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 dynamic scene from sequences acquired by two uncalibrated non-synchronized fixed affine cameras.

It is assumed that (possibly) different points are 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) [14, 11]. 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, the reconstruction from a moving camera becomes harder if not impossible when the scene is not rigid.

In this paper we present an algorithm for reconstructing a non-rigid scene from two fixed video cameras. It is assumed that feature points are tracked for each of the cameras, but no correspondences are available between the cameras. Moreover, The points tracked by the two cameras may be different. Instead 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 part of the scene, it can be expressed as a linear combination of some other points on the same part, tracked in the first camera.

Our non-rigidity concept is quite strong. We put no *explicit* limitation on the motion of the points from one time to another. The linear combination with fixed coefficients assumption suggests that there exist local rigid structure or that some of the points, which are allowed to move freely, are tracked in both sequences (although we do not know the correspondence between the sequences).

This paper is divided to two main sections. First, we show how to synchronize two vidoe sequences.

1.1 Previous work

Using several sequences without correspondences between them was done in [4] for alignment in space and time, and was followed by [18] for synchronization and self-calibration of cameras on a stereo rig. The algorithms suggested compute the motion matrices of each camera independently, which is problematic for dynamic scenes. After computing the motion matrices they combined to express the inter-camera rigidity.

Closely related body of work are the class-based approaches [15, 10, 3, 2]. These assume that the points in 3D can be expressed as a linear combination of a small morph model basis. The basis can either be provided as learned prior[10, 15], or computed by the algorithm [3, 2]. Rather than expressing all the points as linear combinations of some model, in this work we set some of the points as linear combination of other points, thus expressing local rigidity of subsets of the features.

Still, the assumptions of the class-based approach and the presented work can be expressed in a similar way: The matrix containing the positions of the 3D points in different times has a low rank. Therefore a single-camera factorization algorithm designed for class based setting [3] can be adapted also for reconstruction in our 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 to the algorithm [15, 10] and computing the model from the data using factorization [3, 2]. This way we get 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 ourliers 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. 4.1.

A solution for 3D reconstruction of k moving bodies from a sequence of affine views was presented in [5] and later on in [9] using factorization. Improved solutions were suggested in [17, 16]. 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 (usually the number of bodies is small). 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, local rigidity constraints of lines and planes can be exploited to express non-rigid bodies, e.g. trees and clothes. In the experiments we compared our method to [5], and showed that even in a k body setting as well it is worthwhile using a second camera.

2 Formal statement

Assume two affine cameras (2×4 matrices) view different 3D points across time $1 \leq t \leq T$. Let $P_j^{(t)} \Re^3, j = 1...n$, be the 3D points tracked by the first camera.

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

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

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

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

We will use the coordinate system of the first camera as our world (3D) coordinate system and so our projections onto the two cameras are given by:

$$\begin{pmatrix} u_i^{(t)} \\ v_i^{(t)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} dP_i^{(t)}, \begin{pmatrix} \hat{u}_i^{(t)} \\ \hat{v}_i^{(t)} \end{pmatrix} = \begin{pmatrix} \hat{a_1} & \hat{a_2} & \hat{a_3} \\ \hat{b_1} & \hat{b_2} & \hat{b_3} \end{pmatrix} \hat{dP}_i^{(t)}$$

Where we denote points related to the second camera using a similar notation with a $\hat{hat}$.

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

$$\hat{P}_i^{(t)} = (\, P_1^{(t)} \quad P_2^{(t)} \quad ... \quad P_n^{(t)} \,)\, Q_i$$

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

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

$$\begin{pmatrix} \hat{u}_i^{(t)} \\ \hat{v}_i^{(t)} \end{pmatrix} = \begin{pmatrix} \hat{a_1} & \hat{a_2} & \hat{a_3} \\ \hat{b_1} & \hat{b_2} & \hat{b_3} \end{pmatrix} (\, dP_1^{(t)} \quad dP_2^{(t)} \quad ... \quad dP_n^{(t)} \,)\, Q_i =$$

$$= \begin{pmatrix} \hat{a_1} & \hat{a_2} & \hat{a_3} \\ \hat{b_1} & \hat{b_2} & \hat{b_3} \end{pmatrix} \begin{pmatrix} u_1^{(t)} & u_2^{(t)} & ... & u_n^{(t)} \\ v_1^{(t)} & v_2^{(t)} & ... & v_n^{(t)} \\ dZ_1^{(t)} & dZ_2^{(t)} & ... & dZ_n^{(t)} \end{pmatrix} Q_i \tag{1}$$

Where $dZ_i^{(t)}$ are the displacements along the Z axis of the points tracked in the first sequence and the last equality is by substituting the projection equation of the first camera.

Reorganizing the terms we get:

$$\begin{pmatrix} \hat{u}_i^{(t)} \\ \hat{v}_i^{(t)} \end{pmatrix} = \begin{pmatrix} Q_i(1)\hat{a_1} & Q_i(1)\hat{a_2} & Q_i(1)\hat{a_3} & Q_i(2)\hat{a_1} & Q_i(2)\hat{a_2} & ... & Q_i(n)\hat{a_3} \\ Q_i(1)\hat{b_1} & Q_i(1)\hat{b_2} & Q_i(1)\hat{b_3} & Q_i(2)\hat{b_1} & Q_i(2)\hat{b_2} & ... & Q_i(n)\hat{b_3} \end{pmatrix} \begin{pmatrix} u_1^{(t)} \\ v_1^{(t)} \\ Z_1^{(t)} \\ \vdots \\ u_n^{(t)} \\ v_n^{(t)} \\ Z_n^{(t)} \end{pmatrix} \tag{2}$$

Let $A_i = (Q_i(1)\hat{a_1}, Q_i(1)\hat{a_2}, Q_i(1)\hat{a_3}, Q_i(2)\hat{a_1}, Q_i(2)\hat{a_2}, ..., Q_i(n)\hat{a_3})$, and B_i be the similar expression involving $\hat{b_1}, \hat{b_2}, \hat{b_3}$. Let $\hat{M}$ be the matrix whos rows are $A_1, ..., A_m, B_1, ..., B_m$. and let C be the matrix whos columns are $\left(u_1^{(t)}, v_1^{(t)}, Z_1^{(t)}, \ldots, u_n^{(t)}, v_n^{(t)}, Z_n^{(t)}\right)^\top$ for $1 \leq t \leq T$.

Note that $\hat{m}$ is point-dependent, but does not vary with time, whereas 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:

$$\hat{U} = \begin{bmatrix} \hat{u}_1^{(1)} & \dots & \hat{u}_m^{(1)} \\ \vdots & \ddots & \vdots \\ \hat{u}_1^{(T)} & \dots & \hat{u}_m^{(T)} \end{bmatrix}, \hat{V} = \begin{bmatrix} \hat{v}_1^{(1)} & \dots & \hat{v}_m^{(1)} \\ \vdots & \ddots & \vdots \\ \hat{v}_1^{(T)} & \dots & \hat{v}_m^{(T)} \end{bmatrix}$$

Stacking all equations (2) we get:

$$\begin{pmatrix} \hat{U}^\top \\ \hat{V}^\top \end{pmatrix}_{2m \times T} = \hat{M}_{2m \times 3n} C_{3n \times T} \tag{3}$$

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

$$\begin{pmatrix} U^\top \\ V^\top \end{pmatrix}_{2n \times T} = M_{2n \times 3n} C_{3n \times T} \tag{4}$$

3 Synchronization and Action Indexing

Given two sequences of a dynamic scene in the above setting, they are first synchronized. Having synchronization, we later describe how to compute the structure of the scene across time. The ability to synchronize two sequences of the same action can be used for other dynamic-scene applications such as action indexing. Two matching actions should have a minimal synchronization error.

Examine the matrix we get 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 connection between the sequences, i.e when they are not synchronized. Combining equations (3) and (4) we get:

$$E^\top = \begin{pmatrix} M \\ \hat{M} \end{pmatrix}_{(2m+2m) \times 3n} C_{3n \times T} \tag{5}$$

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

These rank upper bounds are 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 drops in the unsynchronized case.

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

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

The first method, instead of bounding the rank of E by $3n$, uses a heuristic to define $\hat{n}$, the *effective rank* of (U, V). We compute the singular values $s_1, ..., s_{h_1}$ of (U, V), and set $N = argmin_j \{\sum_{k=1}^{2j} s_k > J\}$ for some threshold J (we used $J = 0.99 \sum_{k=1}^{h_1} s_k$). This is equivalent to finding the minimal rank of a matrix in a given error bound under Frobenius norm.

In order to synchronize we select 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 $e_1, ..., e_{h_2}$ be all the singular values of E. Then g is defined by:

$$g(E) = \sum_{k=3\hat{n}+1}^{h_2} e_k$$

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, and otherwise, if it is broken by non-synchronized data, this measure should be high.

The second method is based on the idea that for synchronized sequences the linear subspaces spanned by the columns of (U, V) and by the columns 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 [7]. 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 as:

$$cos(\theta_k) = \max_{\mathbf{u} \in U_A} \max_{\mathbf{v} \in U_B} \mathbf{u}^\top \mathbf{v}$$

subject to:

$$\mathbf{u}^\top \mathbf{u} = \mathbf{v}^\top \mathbf{v} = 1, \quad \mathbf{u}^\top \mathbf{u}_i = 0, \mathbf{v}^\top \mathbf{v}_i = 0, \qquad i = 1, ..., k-1$$

A convenient way to compute the principal angles is via the QR factorization, as follows. Let $A = Q_A R_A$ and $B = Q_B R_B$ where Q is an orthonormal basis of the respective subspace and R is a 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 $Q_A^\top Q_B$ are the principal angles $cos(\theta_i) = \sigma_i$.

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.

3.1 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 [8] 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 just the image derivatives I_x and I_y of the reference image (t=0) and the derivative over time $I_t^j = I(j) - I(0)$. The relation

- “the generalized Lukas & Kanade constraint” - is expressed by the following equation:

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

Where A,B and C are diagonal $n \times n$ matrices with the diagonal elements being $a_i = \sum_k (I_x(k))^2$, $b_i = \sum_k (I_x(k)I_y(k))$ and $c_i = \sum_k (I_y(k))^2$ respectively, where the summation is at a small window around pixel i. The (i,j)th element of G and H are given by $g_ij = -\sum_k (I_x(k)I_t^i(k)$ and $h_ij = -\sum_k (I_y(k)I_t^i(k)$ where the summation is over a small neighborhood around pixie j.

In practice we compute matrices G and H only for those pixels for which there is enough gradient information in their neighborhood, hence the matrix $\begin{bmatrix} A,B \\ B,C \end{bmatrix}$ is invertible and the column space of U and V is the same as the column space of G and H respectively. A similar constraint connects the flow matrices in the second sequence U' and V' to measurement matrices G' and H' computed on 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 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 $3N$.
- The column spaces of the matrix $[G,H]$ and of the matrix $[G'.H']$ intersect in a linear subspace of rank N.

These observations can lead to synchronization methods analogous to those presented of the flow matrices. A care should be taken as for the number of frames used. The above “generalized Lukas and Kanade constraint” assumes infitisimal motion. This assumption might hold only a few frames a way from the reference frame. In practice we divide both sequences to small continuous fragments (small windows in time) of length up to 17 frames. We pick the reference frame as the middle frame of each fragment, 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.

Let k, k' be the number of fragments we extract from each sequence. 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 average value . The offset of the line determines the shift of the synchronization and the slope of it 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 $D = null(\begin{bmatrix} U^\top \\ V^\top \end{bmatrix})$ be a matrix whose columns are all orthogonal to the rows of $\begin{bmatrix} U^\top \\ V^\top \end{bmatrix}$ [1]. The rows of $\begin{bmatrix} U^\top \\ V^\top \end{bmatrix}$ are the rows of C which correspond to motion along the X and Y axes. Hence the first two out of every 3 rows in CD vanish (all which is left is the information regarding the Z coordinates of the displacements).

By Multiplying both sides of Eqn. (3) with D we get:

$$K := \begin{pmatrix} \hat{U}^\top \\ \hat{V}^\top \end{pmatrix} D = \hat{M}CD = \begin{pmatrix} \hat{a}_3 Q_1^\top \\ \hat{b}_3 Q_1^\top \\ \vdots \\ \hat{a}_3 Q_m^\top \\ \hat{b}_3 Q_m^\top \end{pmatrix} \begin{pmatrix} Z_1^{(1)} & \dots & Z_1^{(T)} \\ \vdots & \ddots & \vdots \\ Z_n^{(1)} & \dots & Z_n^{(T)} \end{pmatrix} D$$

Observe that each odd row of K equals the next row of K multiplied by the ratio $r = a(3)/b(3)$. Hence this ratio can be recovered by dividing the rows. For added robustness we take the median of all the measurements of r from all points. In Section. 4.1 we elaborate on the robustness of the algorithm.

Consider the vector $l = (\,1 \quad -r\,)^\top$. 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.

[1]We assume $T > 2N$, otherwise synchronization is not possible. If $T < 2n$, we take D to be the vector corresponding to the smallest singular value.

We multiply both sides of Eqn. 1 with $l^\top$ and get:

$$l^\top \begin{pmatrix} \hat{u_i^{(t)}} \\ \hat{v_i^{(t)}} \end{pmatrix} = (\, c_1 \quad c_2 \,) \begin{pmatrix} u_1^{(t)} & u_2^{(t)} & \dots & u_n^{(t)} \\ v_1^{(t)} & v_2^{(t)} & \dots & v_n^{(t)} \end{pmatrix} Q_i \tag{6}$$

Where $c_1 = l^\top(\hat{a_1}, \hat{b_1})$,$c_2 = l^\top(\hat{a_2}, \hat{b_2})$, and the third row of the projection matrix just vanished ($l^\top(\hat{a_3}, \hat{b_3}) = 0$).

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 linearization, 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.

From u_i recovering Q_i up to scale is is done by factoring the elements of u_i as a $2 \times n$ rank 1 matrix [2] and the scale is adjusted such that $\sum_i Q_i = 1$.

Once the Q_i are recovered we can recover the points in the first image corresponding to the points tracked in the second image:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \hat{P}_i^{(t)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} (\, P_1^{(t)} \quad P_2^{(t)} \quad \dots \quad P_n^{(t)} \,) Q_i =$$

$$\begin{pmatrix} x_1^{(t)} & x_2^{(t)} & \dots & x_n^{(t)} \\ y_1^{(t)} & y_2^{(t)} & \dots & y_n^{(t)} \end{pmatrix} Q_i$$

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 $Q_i, i = 1...m$.

4.1 Outlier Rejection

The basis of our algorithm and many similar feature based-algorithms is the assumption that the input points fit some model and were tracked correctly along the sequence. Outlier points, e.g. points which were not tracked properly, may have in some cases a strong effect on the accuracy of the results. In this section we show that outlier points tracked in the first camera have negligible

[2]It is better to first recover c_1, c_2 using all of u_i, since they are not point-dependent.

effect on the results, and 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, this subspace may be reduced, but still the resulting subspace is orthogonal to all inlier trajectories. Therefore the computation of $r = \hat{a_3}/\hat{b_3}$ 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 inliers 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 show next 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, e.g. by the measuring variance of this value for each point and across the points can be used to reject outliers.

In our experiments usually $2n > T > 3\hat{n}$, i.e. the matrix $\hat{B}$ had rank T due to noise, but the effective rank $2N$ was smaller. For numerical reasons we chose D to be the vector corresponding to the smallest singular value, and therefore every point in the second image had a single measurement of the ratio r. Outlier points were chosen by computing the median of the r values of all points, and discarding points with r value far from the median.

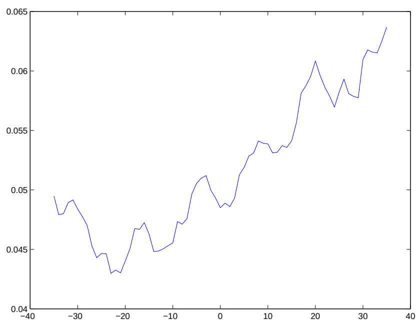
5 Experiments

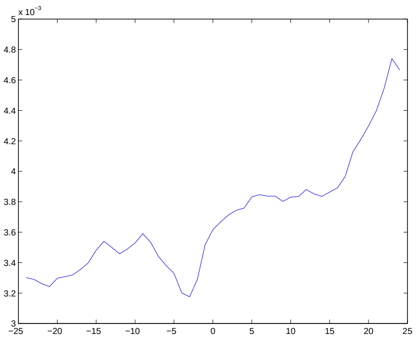
We have tested our algorithm on scenes of moving people. The first set of experiments tested the camera synchronization application. Two video sequences were captured by two unsynchronized cameras viewing the same non-rigid scene. In the first experiment we used tracked points, and examined the algebraic 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 have used the direct method described above. Since this method assumes small image motion, we applied it to small temporal windows $w(t)$ (typically of 15 consecutive frames), and smoothed and decimated the images. For each such pair of windows $w(t_1), w(t_2)$ from the two sequences resp., we measured the smallest principle angles between the subspaces defined on these windows (Lior, the explanation here depends on how you explain before). A typical result of this process is shown in Fig. 2-a. The matrix contains for each pair of times t_1, t_2 the principle angles between the linear spaces computed for $w(t_1)$ on the 1st sequence and $w(t_2)$ on the 2nd sequence. Since both t_1, t_2 increment by the same shift, the optimal synchronization is visualized by a 45^o dark line in the image. The mean value along such lines is shown in Fig. 2-b. Corresponding frames of the two synchronized sequences are shown in Fig. 2-c,d,e,f. We have confirmed these results by manually synchronizing the input sequences.

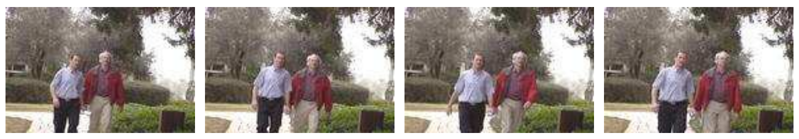
The second experiment tested the reprojection stage. Using the proposed algorithm we established correspondences between the two cameras, by transferring tracked points from one sequence to the other sequence. The results are presented in Fig. 3. Unfortunately, more than 50% of the tracks were not very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

5.1 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 on 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. In [5] an affinity matrix was defined in a similar manner. As pointed out in [9], once the affinity matrix is defined, the choice of the clustering criteria is arbitrary. We have used a software package [1] including several clustering algorithms, and chose the Ward method which brought 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.

a) 

b) 

c) 

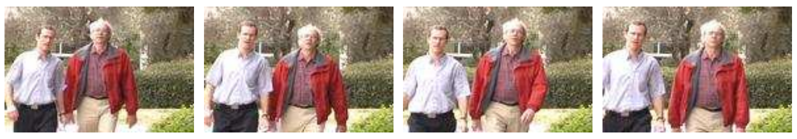
d) 

Figure 1. Sequences synchronization for cameras with different zoom factors (Fig. 1-a) and for cameras with large vergence (Fig. 1-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. frame offset.

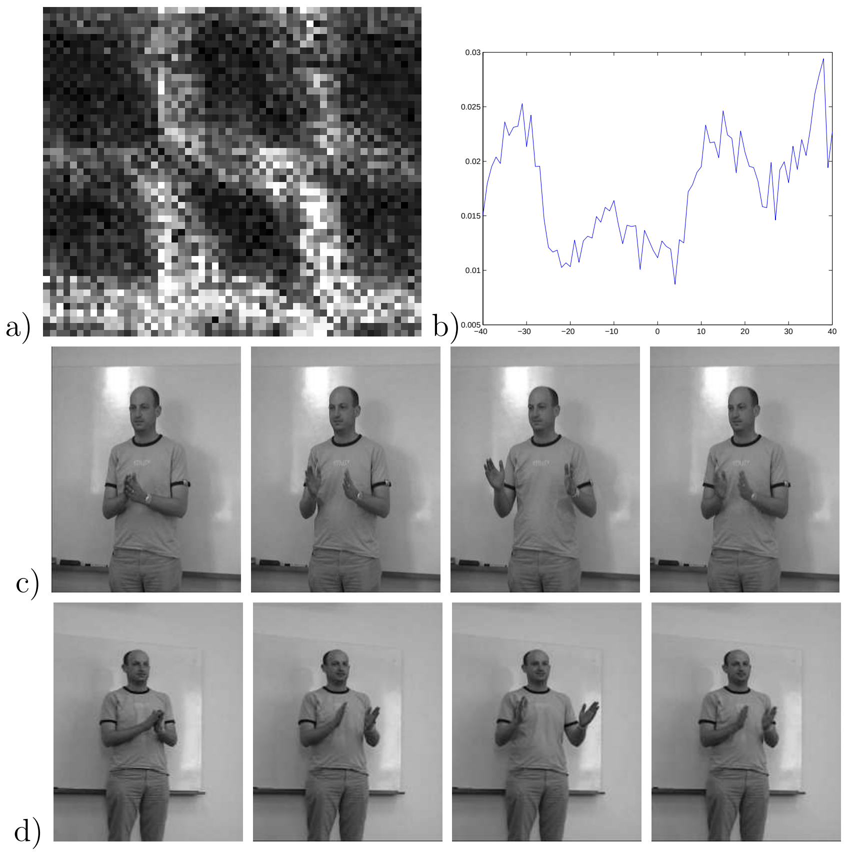


Figure 2. Sequences synchronization for cameras with large vergence using direct method. Panel (a) shows the distance (minimal principle angle, see text) between temporal windows at different temporal shifts in the two sequences. By averaging diagonal directions in this matrix, one achieves a score for different synchronizations, as shown in panel (b). The cyclic nature of the motion can be viewed in the graph. Corresponding frames from the sequences of the first graph are shown in (c) and (d).

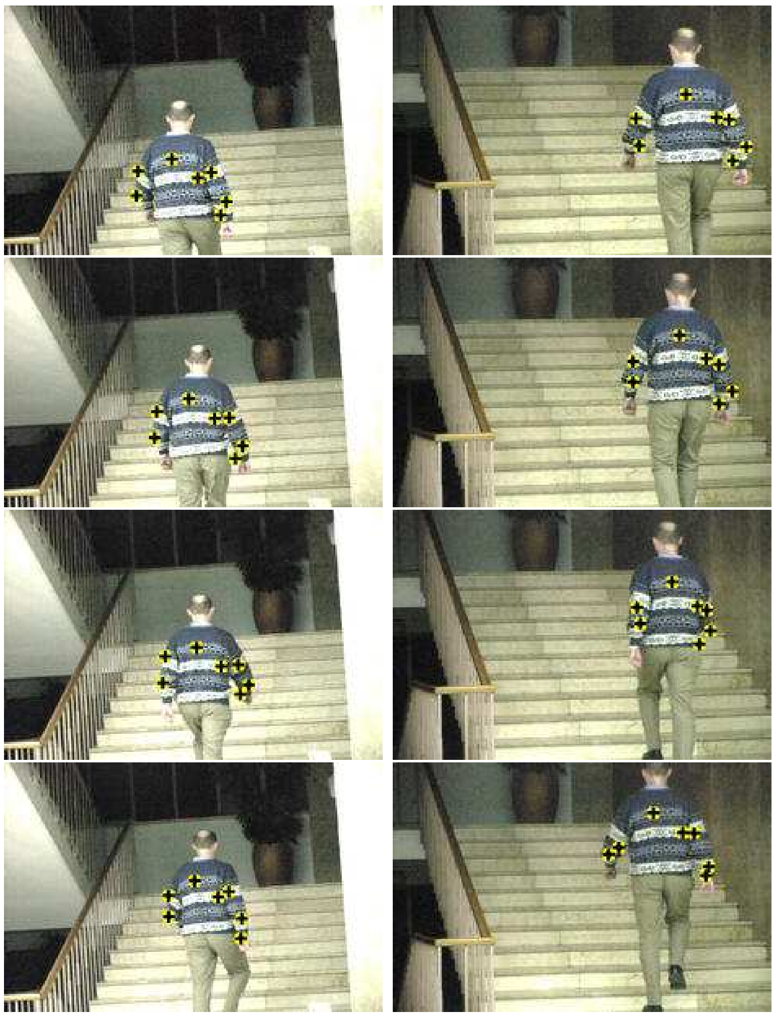


Figure 3. Feature reprojection between two sequences of a non-rigid body.

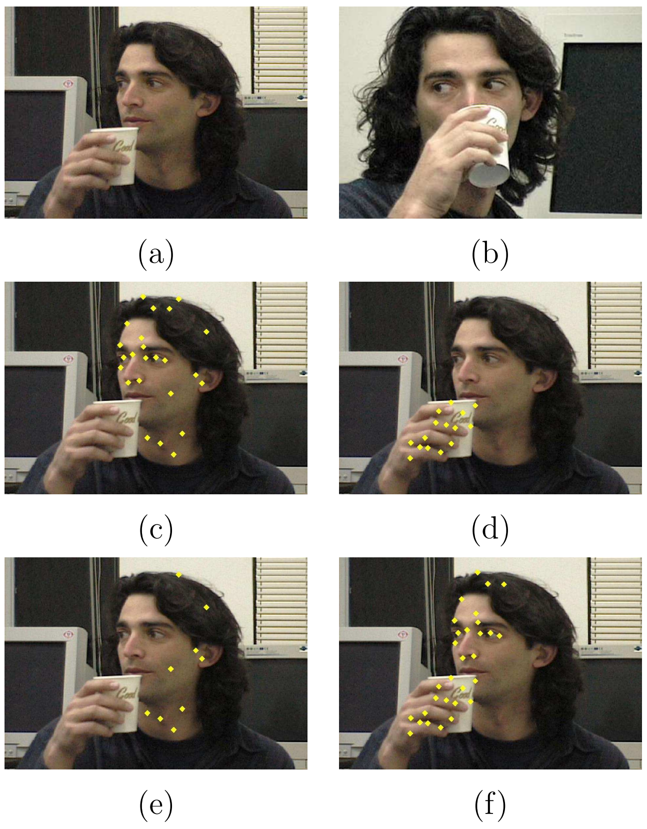


Figure 4. Feature clustering for k-body scene, a comparison with the Costeira-Kanade algorithm. Fig. 4-(a),(b) show examples of input images taken by the two cameras at different times. Note the large vergence between the cameras. Fig. 4-(c),(d) show the two clusters found by the method in this paper.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

6 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 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 then half a second long. A dictionary containing many such fragments and their associated actions will be constructed. In order to recognize an action we will compare each such fragment to the fragments in the dictionary, and try to inference the whole action.

References

[1] Available at http://odur.let.rug.nl/ kleiweg/clustering/clustering.html.

[2] M. Brand. Morphable 3d models from video. In *Computer Vision and Pattern Recognition*, Kauai, Hawaii, pages II:456–463, 2001.

[3] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *Computer Vision and Pattern Recognition*, Hilton Head, SC, pages II:690–696, 2000.

[4] Y. Caspi and M. Irani. Alignment of non-overlapping sequences. In *International Conference on Computer Vision*, Vancouver, BC, pages II: 76–83, 2001.

[5] J. Costeira and T. Kanade. A multibody factorization method for independently moving-objects. *Int. J. of Computer Vision*, 29(3):159–179, Sept. 1998.

[6] F. Dornaika and R. Chung. Stereo correspondence from motion correspondence. In *Computer Vision and Pattern Recognition*, pages I:70–75, 1999.

[7] G. Golub and C.V. Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[8] M. Irani, Multi-Frame Optical Flow Estimation Using Subspace Constraints. In *IEEE International Conference on Computer Vision (ICCV)*, Corfu, September 1999.

[9] K. Kanatani. Motion segmentation by subspace separation and model selection. In *International Conference on Computer Vision*, Vancouver, BC, pages II: 586–591, 2001.

[10] M. Leventon and W. Freeman. Bayesian estimation of 3d human motion. technical report tr 98-06, mitsubishi electric research labs, 1998.

[11] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *International Conference on Computer Vision*, Vancouver, BC, pages II: 636–643, 2001.

[12] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *European Conference on Computer Vision*, pages II:709–720, 1996.

[13] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *Int. J. of Computer Vision*, 9(2):137–154, November 1992.

[14] T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *British Machine Vision Conference*, 2000.

[15] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):733–742, July 1997.

[16] R. Vidal and R. Hartley. Motion Segmentation with Missing Data using PowerFactorization and GPCA. In *Computer Vision and Pattern Recognition*, 2004.

[17] L. Zelnik-Manor and M. Irani. Degeneracies, Dependencies and their Implications in Multi-body and Multi-Sequence Factorizations In *Computer Vision and Pattern Recognition*, June 2003

[18] L. Wolf and A. Zomet. Sequence to sequence self calibration In *European Conference on Computer Vision*, Copenhagen, Denmark, pages II: 370-377, 2002.