). This function is computed for every possible displacement and the synchronization is chosen as to minimize this function.

### 3.3 Direct synchronization using brightness measurements

The synchronization procedures described above are based on having a set of points tracked over time in each one of the video sequences. The accuracy of our results is therefore effected by the accuracy of the tracker which we use. Most trackers, however, have difficulties maintaining accurate positions for all tracked points over time, especially in dynamic scenes.

A method for synchronizing using only brightness measurements in the images is going to be presented next. This method is based on the work of Irani [7] where it was shown that the flow matrices \( U \) and \( V \) are closely related to matrices computed directly out of the derivatives of the brightness values of the images \( I(t), 0 \leq t \leq T \). These derivatives are the image derivatives \( I_x \) and \( I_y \) of the reference image \((t=0)\) and the derivative over time \( I'_t = I(j) - I(0) \). The relation between the flow matrices and the image derivatives - “the generalized Lukas & Kanade constraint” - is expressed by the following equation:

\[
[U, V]_{T \times 2n} \begin{bmatrix} \tilde{A} \tilde{B} \\ \tilde{B} \tilde{C} \end{bmatrix}_{2n \times 2n} = [G, H]_{T \times 2n}
\]

Where \( \tilde{A}, \tilde{B} \) and \( \tilde{C} \) are diagonal \( n \times n \) matrices with the diagonal elements being \( a_i = \sum_k (I_x(k))^2 \),

8
\[ b_i = \sum_k (I_x(k)I_y(k)) \] and \[ c_i = \sum_k (I_y(k))^2 \], respectively. The summation in \( a_i, b_i \) and in \( c_i \) is at a small window around pixel \( i \). The \((i, j)\)th elements of \( G \) and \( H \) are given by \[ g_{ij} = -\sum_k (I_x(k)I_x^*(k)) \] and \[ h_{ij} = -\sum_k (I_y(k)I_y^*(k)) \] where the summation is over a small neighborhood around pixel \( j \).

In practice we compute the matrices \( G \) and \( H \) only for those pixels for which there is enough gradient information in their neighborhood, hence the matrix \( \begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{B} & \tilde{C} \end{bmatrix} \) is invertible and the column spaces of \( U \) and \( V \) are the same as the column spaces of \( G \) and \( \tilde{H} \), respectively. A similar constraint connects the flow matrices \( U' \) and \( V' \) in the second sequence to the measurement matrices \( G' \) and \( H' \) computed for this sequence. We can therefore make the observations below, which are analogous to the observations we made earlier on the ranks of \( U, V, U', V' \). In the synchronized case:

- The effective rank \( \tilde{n} \) of the matrix \([U, V]\) is the same as the effective rank of the matrix \([G, H]\).
- The effective rank of the matrix \([G, H, G', H']\) is bounded by \( 3\tilde{n} \).
- The column spaces of the matrix \([G, H]\) and of the matrix \([G', H']\) intersect in a linear subspace of rank \( \tilde{n} \).

These observations can lead to synchronization methods analogous to those presented for the flow matrices. Care should be taken regarding the number of frames used. The above “generalized Lukas and Kanade constraint” assumes infinitesimal motion. This assumption might hold for several frames surrounding the reference frame. In practice, we divide every sequence into small continuous fragments (small windows in time) of length up to 17 frames each. We choose the middle frame of each fragment as a reference frame, and compute the measurement matrices \( G \) and \( H \) for each such fragment separately. We then compare each fragment in the first sequence to each fragment in the second sequence, and compute an algebraic error based on the principle angels.

Let \( k, k' \) be the number of fragments we extract from the first and second sequence, respectively. The comparison of fragments between both sequences results in a matrix of size \( k \times k' \). A synchronization will appear as the line in this matrix which has the minimum values. The offset
of the line determines the shift of the synchronization and its slope determines the ratio of frame rates between the sequences.

4 Reconstruction

Given the synchronization of the two sequences, the next stage is 3D reconstruction. Let \( D = \text{null}\left( \begin{bmatrix} U^T \\ V^T \end{bmatrix} \right) \) be a matrix whose columns are all orthogonal to the rows of \( \begin{bmatrix} U^T \\ V^T \end{bmatrix} \). The rows of \( \begin{bmatrix} U^T \\ V^T \end{bmatrix} \) are the rows of the matrix \( C \) (defined in Section 2 below eq. (2)) which correspond to motion along the \( X \) and \( Y \) axes. Hence the first two out of every 3 rows in the matrix product \( CD \) vanish, and all which is left is the information regarding the \( Z \) coordinates of the 3D displacements.

By multiplying both sides of eq.( 3) with \( D \) we get:

\[
K = \left( \begin{bmatrix} \hat{U}^T \\ \hat{V}^T \end{bmatrix} \right)^T D = \tilde{MCD} = \begin{bmatrix} \hat{a}_1 Q_1^T \\ \hat{b}_1 Q_1^T \\ \vdots \\ \hat{a}_n Q_n^T \\ \hat{b}_n Q_n^T \end{bmatrix} \begin{bmatrix} Z_1^{(1)} & \cdots & Z_1^{(T)} \\ \vdots & \ddots & \vdots \\ Z_n^{(1)} & \cdots & Z_n^{(T)} \end{bmatrix} D
\]

Observe that each odd row of \( K \) equals the next row of \( K \) multiplied by the ratio \( r = a(3)/b(3) \). Therefore, this ratio can be recovered by dividing the rows. For added robustness we use the median of all the measurements of \( r \) from all points to be the \( r \) we estimate. In Section 5 we elaborate on the robustness of the algorithm.

Consider the vector \( l = [1, -r]^T \). \( l \) is a direction orthogonal to the projection of the \( Z \) axis to the second image. Using this direction we eliminate the unknown depths of the points.

We multiply both sides of eq.( 1) with \( l^T \) and get:

\[
l^T \begin{bmatrix} u_i^{(t)} \\ v_i^{(t)} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} u_1^{(t)} & u_2^{(t)} & \cdots & u_n^{(t)} \\ v_1^{(t)} & v_2^{(t)} & \cdots & v_n^{(t)} \end{bmatrix} Q_i
\]

(6)

Where \( c_1 = l^T(\hat{a}_1, \hat{b}_1) \), \( c_2 = l^T(\hat{a}_2, \hat{b}_2) \), and the third row of the projection matrix just vanishes

\footnote{We assume that \( T > 2n \) (i.e. there are more frames than twice the effective rank), otherwise synchronization is not possible. If there are less frames than the number of tracked points \( T < 2n \), we take \( D \) to be the vector corresponding to the smallest singular value.}
\((\bar{I}^T(\bar{e}_3, \bar{b}_3) = 0)\).

This is a bilinear system in the unknowns \(c_1, c_2\) and \(Q_i\). For each \(i\) we have \(T\) equations. We solve this system by linearizing, i.e. by defining a vector of unknowns \(u_i = (c_1 Q_i, c_2 Q_i)\) and converting the equations into a linear system.

We recover \(Q_i\) from \(u_i\) up to a scale by factoring the elements of \(u_i\) as a \(2 \times n\) matrix of rank 1\(^2\) and the scale is adjusted such that \(\sum_i Q_i = 1\).

Once the \(Q_i\)'s are recovered, we can recover the points in the first image corresponding to the points tracked in the second image:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{pmatrix}
\hat{P}_i^{(t)} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
P_1^{(t)} & P_2^{(t)} & \ldots & P_n^{(t)}
\end{pmatrix} Q_i = 
\begin{pmatrix}
x_1^{(t)} & x_2^{(t)} & \ldots & x_n^{(t)} \\
y_1^{(t)} & y_2^{(t)} & \ldots & y_n^{(t)}
\end{pmatrix}
Q_i
\]

The problem of reconstructing the points tracked in the second image \(\hat{P}_i^{(t)}\) is hence reduced to a simple stereo problem in each frame.

Reconstructing \(P_i^{(t)}\) from \(\hat{P}_i^{(t)}\) is possible by taking the pseudo-inverse of the matrix whose columns are \(Q_i\), \(i = 1...m\).

## 5 Outlier Rejection

In the basis of our algorithm and of many similar feature based-algorithms lies the assumption that the input points fit some model and that they were tracked correctly along the sequence. Outlier points, e.g. points which were not tracked properly, may in some cases have a strong effect on the accuracy of the results. In this section we show that outlier points tracked in the first camera have a negligible effect on the results, and that outlier points tracked the second camera can be detected automatically. Hence the algorithm is robust to outlier tracks from both sequences.

The first step in the algorithm consists of computing the matrix \(D\), i.e. finding the subspace orthogonal to the 2D trajectories of all points tracked in the first camera. In case of outlier points,

\(^2\)It is better to first recover \(c_1, c_2\) using all of the \(u_i\)'s, since they are not point-dependent.
this subspace may be reduced, but still the resulting subspace is orthogonal to all trajectories. Therefore, the computation of \( r = \frac{a_3}{b_3} \) described in the previous section is not influenced by outliers in the first sequence. As for the computation of \( Q_i \) - 3D points in the second camera are a combination of some of the inlier 3D points tracked in the first camera. The coefficients of \( Q_i \) corresponding to the outlier points vanish.

Outliers tracks in the first sequence therefore have little effect as long as there are not too many of them. We next show how to eliminate outliers tracks from the second sequence. For each point in the second image, several measurements of the ratio \( r \) can be computed. A simple test, for example by measuring the variance of the estimates of \( r \) for each point and across the points can be used to reject outliers.

In our experiments usually \( 2n > T > 2\bar{n} \), i.e. the matrix \([U, V]\) had rank \( T \) due to noise, but the effective rank \( 2\bar{n} \) was smaller. For numerical reasons, we chose \( D \) to be the vector corresponding to the smallest singular value. Therefore, every point in the second image had a single measurement of the ratio \( r \). Outlier points were selected by computing the median of the \( r \) values of all points, and discarding points with \( r \) value far from this median.

6 Experiments

6.1 Synchronization

We have tested our algorithm on scenes of moving people. The first set of experiments tested the camera synchronization application. Two video sequences were captured by two unsynchronized cameras viewing the same non-rigid scene. In the first experiment we used tracked points, and examined the algebraic error \( g \) (as explained in Section 3) at different time shifts, and chose the shift with the minimal error. Results are presented in Fig. 1.

In the second experiment we used the direct method described above in Section 3.3. The input to this experiment was two video sequences of a person performing 8 simple motions such as clapping or jumping in place. We wanted to show that not only can we synchronize the whole sequences, but that we can also locate locations in the first sequence which correspond to subsequences of the second sequence showing one action each. Each such subsequence was about 20 seconds in length. The whole sequence was few minutes long.

Since the direct method (section 3.3) assumes small image motion, we applied it to small
Figure 1. Sequences synchronization for cameras with different zoom factors (a) and for cameras with large vergence (b). Corresponding frames from the sequences of the first graph are shown in (c) and (d). The first graph reflects the oscillations in the input motion (walking people). The sequences of the second graph are the same as in Fig. 4. The graphs show the algebraic error $g$ vs. the frame offset.

temporal windows $w(t)$ (typically of 15 consecutive frames), and smoothed and decimated the images. For each such pair of windows $w(t_1), w(t_2)$ taken from the two sequences, we measured the smallest principle angles between the subspaces defined for these windows. Since the first sequence is much longer than any subsequence that we tried to locate, we got a very long rectangular matrix of such measurements for each one of the 8 actions.

A typical result of this process is shown in Fig. 2-a. This matrix is actually a cropped version of the whole rectangular matrix, and the figure shows only the part where synchronization was found. For each pair of time locations $t_1, t_2$, the matrix contains the principle angles between the linear spaces computed for $w(t_1)$ on the first sequence and $w(t_2)$ on the second sequence. Since $t_1, t_2$ are both incremented by the same shift, the optimal synchronization is visualized by a 45° dark line in the image.

The mean value along such lines is shown in Fig. 2-b, for all possible time shifts. Corresponding frames of the two synchronized sequences are shown in Fig. 2-c,d. We have confirmed these results
Figure 2. Sequences synchronization for cameras with large vergence using direct method. (a) shows the distance (minimal principle angle, see text) between temporal windows at different temporal shifts in the two sequences. By averaging diagonal directions in this matrix, one achieves a score for different synchronizations, as shown in (b). The second smallest local minimum corresponds to the similar “stop” motion. Corresponding frames from the sequences are shown in (c) and (d).

by manually synchronizing the input sequences.

We have repeated this experiment for each one of the 8 actions. For 6 of them we found the correct location of the action in the first sequence at the global minimum of those mean diagonal values. For the rest two actions, we were able to find the correct synchronization at the second smallest minimum.

6.2 Reprojection

The next experiment tested the reprojection stage. Using the proposed algorithm we established correspondences between the two cameras, by transferring tracked points from one sequence to the other sequence. The results are presented in Fig. 3. Unfortunately, more than 50% of the tracks
were not very good, and our algorithm was not able to handle these tracks as well as we hoped. Therefore, while in the synthetic experiments the algorithm proved to be robust to outliers, in the presented experiment we have chosen a subset of good tracks in both sequences manually.

![Figure 3. Feature reprojection between two sequences of a non-rigid body.](image)

6.3 Clustering

In order to show the benefits of using our algorithm for the settings of independently moving rigid bodies we implemented the algorithm of Costeira and Kanade [5] and applied it to each of the two sequences separately. Based on the results we clustered the points to different rigid objects, thus comparing the results using real noisy data. The clustering method for both algorithms was based on an affinity matrix of the points. In our algorithm an affinity matrix can be defined by the normalized correlation between the coefficient vectors $Q_i$. 

15
In [5] an affinity matrix was defined in a similar manner. As pointed out in [8], once the affinity matrix is defined, the choice of the clustering criteria is arbitrary. We used a software package [1] including several clustering algorithms, and chose the Ward Method which gave the best results for both algorithms. The results, presented in Fig. 4, show that the use of an additional camera in our algorithm improves the results.

![Feature clustering for k-body scene](image)

**Figure 4.** Feature clustering for k-body scene, a comparison with the Costeira-Kanade algorithm. (a),(b) show examples of input images taken by the two cameras at different times. Note the large vergence between the cameras. (c),(d) show the two clusters found by the method in this paper. (e),(f) show the two clusters found by the Costeira-Kanade K-body factorization algorithm.

7 Summary

We have shown how the correspondence between two video sequences in a non-rigid scene can be solved by using a simple assumption relating the points tracked in the two video sequences. The proposed algorithm is simple, linear and robust to outliers. It combines the measurement
from all frames in a single computation, thus minimizing the error in all frames simultaneously.

In the future we hope to be able to apply our methods presented here not only for synchronization, but also for action recognition. Each action will be represented by a collection of small fragments, each one less than half a second long. A dictionary containing many such fragments and their associated actions will be constructed. An action can be recognized by comparing each such fragment to the fragments in the dictionary.

References