Through-the-Lens Synchronisation for Heterogeneous Camera Networks

Evren Imre
h.imre@surrey.ac.uk
Adrian Hilton
a.hilton@surrey.ac.uk

Centre for Vision, Speech and Signal Processing
University of Surrey
Guildford, UK

Abstract

Accurate camera synchronisation is indispensable for many video processing tasks, such as surveillance and 3D modelling. Video-based synchronisation facilitates the design and setup of networks with moving cameras or devices without an external synchronisation capability, such as low-cost web cameras, or Kinects. In this paper, we present an algorithm which can work with such heterogeneous networks. The algorithm first finds the corresponding frame indices between each camera pair, by the help of image feature correspondences and epipolar geometry. Then, for each pair, a relative frame rate and offset are computed by fitting a 2D line to the index correspondences. These pairwise relations define a graph, in which each spanning cycle comprises an absolute synchronisation hypothesis. The optimal solution is found by an exhaustive search over the spanning cycles. The algorithm is experimentally demonstrated to yield highly accurate estimates in a number of scenarios involving static and moving cameras, and Kinect.

1 Introduction

Camera synchronisation involves the temporal alignment of a set of video sequences, independently acquired by two or more cameras. Accurate synchronisation is crucial for a wide variety of applications requiring multi-camera setups, ranging from 3D modelling of dynamic scenes (*e.g.*, featuring a performance, or a sports event) [11] [13] to video surveillance [1] [7] and super-resolution [24]. A camera network can be synchronised via hardware (*e.g.*, a synchronisation signal), or audio signals [13] [6]. However, the feasibility of hardware synchronisation is limited by the size and the spread of the network: in an outdoor scenario such as [18], which involves 16 cameras spread over an area roughly the size of a football pitch, cabling is a tedious and time-consuming job. Moreover, hardware connection may be impractical for a free-moving camera, or the device in question may not even have an external synchronisation capability- as is the case for the ubiquitous mobile devices, and the increasingly popular Kinect. The alternative, audio synchronisation, also suffers from a number of practical difficulties, such as background noise and relatively slow propagation of sound in air, precluding subframe-level precision [13].

In the case of overlapping fields-of-view, an event observed from multiple viewpoints can provide a very strong synchronisation cue. Such an event can be generated, for example, by discharging a flash. However, through-the-lens synchronisation research of the past decade

http://dx.doi.org/10.5244/C.26.103

yielded techniques that can work with generic dynamic content. These techniques can be broadly classified as direct (dense) and feature-based (sparse) approaches [3].

Dense approaches attempt to utilise all available intensity information. Some interesting examples include [6], where 3D phase correlation is employed to estimate a constant index shift between two sequences; and [9] and [7], where the corresponding frames are identified via an image similarity metric. Despite their reported success, these methods are susceptible to appearance or scale changes [3], limiting their application to narrow-baseline scenarios. Moreover, designed for moving camera platforms, [9] and [7] are likely to have difficulties when the temporal intensity variation is confined to a small portion of the scene, drowning the signal in the background noise [7]. Another dense method described in [22] overcomes the baseline limitation by measuring the total intensity variation on a set of corresponding epipolar lines over time, if both a fundamental matrix and a background plate are available.

In contrast to their dense counterparts, sparse approaches employ a small set of distinctive features. They usually need to establish feature correspondences between the sequences. However, there is a considerable variation as to what makes a feature. At one end of the spectrum, space-time interest points [29] [5] localise an event both on the spatial and the temporal axis, but they are reported to be unsatisfactory in the presence of wide-baseline conditions and cluttered background [20]. The entire trajectory of a dynamic (*i.e.*, moving) point can be considered as the other extreme, as a spatial interest point smeared across the temporal axis. [16] and [3] propose techniques to characterise and match point trajectories. Alternatively, the trajectories can be stacked into a matrix, whose rank is bounded by the rigidity of the scene [28] [26]. Another method, presented in [27], estimates the shift between the sequences by aligning the 3D planes spanned by the backprojection rays of a point observed in two sequences. Trajectory-based methods are effective, but their applicability is limited by the availability of long and accurate trajectories across the sequence [27] [16] [26]. This is a non-trivial requirement, especially for scenes involving deformable objects such as humans, or rigid objects with significant appearance variation.

Between these two extremes lies methods that rely on *tracklets*, or even, point correspondences. [23] assesses the fitness of a frame pair candidate by evaluating the rigidity of 9 point correspondences, with at least one dynamic point, and then employs dynamic time warping to find the best index correspondence set. [26] generalises this to homography and affine models. [20] and [14] assume that an estimate of the projective calibration is available (or can be computed from the static background), and utilise the epipolar constraint and the tri-focal transfer error, respectively, to estimate the synchronisation parameters. [25], on the other hand, can work with fully dynamic scenes, such as a blue screen scenario, and given an accurate foreground segmentation, computes a joint calibration and an index shift estimate.

[26], [20], [14] and [25] are unique in the sense that they discuss scenarios with more than 2 cameras. However, [25] and [14] assume a constant frame rate, and estimate only the temporal offset. [20] returns a frame rate and an offset for each camera in the network, but only if there are points visible in all frames, an assumption which imposes restrictions on the content and the camera placement. Moreover, the performance degrades as the number of available features increases beyond a certain level, as the timeline reconstruction algorithm cannot cope with the increasing complexity of the solution space and the outlier ratio [20]. [26] mentions an extension to more than 2 cameras, however, does not present any results.

In this paper, we propose a synchronisation algorithm to estimate a frame rate and an offset for each member of a network composed of 2 or more cameras. The design is guided by the following considerations:

- **Heterogeneity:** The algorithm should be able to handle cameras with different frame rates, resolutions, and motion characteristics. Sample networks include Kinect and HD cameras, multiple Kinects, mobile devices, and hybrid static-moving camera setups for movie production.

- **Modularity:** The 2-camera synchronisation problem requires only pairwise overlapping fields-of-view. Moreover, there is a wealth of literature to draw upon. Therefore, the algorithm should decompose the original problem into pairwise synchronisation tasks, and fuse the resulting *relative synchronisation* estimates for pairs into a consistent and accurate *absolute synchronisation* for the entire set.
- **Spatial locality:** In order to achieve robust wide-baseline performance, the algorithm should follow the feature-based paradigm, preferably with affine-invariant features, such as [17].
- **Temporal locality:** Long and accurate tracks on deformable objects are difficult to establish. Therefore, the algorithm should rely on short tracklets or points.
- **Frugality:** In order to be able to cope with self-occlusions, lack of texture, and viewpoint variation, the algorithm should need as few features as possible, and should not assume the existence of points visible in more than 2 cameras.

The algorithm treats the set of pairwise synchronisation estimates as a graph with potentially missing links, where each vertex represents a camera, and each edge, the relative synchronisation between the cameras it connects. An absolute synchronisation hypothesis can be computed from any cycle that spans the entire vertex set (Figure 3). The fusion step seeks such a cycle which minimises a consistency measure over the index correspondences. The pairwise synchronisation step assesses the similarity of each frame pair in the sequences by the agreement of the feature correspondences with the available geometric constraints (*e.g.*, epipolar constraint), and then estimates the subframe index shift which maximises this score. A Viterbi scheme establishes the index correspondences while imposing the ordering constraint. The frame rate and the offset are computed by robustly fitting a 2D line to the index correspondence set. The novel contributions are:

- **A bottom-up absolute synchronisation framework.** We propose a new, graph-based synchronisation fusion framework, which seeks an absolute synchronisation that explains the observed index correspondences. Unlike the sequential method presented in [14], a graph is not order dependent, and can represent the relationships between non-neighbouring pairs. Our approach is superior to the graph-based method of [25], as it can incorporate more constraints, in the form observed index correspondences. Also, unlike [14] and [25], it does not assume a known frame rate, and unlike [20], existence of points visible in all cameras.
- **A new pairwise synchronisation algorithm:** We propose a novel 2-camera synchronisation algorithm, which fits a 2D line to the observed subframe index correspondences. Similar to [7], the algorithm can exploit the ordering constraint (*i.e.*, monotonicity of the index pairs) via dynamic programming, but it also benefits from a linear prior to reject the outlier correspondences. [26] employs 2D line-fitting, but lacks a mechanism as powerful as the ordering constraint to eliminate the outliers. Subframe refinement over a trajectory is proposed in [3], but unlike our algorithm, can only operate over one direction. The combination of these techniques leads to a robust algorithm which is unique in its ability to work with minimal amount of image information: the minimal requirement is two (possibly independent) point correspondences observed at different time instants.

The rest of the paper is organised as follows: The 2-camera synchronisation and the fusion algorithms are discussed in the next section. The performance of the algorithm is

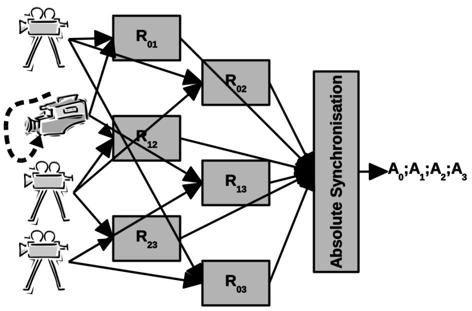

Figure 1: Overview of the algorithm, with one moving and 3 static cameras. R_{ij} blocks compute the relative synchronisation between the ith and the jth cameras. A_i is the absolute synchronisation for the ith camera.

experimentally demonstrated in Section 3, on a variety of networks involving static and moving HD cameras and a Kinect. Section 4 presents the conclusions.

2 Synchronisation Framework

2.1 Problem Definition

Assuming no frame-drops, the relation between the indices of an image sequence, t, and the reference timeline, t_r, is captured by the line [20]

$$t = \alpha t_r + \tau, \tag{1}$$

where α is the frame rate of the camera, and τ is the offset between the first frame of the sequence and the origin of the timeline. Given an ensemble of N cameras, the network synchronisation problem involves the estimation of the sets $\{\alpha_i\}_{i=0}^{N-1}$ and $\{\tau_i\}_{i=0}^{N-1}$, a frame rate and an offset for each camera in the network. The absolute synchronisation for the ith camera, A_i is defined as the pair $\{\alpha_i;\tau_i\}$.

Through Equation 1, the relative synchronisation of the jth camera with respect to the ith defines the line linking the associated index sets, t_j and t_i as

$$t_j = \alpha_{ij} t_i + \tau_{ij}. \tag{2}$$

R_{ij} denotes the corresponding relative synchronisation estimate, the pair $\{\alpha_{ij};\tau_{ij}\}$.

The absolute synchronisation algorithm (Section 2.3) obtains $\{A_i\}$ from $\{R_{ij}\}$ (Figure 1). In order to compute R_{ij}, two inputs are necessary: The geometric relationship between the images, imposed by the scene and the imaging geometry, *e.g.*, a fundamental matrix ($\mathbf{F_{ij}}$), or a homography ($\mathbf{H_{ij}}$); and a set of feature tracks on the dynamic scene elements.

In the synchronisation literature, $\mathbf{F_{ij}}$ and $\mathbf{H_{ij}}$ are often estimated from the static background [20] [14] [26], by various methods described in [12]. In our experiments, we calibrated the static cameras by [19], and the moving cameras by [8].

As for the tracks, since the algorithm does not rely on long trajectories, simple trackers such as KLT [2] are sufficient. In our implementation, we track the Hessian-affine features [17] described by SIFT [15], as affine-invariance is necessary for wide-baseline matching across the members of the network. In order to weed out the static trajectories, which do not carry any useful information for synchronisation, we estimate the homographies relating the successive frames in a sequence, and retain the features that do not conform to the model. Such features are either on dynamic trajectories, or genuine outliers. However, if foreground

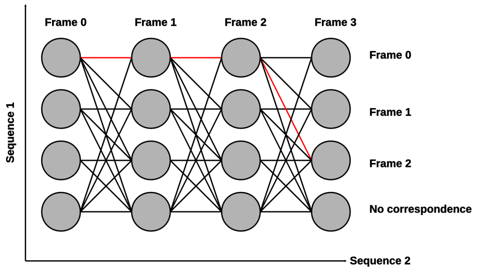

Figure 2: Trellis diagram for two sequences, with 3 and 4 images, respectively. An admissible path is marked by red, for the correspondence set $\{f_1^0;f_2^0\}$, $\{f_1^1;f_2^0\}$, $\{f_1^2;f_2^0\}$, $\{f_1^3;f_2^1\}$

masks are available, they can be used to identify and remove the static features.

2.2 Relative Synchronisation

The relative synchronisation block estimates $R=\{\alpha;\tau\}$ for a camera pair, by fitting a line to the indices of the matching frame pairs. The frame matching process is constrained by the ordering rule, which states that, if the pth frame in the first sequence (f_1^p) corresponds to the qth frame in the second (f_2^q), f_2^{q+1} cannot be matched to a frame preceding f_1^p. This is analogous to the stereo-matching problem, which can be solved via dynamic programming [4]. On the trellis in Figure 2, each axis represents a sequence, and each node, a correspondence hypothesis. A node N_p^q holds a value measuring the strength of the hypothesised match $\{f_1^p;f_2^q\}$. The transition probability indicates whether $\{f_2^{p'};f_2^{q+1}\}$ is a valid match with respect to the ordering constraint, given $\{f_1^p;f_2^q\}$. Each column of the trellis is augmented by a node $N_\emptyset^q$, indicating a no-correspondence event. The no-correspondence nodes are exempt from the ordering constraint.

The transition probability from N_p^q to $N_{p'}^{q+1}$ is defined as

$$T(N_p^q,N_{p'}^{q+1})=\begin{cases}1 & p\leq p'\\ 1 & p\vee p'=\emptyset\\ 0 & \text{otherwise}\end{cases}. \tag{3}$$

This formulation allows one-to-many matches, so that the algorithm can handle the cases where $\alpha\neq 1$. The admissible transitions can be further constrained through an initial estimate of the synchronisation parameters, if available.

M_p^q, the match strength for $\{f_1^p;f_2^q\}$ is computed by first establishing the feature correspondences between the two images, guided by the available geometric constraint [12], *e.g.*, **F** or **H**. Then, the distance of each correspondence to the constraint is calculated. M_p^q is a function of the median value of the distances for the entire correspondence set. More concretely, if the geometric constraint is **F** and the constraint violations are measured by the Sampson error [12], the value the node N_p^q holds is

$$M_p^q=\begin{cases}W(\underset{\{\mathbf{x_p};\mathbf{x_q}\}\in K}{\text{median}}S(\mathbf{x_p},\mathbf{x_q},\mathbf{F})) & |K|>\nu_K\\ \varepsilon & q=\emptyset\\ 0 & \text{otherwise}\end{cases}, \tag{4}$$

where $\mathbf{x_p}$ is a homogeneous point coordinate in f_1^p, $\mathbf{x_q}$ is the corresponding coordinate in f_2^q, and $\{\mathbf{x_p};\mathbf{x_q}\}$ is a member of the set of correspondences, K. ν_K is a threshold over the cardinality of K. S denotes the Sampson error function, and W, a truncated Gaussian

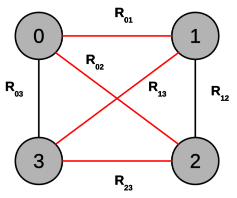

Figure 3: Synchronisation graph for 4 cameras. The red line traces a spanning cycle.

nonlinearity that converts the constraint error to a fitness score in $[0,1]$. ε is a positive scalar that indicates the minimum admissible match score. Although image feature matching is a computationally demanding process, the use of guided matching on relatively small sets of features helps to mitigate the cost.

Subframe resolution is achieved by minimising a shift parameter, β, over the point trajectory. Given a point $x_q \in f_2^q$, the preceding point x_{q-1}, and the succeeding point x_{q+1}, β is the solution to the minimisation problem

$$\beta = \underset{\beta' \in [-1,1]}{\arg\min} \sum_{\{\mathbf{x_p},\mathbf{x_q}\} \in K} S(\mathbf{x_p}, \mathbf{x_q}(\beta'), \mathbf{F})$$
$$\text{where } \mathbf{x_q}(\beta') = \begin{cases} (1-\beta')\mathbf{x_q} + \beta'\mathbf{x_{q+1}} & \beta' > 0 \\ (1+\beta')\mathbf{x_q} - \beta'\mathbf{x_{q-1}} & \beta' \leq 0 \end{cases} \tag{5}$$

The minimisation is performed over K via Powell's dog leg algorithm [21] , and β is initialised as 0. Unlike [3], the minimisation can be performed over crooked paths. If $\mathbf{x_q}$ is a terminal point, depending on whether it is the first or the last, $\mathbf{x_{q-1}}$ or $\mathbf{x_{q+1}}$ is set to $\mathbf{x_q}$.

The first step, frame matching, processes the frame pairs independently, hence, does not need any tracks, or a fixed $\mathbf{F}$. In order to use the subframe refinement with varying $\mathbf{F}$, $\mathbf{x_{q-1}}$ and $\mathbf{x_{q+1}}$ should be registered to the same frame as $\mathbf{x_q}$. However, since the camera motion in successive frames satisfies the narrow-baseline conditions, the mapping can be approximated as a homography, and computed from $\mathbf{F_{q-1}}, \mathbf{F_q}$ and $\mathbf{F_{q+1}}$.

The subframe refinement step is not symmetrical: fixing $\mathbf{x_q}$ and moving $\mathbf{x_p}$ on its trajectory is likely to yield a different β. Therefore, the matching algorithm is repeated twice, by swapping the roles of the sequences. Then a RANSAC-based [10] robust line fitting algorithm is applied to the index correspondences. As per Equation 2, the resulting line defines the relative synchronisation between the cameras. If α is known, this step is reduced to a 1-dimensional offset estimation problem.

The algorithm is presented for a fundamental matrix, however, an equivalent derivation for a homography is straightforward. In this case, the Sampson error should be replaced by the transfer error.

It should be noted that this method, like any other method relying on epipolar geometry [23] [26] [20], is not applicable when the motion is purely along the epipolar line. However, this is an issue only with specific combinations of camera setups and scenes, and seldom a problem with general networks, and generic video content.

2.3 Absolute Synchronisation

The relative synchronisation algorithm of Section 2.2 is applied to all camera pairs, to obtain the set $\{R_{ij}\}$. Upon removing the members with too few index correspondences, the remaining $\{\alpha_{ij}; \tau_{ij}\}$ pairs can be represented as a graph (Figure 3), where each vertex is a camera, and each link is a relative synchronisation. The relation between R_{ij} and the corresponding

Algorithm 1 Evaluation of an Absolute Synchronisation Hypothesis

```
Input: ${A_i}$, and $I$, the index correspondences for all relative synchronisations in the cycle
Output: A scalar indicating the fitness of ${A_i}$
  1. For each $R_{ij}$ in the cycle, compute a $\hat{R}_{ij}$ from the corresponding $A_i$ and $A_j$ (Equation 6).
  2. For each $\hat{R}_{ij}$, compute the Euclidean distance of the index correspondences between
     the $i$th and the $j$th camera ($I_{ij}$) to the line defined by $\{\alpha_{ij};\tau_{ij}\}$. The evaluation metric
     for a pair is the sum of all Euclidean distances over $I_{ij}$.
  3. The evaluation metric for a cycle is the sum of all metrics over the $\{i;j\}$ pairs in the
     cycle.
```

absolute synchronisations, A_i and A_j is

$$\begin{aligned} \alpha_{ij} &= \frac{\alpha_j}{\alpha_i} \Rightarrow \quad \alpha_{ij}\alpha_i - \alpha_j = 0 \\ \tau_{ij} &= \tau_j - \frac{\alpha_j}{\alpha_i}\tau_i \end{aligned} \tag{6}$$

For an $L-$camera set, a spanning cycle provides L constraints, which can be stacked into the linear equation systems of the following form:

$$\mathbf{A}[\alpha_0 \dots \alpha_{L-1}]^T = \mathbf{0} \tag{7}$$

$$\mathbf{B}[\tau_0 \dots \tau_{L-1}]^T = \mathbf{b} \tag{8}$$

Both systems can be solved via standard linear algebra techniques. We first solve Equation 7, and use the result in Equation 8. Alternatively, the systems can be solved jointly, but the potentially large scale difference between $\{\alpha_i\}$ and $\{\tau_i\}$ is likely to cause problems.

Equation 7 can be solved only up to an unknown scale. We fix the scale by setting $\alpha_0 = 1$. Also, since the origin is arbitrary, we set $\tau_0 = 0$. Therefore, the first camera provides the reference timeline t_r in Equation 1.

Since not all R_{ij} are equally accurate, the quality of the final $\{A_i\}$ depends on the exact cycle, *i.e.*, the exact set of relative synchronisation estimates, selected to solve Equations 7 and 8. Consistency of a cycle, *i.e.*, whether the composition of all transformations along the cycle is identity, can be used as a measure of quality. However, our experiments indicated that two poor relative synchronisation estimates can cancel each other out, yielding a consistent cycle, but an incorrect $\{A_i\}$. Instead, we do the evaluation directly on the index correspondences by using Algorithm 1.

The absolute synchronisation is performed via Algorithm 2, which exhaustively evaluates all spanning cycles in the graph through Algorithm 1. Any promising cycles are further refined by minimisation over the set of consistent relative synchronisations (steps 2-4). The computational load due to the exhaustive search is negligible, as often the graph is not fully connected, and any unpromising hypotheses can be culled early on.

3 Experimental Results

The performance of the algorithm is evaluated on 3 scenarios, *Acrobatics*, *Stretch* and *Possessed* (Figures 4, 5 and 6). *Acrobatics* involves a static network of 8 HD cameras and a Kinect viewing an actor doing somersaults and handstands for 23 seconds. *Stretch* and *Pos-*

Algorithm 2 Computation of an Absolute Synchronisation Hypothesis

```
Input: $\{R_{ij}\}$ and $\{I_{ij}\}$
Output: $\{A_i\}$
For each spanning cycle in the graph
    1. Evaluate the cycle via Algorithm 1.
    2. If it is better than the current best, compute $\hat{R}_{ij}$ for each element of $\{R_{ij}\}$ (i.e., not just
       the ones in the cycle) via Equation 6. Any $R_{ij}$ that satisfies the conditions $|\alpha_{ij} - \hat{\alpha}_{ij}| <$
       $\nu_{alpha}$ and $|\tau_{ij} - \hat{\tau}_{ij}| < \nu_{tau}$ is an inlier synchronisation. Else, next cycle.
    3. Re-estimate $\{A_i\}$ over all inliers, via Equation 7 and 8.
    4. Refine $\{A_i\}$ by minimising the total Euclidean distance computed over the index cor-
       respondences of all inlier synchronisations.
```

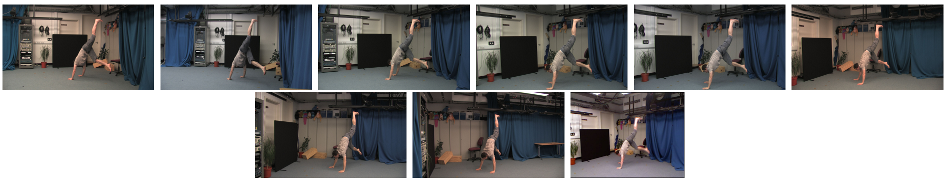
Figure 4: *Acrobatics. Top:* Cameras 0-6. *Bottom:* Camera 7, and the Kinect.

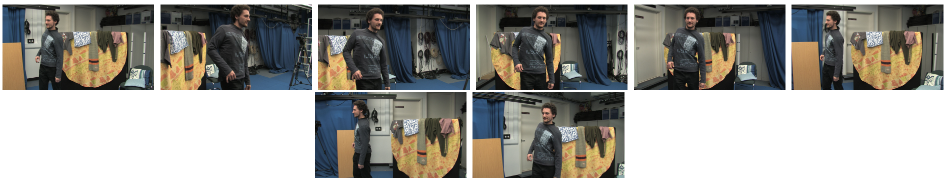
Figure 5: *Stretch. Top:* Cameras 0-6. *Bottom:* The nodal camera.

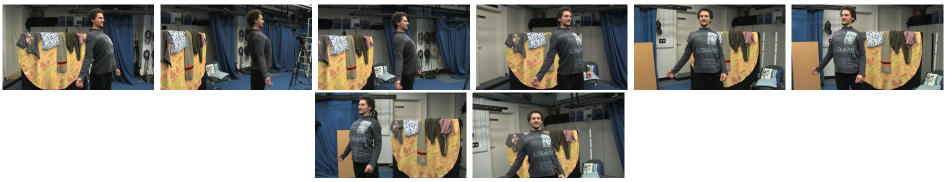
Figure 6: *Possessed. Top:* Cameras 0-6. *Bottom:* The free-moving camera.

sessed feature 1 moving (nodal, and free-moving, respectively) and 7 static HD cameras, viewing an actor rocking and stretching for 15 seconds. The Kinect captures images at a resolution of 640x480 pixels, at a frame rate of 30 fps, whereas for the HD cameras, the resolution is 1920x1080 and the frame rate is 25 fps. The ground-truth values for τ in Table1 is obtained by first hardware-synchronising the HD cameras, and then introducing a known shift. τ for the Kinect is manually determined from the images. The details of the calibration and tracking are discussed in Section 2.1. Table 1 shows the results.

Acrobatics is a good example of the challenges posed by a heterogeneous camera network: The Kinect has a lower resolution and slower shutter speed, causing blur. Moreover, the white balance exhibits a significant variation across the set, and the Kinect images suffer from saturated patches. The content involves fast motion and rapid deformation of textureless elements. This proves to be a double-edged sword: It provides a strong synchronisation cue, but also leads to motion blur, and difficulties in feature tracking and matching. However, Table 1a indicates that the algorithm achieves a very high accuracy, both in α and τ. These

(a) Acrobatics

	0	1	2	3	4	5	6	7	Kinect
α	1	1	1	1	1	1	1	1	1.2
τ	0	-11	-15	-31	-9	-27	-26	-32	≈ 26.8
$\Delta\alpha$	0	5.8×10^{-7}	1.1×10^{-6}	7.6×10^{-7}	1.1×10^{-6}	1.2×10^{-6}	7.1×10^{-7}	1.7×10^{-6}	5.9×10^{-4}
$\Delta\tau$	0	5.8×10^{-4}	1.3×10^{-4}	3.7×10^{-4}	6.8×10^{-5}	2.7×10^{-4}	1.9×10^{-5}	7.6×10^{-4}	6.2×10^{-2}

(b) Stretch

	0	1	2	3	4	5	6	Nodal
α	1	1	1	1	1	1	1	1
τ	0	-30	-1	-5	-17	-13	-10	0
$\Delta\alpha$	0	9.7×10^{-6}	1.9×10^{-6}	5.2×10^{-6}	2.6×10^{-6}	2.4×10^{-6}	3.4×10^{-7}	7.0×10^{-7}
$\Delta\tau$	0	3.5×10^{-3}	4.7×10^{-4}	8.6×10^{-5}	1.8×10^{-4}	1.2×10^{-3}	3.0×10^{-4}	3.6×10^{-4}

(c) Possessed

	0	1	2	3	4	5	6	Free
α	1	1	1	1	1	1	1	1
τ	0	-4	-24	-16	-9	-38	-5	0
$\Delta\alpha$	0	3.5×10^{-5}	3.2×10^{-5}	3.5×10^{-5}	3.4×10^{-5}	3.6×10^{-5}	3.0×10^{-5}	2.6×10^{-5}
$\Delta\tau$	0	2.0×10^{-4}	5.0×10^{-4}	7.0×10^{-4}	3.5×10^{-4}	1.4×10^{-3}	2.0×10^{-3}	4.7×10^{-3}

Table 1: Absolute synchronisation estimates, with Camera 0 as the reference. α and τ are the ground-truth values of the synchronisation parameters, whereas $\Delta\alpha$ and $\Delta\tau$ indicate the *absolute value* of the estimation error. The ground-truth offset for the Kinect is not known, but manually determined from the sequence.

figures should be contrasted with those reported by two recent methods [26] and [20] on their own datasets: best α and τ errors are 0.0041 and 1.4 for [26], and 0.0027 and 0.34 for [20], respectively; *i.e.*, several orders of magnitude higher from those presented in Table 1.

Stretch and *Possessed* demonstrate the performance of the algorithm when the network has dynamic elements. The slow and repetitive rocking motion, repetitive texture, and potential calibration errors all introduce an ambiguity to the frame matching process, and cause mismatches. However, as shown in Tables 1b and 1c, the algorithm effectively copes with these issues. Although the absence of subframe shifts probably improves the performance, Table 1a clearly shows that the algorithm can work under subframe shifts (introduced by the frame rate difference, as well as the lack of hardware synchronisation).

The algorithm owes its robustness and accuracy to the following:

- Dynamic programming is an effective method to resolve the ambiguities in the matching process, as it enforces a higher-level consistency constraint. It is also very robust to large shifts, which only changes the location of the node corresponding to the correct match in the trellis, without significantly altering its match fitness score.
- The use of tracklets and affine-invariant features mitigates the difficulties posed by fast motion, deformable objects and viewpoint change.
- Fitting a line to estimate the relative synchronisation gives the algorithm the ability to cope with index correspondence sets with a very large number of outliers and few true matches: In 600 frames, we regularly observe that less than 3% inliers are sufficient for a satisfactory estimate.
- The decomposition of the global synchronisation problem into a set of pairwise synchronisation problems allows effective exploitation of the high-quality estimates ob-

tained from neighbouring camera pairs.

- The absolute synchronisation stage eliminates the erroneous pairwise relations, and effectively pools the information extracted from the entire data to improve the accuracy of the results.

4 Conclusion

This paper presents an algorithm for the synchronisation a network of 2 or more cameras. The algorithm employs a pairwise synchronisation block, which, through the use of guided feature matching and dynamic programming, first finds a set of index correspondences, and then computes the optimal subframe shifts. The pairwise relations are fused into an absolute synchronisation estimate, by finding the best spanning cycle over a graph of relative synchronisations, and then further refined by imposing all available pairwise synchronisation and index correspondence constraints. Our contributions include a new 2-camera synchronisation algorithm, and a novel framework for fusing the relative synchronisation estimates.

The algorithm follows a bottom-up approach, leveraging both the robustness and frugality of the 2-camera solver, and the precision offered by multiple constraints. It is shown to perform very well on a number of scenarios involving a Kinect and moving cameras.

Acknowledgements: This work is supported by the TSB project “SYMMM: Synchronising Multimodal Movie Metadata”.

References

[1] J. Black and T. Ellis. Multi camera image tracking. In *Proc. PETS*, 2001.

[2] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. Technical report, Intel Corporation Microprocessor Research Labs, 2000.

[3] Y. Caspi, D. Simakov, and M. Irani. Feature-based sequence-to-sequence matching. *International Journal of Computer Vision*, 68(1), June 2006.

[4] I. J. Cox, S. L. Hingorani, B. M. Maggs, and S. B. Rao. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, May 1996.

[5] D. Huynh D. Wedge and P. Kovesi. Using space-time interest points for video sequence synchronization. In *Proc. IAPR Conf. Machine Vision Applications*, pages 190–194, 2007.

[6] C. Dai, Y. Zheng, and X. Li. Subframe video synchronization via 3d phase correlation. In *Proc. ICIP*, pages 501–504, 2006.

[7] F. Diego, D. Ponsa, and J. Serrat. Video alignment for change detection. *IEEE Transactions on Image Processing*, 20(7):1858–1869, July 2011.

[8] J.-Y. Guillemaut E. Imre and A. Hilton. Calibration of nodal and free-moving cameras in dynamic scenes for post-production. In *Proc. The First Joint 3DIM/3DPVT Conference*, pages 260–267.

[9] G. D. Evangelidis and C. Bauckhage. Efficient and robust alignment of unsynchronized video sequences. In *Proc. DAGM*, pages 286–295, 2011.

[10] A. Fischler and C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] J.-Y. Guillemaut and A. Hilton. Joint multi-layer segmentation and reconstruction for free-viewpoint video applications. *International Journal of Computer Vision*, 93(1): 73–100, 2011.

[12] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[13] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall, and H.-P. Seidel. Markerless motion capture with unsynchronized moving cameras. In *Proc. CVPR*, pages 224–231, 2009.

[14] C. Lei and Y.-H. Yang. Tri-focal tensor-based multiple video synchronization with subframe optimization. *IEEE Transactions on Image Processing*, 15(9), September 2006.

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[16] B. Meyer, T. Stich, M. Magnor, and M. Pollefeys. Subframe temporal alignment of non-stationary cameras. In *Proc. BMVC*, 2008.

[17] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[18] A. Miller. R&d and blue peter- ski rossendale free-viewpoint visualisation. http://www.bbc.co.uk/blogs/researchanddevelopment/2011/03/rd-and-blue-peter--ski-rossend.shtml.

[19] J. Mitchelson and A. Hilton. Wand-based multiple camera studio calibration. Technical Report VSSP-TR-2/2003, University of Surrey, CVSSP, 2003.

[20] F. L. C. Padua, R. L. Carceroni, A. M. R. G. Santos, and K. N. Kutulakos. Linear sequence-to-sequence alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):304–320, Februrary 2010.

[21] M. J. D. Powell. A hybrid method for nonlinear equations. *Numerical Methods for Nonlinear Algebraic Equations*, 1970.

[22] D. Pundik and Y. Moses. Video synchronization using temporal signals from epipolar lines. In *Proc. ECCV*, volume 3, pages 15–28, 2010.

[23] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood. View-invariant alignment and matching of video sequences. In *Proc. ICCV*, volume 2, pages 939–945, 2003.

[24] Eli Shechtman, Yaron Caspi, and Michal Irani. Space-time super-resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):531–545, 2005.

[25] S. N. Sinha and M. Pollefeys. Camera network calibration and synchronization from silhouettes in archived video. *International Journal of Computer Vision*, 87(3):266–283, May 2010.

[26] P. A. Tresadern and I. D. Reid. Video synchronisation from human motion using rank constraints. *Computer Vision and Image Understanding*, 113(8):891–906, August 2009.

[27] T. Tuytelaars and L. v. Gool. Synchronizing video sequences. In *Proc. CVPR*, volume 1, pages 762–768, 2004.

[28] L. Wolf and A. Zomet. Wide baseline matching between unsynchronized video sequences. *International Journal of Computer Vision*, 68(1):43–52, June 2006.

[29] J. Yan and M. Pollefeys. Video synchronization via space-time interest point distribution. In *Proc. ACIVS*, pages 501–504.