Feature-Based Multi-Video Synchronization with Subframe Accuracy

A. Elhayek, C. Stoll, K. I. Kim, H.-P. Seidel, C. Theobalt

MPI Informatik
{elhayek, stoll, kkim, hpseidel, theobalt}@mpi-inf.mpg.de

Abstract. We present a novel algorithm for temporally synchronizing multiple videos capturing the same dynamic scene. Our algorithm relies on general image features and it does not require explicitly tracking any specific object, making it applicable to general scenes with complex motion. This is facilitated by our new trajectory filtering and matching schemes that correctly identifies matching pairs of trajectories (inliers) from a large set of potential candidate matches, of which many are outliers. We find globally optimal synchronization parameters by using a stable RANSAC-based optimization approach. For multi-video synchronization, the algorithm identifies an informative subset of video pairs which prevents the RANSAC algorithm from being biased by outliers. Experiments on two-camera and multi-camera synchronization demonstrate the performance of our algorithm.

1 Introduction

The last ten years have observed significant advances in mobile camera technology. The widespread use of smart-phones facilitated casually capturing and sharing any scenes of interest. The abundance of these data resulted in new opportunities and challenges in computer vision and computer graphics. For instance, there are more chances than ever to capture the same scene with multiple cameras: e.g., street performance captured by several spectators. This can significantly broaden the domain of multiple-camera computer vision and graphics applications (e.g., markerless motion capture and video-based rendering [1]). However, it should be noted that these algorithms typically assume that the cameras are synchronized, i.e., the ratio between the frame rates and the relative offsets are known. In general uncontrolled settings, this may not be true: the camera hardwares maybe heterogeneous and accordingly the recorded sequences (videos) have different frame rates. Sometimes, we only have the sequences with unknown source cameras. Furthermore, it is unlikely that the recorded sequences have the same offset. Accordingly, automatic synchronization is required.

There exist several synchronization algorithms. However, these algorithms are limited to specific scenes where it is possible to track the objects of interest, or to scenes where the objects show specific motions such as ballistic motion [2], or to synchronizing two sequences only.

As a step towards the general case of uncontrolled video synchronization problem, we propose a multi-video synchronization algorithm which works for objects exhibiting any type of motion. In particular, we do not assume that the objects of interest are explicitly tracked. This is facilitated by feature-based matching: we extract a set of features and track them in each video, which constitute a set of feature trajectories. Then, the problem of synchronization is cast into spatio-temporally matching the trajectories across different sequences. Since such general features usually exist in any video, our algorithm is applicable to general scenes with any number of objects therein. Moreover, the dynamic properties of these trajectories enable the algorithm to achieve sub-frame accuracy of the synchronization parameters.

The technical challenges lie in the fact that the tracked trajectories are in general very noisy, *e.g.*, the tracked location of detected feature points are not precisely aligned in a video and tracking could fail. Furthermore, since there can be many trajectories in a given set of videos, identifying correctly matching pairs of trajectories across different videos is challenging. One of our main contributions is a method for resolving these problems. We propose a set of criteria to filter out noisy and uninformative trajectories and pairs of trajectories (details will be discussed in Sec. 3). As a result, a set of tentative trajectory pairs are generated. Among them, the correct subset (inliers) is identified by minimizing a global energy based on RANSAC-type optimization. Since the energy is defined for any number of sequences, our algorithm can be consistently applied to the multi-video case as well as to the two-video case. However, in this case, additional robustness is achieved by identifying weakly coupled pairs of cameras and removing them from the evaluation of energy. This leads to an automatic generation of a graph representing the cameras and their connectivity. In the experiments, we demonstrate the effectiveness of our algorithm with datasets that are difficult to synchronize with the existing object tracking based synchronization techniques.

Related work. One of the first video synchronization algorithms is described in [3] where the algorithm detects static features and tracks moving objects. Based on these detected and tracked features, it estimates the planar alignment as well as the epipolar geometry. This algorithm permits for synchronizing videos which show significantly different view points. However, its usage is limited by the fact that it requires explicitly tracking objects and is applicable only to a pair of videos. One or both of these limitations are shared by most existing algorithms. For instance, the algorithms of Dai et al.[4] and Caspi et al. [5] are designed specifically for the two-video case. On the other hand, Sinha and Pollefeys' silhouettes-based algorithm [6] and Meyer et al.'s algorithm for moving cameras [7] can synchronize multiple cameras, which are based on explicit feature tracking or on the (often violated) assumption of the existence and detection of reliable (long and clean) trajectories.

Most strongly related to the proposed algorithm is [5], where the concept of feature trajectory matching was introduced for video synchronization. Our algorithm extends this algorithm and explicitly overcomes the two main limitations of [5]: 1) our algorithm is applicable when there is arbitrary time shift and frame rate differences, 2) our algorithm enables multi-camera synchronization. Neither of this is directly feasible using Caspi et al.'s algorithm [5] since they use grid search of parameters, which is applicable when only one or few parameters need to be estimated. An alternative to video-based synchronization is to exploit additional data, e.g., audio [8] or still images obtained with controlled flashes [9].

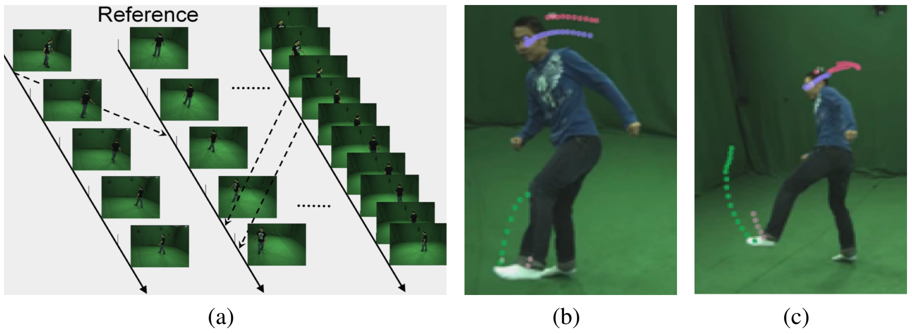

Fig. 1. (a) Schematic diagram of multi-video synchronization. Time lines of different sequences with different frame rates are mapped to non-integer points along a single reference time line, as indicated by the arrows. **(b)** and **(c)** show two temporally corresponding frames from two different video sequences with some corresponding space-time trajectories resulting from the actor's motion in the previous frames.

2 Problem Formulation

Similar to other synchronization methods [5, 10], we assume that each video is recorded by a camera which has a constant frame rate. In this case, the temporal misalignment between a set of videos occurs if they have a time-shifts (offsets) between their start times, and/or when they have different frame rates (Fig. 1(a)). Accordingly, there is an affine relationship between the time lines (time coordinate values) of each pair of sequences.

For the two-video case, synchronization can be performed by firstly setting one sequence as a reference (denoted as S_r) and estimating the relative offset θ_i and the frame rate R_i of the other sequence (denoted as S_i) with respect to the reference time line t_r of S_r:

$$t_r = R_i * t_i + \theta_i, \tag{1}$$

where t_i is the time line of S_i. For general multi-video synchronization, consistent comparison of multiple sequences can be facilitated by establishing a global reference time line. While any global parametrization should work, for a given set of input unsynchronized sequences $\mathcal{S} = \{S_0, \ldots, S_N\}$, we simply set the time line of the first sequence S_0 as the reference. This sequence and the corresponding time line will henceforth be denoted as S_r and t_r, respectively. With this representation, our algorithm produces an estimate of synchronization parameters $\{\theta_i, R_i\}$ (with respect to t_r) for each sequence $S_i \in \mathcal{S} \setminus S_r$.

Since the sequences in $\mathcal{S}$ capture the same scene, there is a geometrical relationship between the appearances of the scene components in each pair of sequences: Let $\mathbf{x}_r = (x_r, y_r, t_r)$ be a space-time point in the reference sequence S_r and $\mathbf{x}_i = (x_i, y_i, t_i)$ be the corresponding points in $S_i \in \mathcal{S} \setminus S_r$ (i.e., t_r and t_i are related based on Eq. 1). Then, they should satisfy the below given fundamental geometrical relationship:

$$p_r(t_r)^\top F_i(t_i) p_i(t_i) = 0 \tag{2}$$

where, $p_r(t_r)$ is a vector consisting of the spatial coordinate values (i.e., $\{x_r, y_r, 1\}$) of $\mathbf{x}_r$ and F_i is the fundamental matrix relating the reference camera and the i-th camera.[1]

3 General synchronization algorithm

This section presents our synchronization algorithm. We first discuss the two-video synchronization setting and illustrate the essential idea. Then, we discuss how this framework can be applied to multi-video data sets.

3.1 Two-video synchronization

Our algorithm is based on matching trajectories of features appearing in a pair of videos. First, a set of features (SIFT features) are extracted from each frame of a sequence. Then, we use Best-Bin-First (BBF)-based feature matching to establish correspondences between features appearing in each pair of consecutive frames; see [11] for details. If the features corresponding to a single 3D-point are matched across more than two consecutive frames, the corresponding trajectory is constructed. Each trajectory is represented based on spatial coordinates of the corresponding feature points, each of which is assigned with the corresponding frame index. For instance, a trajectory in S_r can be represented as

$$T_r = \{p_r(t), p_r(t+1), p_r(t+2), ..., p_r(t+k)\},$$

where $k+1$ is the length of the trajectory (i.e., tracking is successful for $k+1$ consecutive frames).

Matching a pair of trajectories implies establishing the correspondence between two sets of points contained in the two trajectories, respectively. Precisely matching a pair of *non-trivial* trajectories (details will be discussed shortly), uniquely defines the spatial parameters (i.e. fundamental matrix; cf. Eq. 2), and since each point is assigned with the time index, the corresponding temporal parameters (offset and frame rate ratio).

In general, the construction of trajectories is noisy. For example, usually the locations of detected features do not precisely correspond to each other across the consecutive frames and the tracking can be erroneous. Accordingly, the constraint (2) might not be exactly satisfied. Alternatively, one could minimize the following residual error with respect to those parameters [5]:

$$E(F_i, \theta_i, R_i) = \sum_{t_i \in support(T_i)} d_{F_i}\left(p_r(R_i \cdot t_i + \theta_i), p_i(t_i)\right), \tag{3}$$

where $d_F(A, B)$ is the Euclidean distance between a feature A and the epipolar line corresponding to a feature B mapped based on F (see Fig. 2(c)).

The above-described strategy is applicable only when a correct pair of trajectories (each from a single sequence) is identified. In general, there are multiple trajectories

[1] Throughout the current paper, we assume that the cameras are static. For the general moving camera case, F has to be defined for each pair of corresponding frames as in [8].

constructed in each sequence and the correspondences between them are not known *a priori*. Suppose that m and n trajectories are constructed from S_r and S_i, respectively. Then there are $m \times n$ potential matching pairs of trajectories, only a few of which are correct. Our approach is to use RANSAC which can effectively filter out the outliers matches. However, naively feeding all potential matches into a RANSAC step does not yield a proper parameters estimate: there exist several *trivial* trajectories which geometrically match many other trajectories. Moreover, the large number of trivial trajectories decreases the computational efficiency of the method. Therefore, we introduce three trajectory filtering steps. Firstly, we remove very short trajectories which are shorter than a specific number of frames (5 frames in our experiments). The second filter removes trajectories corresponding to static feature points: a trajectory is removed if the variance of its spatial coordinate values is small (i.e. less than 15 pixels in our experiments). Finally, we remove all trajectories which may generate ambiguous matches. This happens when the tangents of trajector points are nearly parallel to the points epipolar line defined by the fundamental matrix of the camera pair. We cannot distinguish any motion along that line in the other camera. This may lead to the feature match being classified as an inlier with low energy even for wrong matches. To find these trajectories, for each point in the trajectory, we check the angles between the tangent and the epipolar line of the point in its own camera. If the sum of these angles is too small, the corresponding trajectory may erroneously match many trajectories in the other sequence. We reject a trajectory if the score $\sum_{t_i \in sup(T_i)} 1 - cos(angle)$ is less than 0.32 (see supplementary material for more details).

Even after the trajectory filtering stage, eroneous candidate trajectory pairs may remain. These may negatively influence the run-time of a RANSAC optimization, and for a prescribed finite run-time, can bias RANSAC toward a unreliable solution. It should be noted that in order for a pair of trajectories to match, they have to overlap with each other in space and in time. Checking this can quickly filter out most wrong matches: Given a candidate match, we intersect the epipolar line corresponding to each feature point in the shorter trajectory with the longer trajectory (Fig. 2(a)). Since the frame rates of corresponding source videos are fixed, the consecutive epipolar lines should intersect with the longer trajectory such that the points of intersection are roughly equally spaced.[2] To check this, we first calculate the hypothetical frame rate ratios of two videos (denoted as R_T) based on the entire interval of intersection. For instance, in Fig. 2(a), R_T is calculated by dividing the number of feature points lying between $F_i \cdot p_1$ and $F_i \cdot p_7$ on the longer trajectory with 7 which is the number of intersecting epipolar lines. In the same way, we calculate hypothetical frame rate ratios from each consecutive interval on the trajectory (e.g., $[F_i p_1, F_i p_2]$). All of these estimated frame rate ratios should agree roughly with R_T: we decide that a new hypothetical frame rate ratio R_N agrees with R_T if $|R_T - R_N| < 0.5 R_T$.

Then, the degree of overlap between two trajectories is measured based on the number of consecutive epipolar lines (P_{min}) which satisfies the above described condition. When, P_{min} is smaller than 5 (threshold found by experimental validation), the corresponding trajectory pair is rejected. It should be noted that in general, an epipolar line

[2] Note that in case of corresponding trajectories from identical cameras (i.e. equal frame rates) the distances between consecutive points of intersection along the time dimension must be 1.

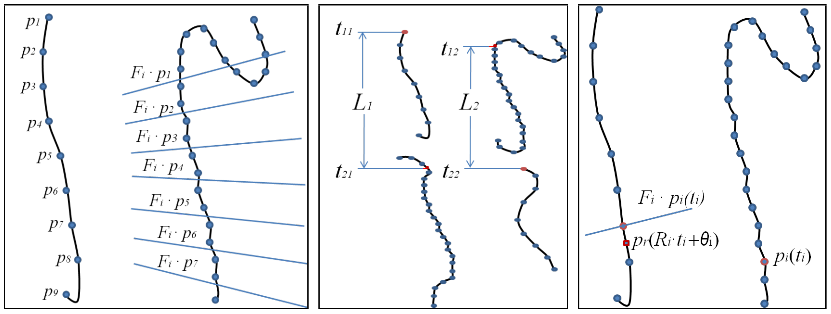

Fig. 2. (a) Left: Epipolar trajectory matching test. **(b) Center:** Estimation of synchronization parameters based on two distant pairs of matching trajectories. **(c) Right:** Trajectory point residual error measure as the distance between the point $p_r(R_i \cdot t_i + \theta_i)$ and the epipolar line $F_i \cdot p_i(t_i)$.

can intersect with a trajectory more than once (Fig. 2(a)). This case can be dealt with by retaining multiple hypothetical framerate ratios (R_T) accordingly. The result of this step is a table of tentative matching trajectories.

The extension of the energy functional (3) for multiple trajectory case, given a precomputed fundamental matrix F_i for static cameras, is as folows:

$$E(\theta_i, R_i) = \sum_{T_i \in \Gamma_i} \sum_{t_i \in support(T_i)} d_{F_i}\left(p_r(R_i \cdot t_i + \theta_i), p_i(t_i)\right), \qquad (4)$$

where Γ_i is the set of trajectories for the i-th video, minimizing it with the tentative matches does not correctly estimate the synchronization parameters since the tentative matches still contain a lot of outliers. Accordingly, we apply the RANSAC algorithm instead. It should be noted that each iteration of RANSAC requires generating hypothetical synchronization parameters. This can be determined from two pairs of corresponding feature points. These can be sampled from a single pair of matching trajectories, but we select them from two distinct candidate matches. This turned out to be more robust; see Fig. 2(b). Then, the hypothetical parameters are computed by solving the following equations for the two unknowns:

$$t_{11} = R_i * t_{12} + \theta_i,$$
$$t_{21} = R_i * t_{22} + \theta_i$$

and the corresponding residual error is used to classify the tentative matches into inliers and outliers; see Fig. 2(c). The number of iterations of RANSAC adaptively changes based on the number of inliers [12]. At the end of the RANSAC loop, the parameters with the highest number of inliers are selected. The estimated parameters are further refined by continuously optimizing (4) with only inliers: We first render the problem into continuous optimization by interpolating each trajectory with cubic-splines. Then a standard gradient descent is performed. However, our preliminary experiments revealed that the continuous optimization step does not significantly improve the result over the initial RANSAC estimate. Further detail of the algorithm can be found in the supplementary material.

3.2 Multi-video synchronization

Once the global time coordinate is established, the extension of two-video synchronization framework to multi-video case is straightforward. In this case, the global energy functional can be defined as the sum of pair-wise energies (4) for any possible pairs. However, naively optimizing this energy functional is sub-optimal: some pairs of videos have more matching candidates and, accordingly, they are more informative than the other pairs. For instance, for two videos showing the same scene but with from significantly different viewpoints, the number of candidate trajectory matches might be very small. In this case, the parameters estimated by emphasizing the error corresponding to this camera pair might not be reliable. The remainder of this section discusses a strategy for solving this problem.

The relationships between a set of videos (or cameras) can be represented as a graph (see Fig. 3) in which a node corresponds to a sequence and an edge represents a set of tentative matching pairs of trajectories plus the corresponding synchronization parameters (of one node, with the other node treated as a reference). In this case, there are as many sets of parameters as the number of edges (i.e. local edge parameters), while the actual number of sets of parameters should correspond to the number of nodes (i.e. global parameters related to reference time line).

To ensure that a consistent global parametrization can be recovered from a set of local edge parameterizations, in each RANSAC step, we remove any cycle in the graph. This can be done, in principle, by randomly building a spanning tree. However, we have empirically observed that the accuracy of the estimated synchronization parameters between a pair of videos decreases with increasing distance between the cameras. Specifically, the lower the number of tentative matches between a pair of sequences, the less accurate the resulting estimation of synchronization parameters becomes. We exploit this observation by pre-filtering edges between distant pairs of cameras based on the number of tentative matches (35 in our experiments). An example of the resulting *connectivity graph* is shown in Fig. 3.

Figure 3 exemplifies a single step of RANSAC iteration. The global parameters R_2 and θ_2 (with respect to the reference sequence S_0) can be estimated based on the edges e_{21}, e_{10} and e_{20}. In general, the pairwise estimates of local parameters for each of these edges conflict with each other. To rule this out, in the RANSAC step, we construct a random spanning-tree, e.g., by removing edges E_{20} and E_{10}.

The estimated local edge parameters are converted to the global parameters using the relations

$$R_{xy} = \frac{R_x}{R_y}, \quad and \quad \theta_{xy} = \frac{\theta_x - \theta_y}{R_y}, \tag{5}$$

where R_{xy} and θ_{xy} are the parameters of the edge between any two nodes x and y. Once the global synchronization parameters are constructed, they are evaluated based on the number of inliers using every edge in the graph, i.e, the trajectory pairs which are not contained in the spanning tree are used as well. After the RANSAC iteration, the set of global parameters corresponding to the highest number of inliers is selected.

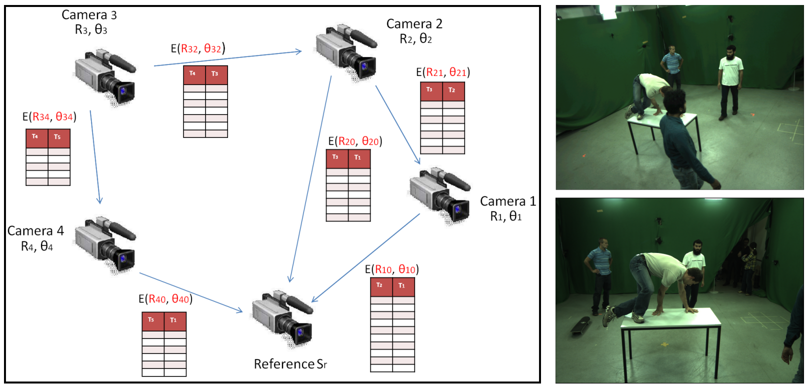

Fig. 3. (a) Left: An example of multi-video connectivity graph constructed by our algorithm. **(b) Right:** An example frames (1296 x 968) from the videos S_r^2 and S_1^2 (250 and 300 frames respectively).

4 Experimental Evaluation

In this section, we evaluate our algorithm based on two sets of unsynchronized videos capturing different scenes with different number of moving persons. To facilitate quantitative evaluation, we set the cameras up such that accurate timestamps for each frame can be obtained, which provide the corresponding ground-truth synchronization parameters for each set of videos. Once features are extracted, our algorithm took on average 20 seconds and 3.5 minutes for two-video and four-video synchronization, respectively.

In our evaluation, we show the *residual error* of the parameters as well as the average and maximum *frame errors*, that are computed by aligning each frame of the synchronized video to the reference time line and by computing the deviations of the corresponding frame numbers from the ground-truth.

In the first set of experiments, we evaluated the performance of our algorithm for the two-video case. To gain an insight into the role of individual filtering steps (Sec. 3.1), we constructed two different versions of our algorithm - one of them is constructed by removing the static filtering, the other one by removing the epipolar filtering stage from the original algorithm. We have also performed experiments with known framerates, which are assumed to be known for most existing synchronization algorithms. Table 1 shows the results for a dataset consisting of two videos.[3] Additional results (with another set of videos) are provided in the accompanying supplementary material. The results suggest that both filtering stages, most notably the epipolar filtering, are critical to the performance of our algorithm; and that if once the frame rates are known, significant improvement can be gained.

Table 2 summarizes the result of multi-video synchronization experiments for two sets of videos, which show two different scenes containing four ($\mathcal{S}^1 = \{S_r^1, S_1^1, S_2^1, S_3^1\}$)

[3] The continuous optimization step improved the average error by only 0.01 from the RANSAC results with significant additional computation. Accordingly, for the rest of the experiments, we do not adopt this stage.

and three ($\mathcal{S}^2 = \{S_r^2, S_1^2, S_2^2\}$) video-sequences, respectively. This result demonstrates the effectiveness of our multi-video synchronization algorithm. The average frame error is less that two frames except for one pair of cameras: the average error for the sequence S_3^1 is rather high, which is most likely caused by the significantly different viewpoint from the rest of the videos in $\mathcal{S}^1$.

Our algorithm is capable of exploiting relationship among more than two video streams, and accordingly, it is naturally suited for multi-video applications. However, it should be noted that it is always possible to decompose a given multi-video synchronization problem into a set of two-video problems: one could first build a spanning tree and estimate the local pairwise synchronization parameters for each edge. Then, a globally consistent set of synchronization parameters can be estimated based on Eq. (5).[4] In general, the performance of multi-video synchronization should be better than this two-video synchronization-based approach, since the former can exploit all the available pairwise relationships, most of which are discarded when building a spanning tree. To exemplify this, we have selected three pairs of videos (namely $\{S_r^1, S_1^1\}$, $\{S_r^1, S_2^1\}$ and $\{S_r^1, S_3^1\}$), estimated pair-wise synchronization parameters, and obtained the global synchronization parameters based on (5). The performance of this algorithm is significantly worse than of our original multi-video synchronization algorithm: the average frame errors for S_1^1 , S_2^1 and S_3^1 were 0.753, 0.298 and 37.5, respectively. Especially, the two-video algorithm completely failed for S_3^1 since, as mentioned above, the camera's viewpoint is very different from the rest of the cameras; only one edge in the graph is not sufficient to compute reasonable estimate of the parameters.

In a final experiment, we evaluated the performance of a variant of our algorithm which determines the parameters based on grid search: each parameter is sampled at regular grid and the parameter set corresponding to the largest number of inlier is selected for multi-video synchronization. This can be regarded as an instantiation of Caspi el al.'s algorithm [5] in our feature-based setting. We found out that the grid search algorithm needs much longer computation times to yield results of comparable accuracy than our method because of the high dimensionality of the parameter space. For instance, for four videos in $\mathcal{S}^1$, to achieve a comparable runtime efficiency to our original algorithm, we had to choose very coarse grid spacings of more than 50 and 0.5 for θ_i and R_i, respectively (with reasonable search ranges of parameters $[-150, 150]$ and $[0.1, 2]$ for θ_i and R_i, respectively). The parameters S_1^1 , S_2^1 and S_3^1 optimized in this way are $-150/1.6$, $-150/0.1$ and $-50/1.1$, respectively, which are considerably worse than the results of our original algorithm.

[4] This corresponds to a single step of our multi-video RANSAC iteration.

Table 1. Two-video synchronization results. The ground truth parameters are $\theta_i = -50, R_i = 1$.

	Residual error in θ_i/R_i	Average frame error	Maximum frame error
Without static filtering	2.74 / 0.014	1.32	2.73
Without epipolar filtering	9.70 / 0.052	4.55	9.70
Complete algorithm	1.57 / 0.008	0.75	1.57
With given R_i	0.19 / 0.000	0.19	0.19

5 Conclusion and Future Work

We have presented a multi-video synchronization algorithm that succeeds on multi-video sets comprising two or more views of general scenes. It does not require tracking of a specific object but utilizes feature trajectories tracked in individual cameras that are matched across views. To enable this, we contributed a robust trajectory filtering and energy minimization framework based on RANSAC for the multi-camera case. In the future, we plan to extend our approach to moving cameras, in order to pave the way for handling general outdoor videos.

References

1. Ballan, L., Brostow, G.J., Puwein, J., Pollefeys, M.: Unstructured video-based rendering: interactive exploration of casually captured videos. In: ACM SIGGRAPH. (2010)
2. Wedge, D., Huynh, D., Kovesi, P.: Motion guided video sequence synchronization. In: ACCV. (2006)
3. Stein, G.P.: Tracking from multiple view points: Self-calibration of space and time. In: DARPA IU Workshop. (1998) 521–527
4. Dai, C., Zheng, Y., Li, X.: Subframe video synchronization via 3d phase correlation. In: Image Processing, 2006 IEEE International Conference on. (2006)
5. Caspi, Y., Simakov, D., Irani, M.: Feature-based sequence-to-sequence matching. Int. J. Comput. Vision **68** (2006) 53–64
6. Sinha, S.N., Pollefeys, M.: Synchronization and calibration of camera networks from silhouettes. In: ICPR. (2004)
7. Meyer, B., Stich, T., Pollefeys, M.: Subframe temporal alignment of non-stationary cameras. In: BMVC. (2008)
8. Hasler, N., Rosenhahn, B., Thormählen, T., Wand, M., Gall, J., Seidel, H.P.: Markerless motion capture with unsynchronized moving cameras. In: CVPR. (2009)
9. Shrestha, P., Weda, H., Barbieri, M., Sekulovski, D.: Synchronization of multiple video recordings based on still camera flashes. In: ACM Multimedia. (2006)
10. Pádua, F.L.C., Carceroni, R.L., Santos, G.A.M.R., Kutulakos, K.N.: Linear sequence-to-sequence alignment. IEEE Trans. Pattern Anal. Mach. Intell. **32** (2010) 304–320
11. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60** (2004) 91–110
12. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. 2 edn. Cambridge University Press (2004)

Table 2. Multi-video synchronization results.

Video	Ground truth (θ_i/R_i)	Estimated parameters	Average frame error
S_1^1	-50.00 / 1.000	-50.84 / 1.005	0.35
S_2^1	80.00 / 2.000	80.35 / 1.999	0.24
S_3^1	-30.00 / 1.000	-23.85 / 0.969	2.61
S_1^2	79.20 / 1.000	78.41 / 1.027	1.67
S_2^2	50.12 / 1.000	51.06 / 1.001	1.01