Electronic Letters on Computer Vision and Image Analysis 9(1):1-17; 2010

Formalization of the General Video Temporal Synchronization Problem

Anthony Whitehead[*], Robert Laganiere[+], Prosenjit Bose[*]

[*] *Carleton Univeristy, 1125 Colonel By Drive, Ottawa,Canada*

[+]*University of Ottawa, 550 Cumberland Street, Ottawa, Canada*

Received 1st May 2009; accepted 15th March 2010.

Abstract

In this work, we present a theoretical formalization of the temporal synchronization problem and a method to temporally synchronize multiple stationary video cameras with overlapping views of the same scene. The method uses a two stage approach that first approximates the synchronization by tracking moving objects and identifying curvature points. The method then proceeds to refine the estimate using a consensus based matching heuristic to find frames that best agree with the pre-computed camera geometries from stationary background image features. By using the fundamental matrix and the trifocal tensor in the second refinement step, we improve the estimation of the first step and handle a broader more generic range of input scenarios and camera conditions. The method is relatively simple compared to current techniques and is no harder than feature tracking in stage one and computing accurate geometries in stage two. We also provide a robust method to assist synchronization in the presence of inaccurate geometry computation, and a theoretical limit on the accuracy that can be expected from any synchronization system.

Key Words: Computer Vision, Image Analysis, Pattern Recognition, Image Segmentation, 3D Reconstruction, Active Vision, Tracking, Video and Image Sequence Analysis.

1 Introduction

There are many common applications of multiple video cameras today that range from video surveillance of large areas such as shopping centers, parking lots and campuses, to videography and filmmaking that utilize multiple video cameras when shooting individual scenes. However, in some situations such as photogrammetry and camera metrology, the use of multiple cameras is required when there are moving objects in the scene [1]. As different human operators may control these cameras, and only in certain situations is it feasible to use a professional camera synch slate, there is the fundamental problem of sequence synchronization that needs initial resolution.

A *camera sequence* (S) is the linear sequence of frames from a single video camera; like a single reel of film. A *cross camera subset* (CCS) is a set of K images, where each image in the subset comes uniquely from one of the K cameras, e.g. CCS=$\{f_1, f_2, \ldots, f_K\}$. A cross camera subset is not necessarily aligned in time, we denote a CCS to be simply a selection of K frames, one from each of K camera sequences. A *synchronized CCS* is a cross camera subset where each frame of the set is full frame synchronized to the same point in time. The problem of camera synchronization is that of determining the exact the same moment in time for each of the video sequences, i.e. finding a synchronized CCS.

Correspondence to: <awhitehe@connect.carleton.ca>

Recommended for acceptance by <Xiaoyi Jiang>
ELCVIA ISSN: 1577-5097
Published by Computer Vision Center / Universitat Autonoma de Barcelona, Barcelona, Spain

Intuitively, the synchronization problem refers to the following: Given K different video sequences that overlap in time, identify one frame from each of the different sequences that refer to the same point in time. Such a set of frames is called a *synchronized cross camera subset* or a synchronized CCS. More formally, for each video sequence *i*, let the *frame-time* function $T_i(f)$ map an integral frame number f of sequence i to a universal time (a Real number), i.e.

$$T_i : \mathrm{N} \rightarrow \mathrm{R} \tag{1}$$

This manifests itself as a function of the camera frame rate and time.

The synchronization problem can now be expressed as finding a set of integer frames numbers, CCS = $\{f_1, f_2, \ldots, f_K\}$, one from each video sequence, such that the frame time function $\mathrm{T_i}$ yields the same results for each frame of the set of frame numbers creating the *synchronization equality:*

$$T_1(f_1) = T_2(f_2) = \ldots = T_K(f_k) \text{ with } f_k \in N \tag{2}$$

Such a set of frames that exactly solves the synchronization equality is said to be in *perfect integral (or full frame) synchronization*.

However, due to possible minute variations in camera start times and variations in frame rates, perfect integral synchronization does not generally exist. If we remove the restriction of integral frame numbers, the frame-time function maps frame values (possibly non-integral) to a universal point in time, i.e:

$$T_i : R \rightarrow R \tag{3}$$

In this case, the frame-function maps a Real frame number (sub-frame accurate) to an exact moment in time. In such a case, there will always exist a set of real frame numbers, CCS=$\{f_1, f_2, \ldots, f_K\}$, one from each video sequence such that synchronization equality:

$$T_1(f_i) = T_2(f_j) = \ldots = T_K(f_K) \text{ with } f_k \in R$$

holds.

In summary the synchronization problem can be referred to as:

1. … the *full frame synchronization problem* when restricted to integral frame numbers. This is the problem in the discrete time domain and seeks to find a set of frame numbers CCS=$\{f_1, f_2, \ldots, f_K\}$ that minimizes the pair-wise differences of the *synchronization equality*. i.e:

$$\min(\text{For all i,j in CCS}, \sum |T_i(f_i) - T_i(f_j)|) \tag{4}$$

2. … the *exact synchronization problem* when unrestricted, and seeks to exactly solve the *synchronization equality* for any given point in time. This is the problem in the continuous time domain.

Practically speaking, all work is done in the discrete domain (for example computing the geometries, tracking etc) the images used are discrete time samplings. Once you have synchronized your cameras (continuous or discrete) you then have some follow on computational problem. For example, when you have to do a 3D reconstruction you still have to use the full frames, which in theory have a known offset in time, but since you cannot time warp the frames to the exact same time the problem continues in the discrete domain. This will bound your accuracy for tasks like 3D reconstruction, but continuous domain synchronization does not necessarily aid the follow on problem. We address both the theory (continuous) and the practice (discrete) using the same model and fully explore the details of the functions, the equality and their use in both flavours of the synchronization problem in section 2.

1.1 Additional Background

Synchronization is often assumed, however, since the processing of large volumes of video data is becoming tractable, recent work has investigated the problem of automatic synchronizing of video sequences. In [3] the synchronization problem is constrained to having a large planar surface present. The method computes the homography that describes the transformation of the ground plane and looks for the frame pair with the most consensus of the moving objects. The method suffers under certain 3D motions such as similar objects moving in a line with constant velocity due to the likelihood that objects will be similar enough to create false matching and consensus inconsistencies. In [4], the method is also constrained by a large ground plane being present, but further requires intrinsic camera parameters so that the 3D information about trajectories can be computed and subsequently corrected in conjunction with the epipolar geometry. Furthermore, the method assumes a homogenous camera system. In [5], a fixed set of extrinsic camera parameters, identical frame rates, and a static scene are required so that motion of the rig is identical on a frame to frame basis. This allows the algorithm to simply find the matching geometric changes between frames N and N+1 for camera 1, M and M+1 for camera 2, leaving the offset in frames being |M-N|. This is a formal calibration process that is not always possible. In [6], the authors also take advantage of the fact that objects moving on a planar surface produce a 3D trajectory contour that is identical from camera to camera. Upon finding the contour similarities, the frame synchronization is identified. Under repeating (periodic) motion the contours will be identical, and synchronization will not be possible. In [7], the synchronization is based on viewing similar non planar 3D motion trajectories in time with applications to telelearning so that exact precision in synchronization is not fundamentally necessary. In [7, 8], the imposition of rank constraints on corresponding frame features is examined, rather than the epipolar geometry. In order to determine the synchronization a search is performed for frame pairs that minimize the rank constraint. However, in robust computations of the epipolar geometry, the rank constraint should be enforced. In [21] the synchronization of two cameras is recovered by the estimation of a planar homography or by the estimation of the epipolar geometry through the Fundamental matrix. The use of trajectories to determine the space/time synchronization is done by assuming temporal cues consistent in the two views. However this assumption is not view invariant and the peak height of a trajectory may occur at two different points in time in two different views. Our work shows that trajectory features can be used to recover a close synchronization, but the exact synchronization requires a more detailed examination of the frames surrounding the trajectory feature.

Generally speaking these methods are restrictive because of the requirements of large planar surfaces being present or the requirement that the camera system be partially, if not fully, calibrated or they are restricted to pairs of cameras. More recently, examinations into automatic synchronization involving overlapping camera viewpoints and dynamic objects and scenes have been investigated in [16-21] with the same intention. Furthermore, as synchronization is simply a means to an end, they examine the full frame synchronization problem rather than detail the exact synchronization problem. Conversely, one of our contributions brings forth a theoretical foundation of the generalized problem and proves the bounds of the synchronization error regardless of the synchronization method employed. As well, the method presented here is valid for both heterogeneous camera systems as well as homogeneous camera systems and is strictly independent of the capture system hardware and the independent of the scene altogether.

In this work we examine the theoretical nature of the synchronization of multiple video sequences and prove the maximum upper bound on the difference between full frame and exact synchronizations. We propose a novel method that handles a much larger set of input sequences and does not rely on any particular camera configuration or constraints on the objects. The method is performed solely in projective space and does not require trajectory correspondence to be solved apriori. The solution is oriented around the trilinear tensor and trajectory curvatures to solve the space/time synchronization problem. We provide a solution to compute the camera geometries and the retrieve the camera synchronization. The main constraint of our method is that there are at least three cameras that remain stationary throughout the video capture process; a very common situation in many multi-video applications. The motion of the moving objects is slightly constrained in that it cannot have a periodic characteristic such as a pendulum, nor can the motion be directly along the optical axis of one of the cameras or in the epipolar plane of any of the cameras pairs being synchronized. Any camera count over three can be handled by our method on an overlapping basis.

2 Synchronization Problem Formalization

We begin by defining and formalizing the synchronization problem in general. This formalization outlines the synchronization problem independent of any proposed solution in both the continuous and discrete time domains. We will introduce terminology and concepts that will allow us to clearly bound the accuracy of any synchronization. Moreover, we present the mathematical formalization of the problem and follow up by introducing terminology used in the description of the proposed solution.

2.1 The synchronization problem

We first examine some properties of the relationship between multiple video sequences and specify terminology. We let C_k:R→{0,1} be the *frame-capture function* for a sequence k. $C_k(t) = 1$ if at time *t*, a frame in video sequence k is being captured and $C_k(t) = 0$ otherwise. Close examination reveals that the frame-capture function exhibits periodicity and therefore the model for video capture and synchronization we use is not linear as one might expect by looking at the frame time functions of (1) and (3). The time between peaks in the function C_k is known as the *period* and what is commonly referred to as the *frame rate* (*ρ)*, is actually the *frequency*. Recall that frequency and the period are inversely related. Figures 1 and 2 plot the function $C_{i,}$ C_j , C_k for 3 cameras. The peaks (value 1) occur when a frame is being captured; the length of the plateau is effectively the shutter speed of the camera, and the valleys occur (value 0) when frames are not being captured. Notice that in the case of multiple video sequences there exist what we call *primary synchronization points* that minimize the distance between the exact synchronization times and the full frame synchronization times.

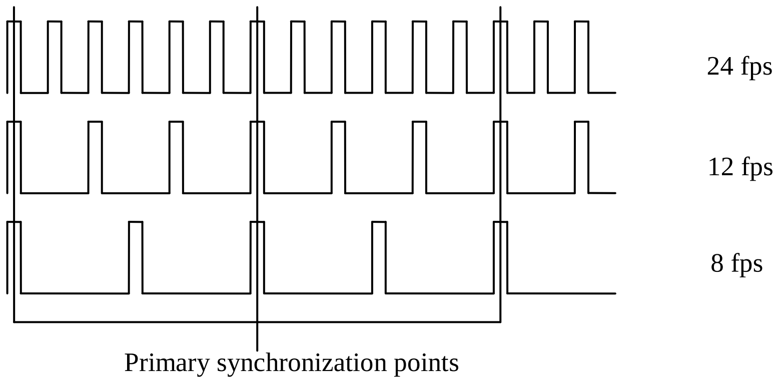

Figure 1: Perfect integral synchronized video sequences with varying frame rates showing primary synchronization points. Note that not all frames are perfectly synchronized and some frame sets provide better synchronization than others.

Defintion: A *primary synchronization point* is made of the set of real frame numbers CCS={f_1, f_2, …, f_K} that, when its elements are rounded to the nearest integers, satisfies the full frame synchronization condition. Any synchronization point that is not a primary synchronization point is termed a *secondary synchronization point*. As we see in Figure 1, given 3 video sequences of differing frame rates that are perfectly synchronized in time, clearly visible cycles of primary synchronization occur. For perfectly synchronized video sequences, these primary synchronization points correspond to the full frames that were taken at the exact same moment in time and as we shall discuss next, perfectly synchronized video sequences are rare in practice. Knowing the difference between the discrete solution vs. the continuous solution can allow you to know if you have a good set of frames for 3D reconstruction for example. Thus a primary synchronization point is the point in time that minimizes this difference between the discrete and continuous domain solutions.

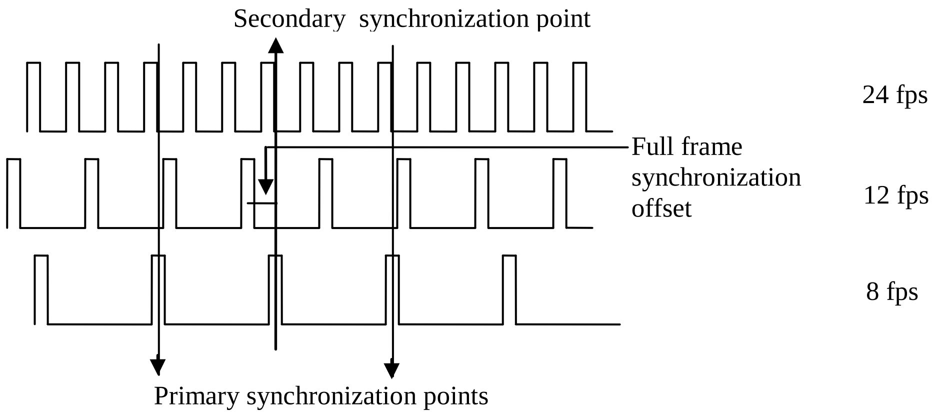
Secondary synchronization point
24 fps
Full frame synchronization offset
12 fps
8 fps
Primary synchronization points

Figure 2: Imperfectly synchronized video sequences with varying frame rates showing primary and secondary synchronization points. Note the primary synchronization points minimize the difference between the discrete and continuous solutions.

In this work, we assume that the frame rate for each sequence is known. Practically speaking this is safe since frame rates are stored in the video files or is available directly from the capture device. Primary synchronization points occur at regular intervals that are a function of the frame rates of the individual sequences. The number of frames between these primary synchronization points for two sequences *i* and *j* is a function of the frame rates given by:

$$frames(\rho_i, \rho_j) = \frac{\max\{\rho_i, \rho_j\}}{\min\{\rho_i, \rho_j\}} \tag{5}$$

In a set of *K* sequences, the duration between two primary synchronization points (λ) is determined by the maximum frame rate and the least common multiplier of (5) for all pairs of sequences.

$$\lambda = \rho_{\max} \bullet LCM\left\{frames(\rho_i, \rho_j) \mid \forall ij \text{ s.t. } 1 < i < j < K\right\} \tag{6}$$

We coin the term *primary synchronization period*, denoted here by the symbol λ, to be the time between these events. If CCS={f_1, f_2, … , f_K}, is a primary synchronization point from *k* video sequences, then (f_1+ *d*λ, f_2+ *d*λ, …, f_k+ *d*λ) for all integers *d*, such that f_i + *d*λ falls within the individual sequences local time line are as well.

In practice however, we do not always have perfectly synchronized video sequences as shown in Figure 1. We can see in Figure 2, for sequences that are slightly out of sync, that the offset is minimal at the primary synchronization points. The primary synchronization points will always be anchored around full frames of the sequence with the slowest frame rate. By selecting a full frame from the slowest sequence as the anchor to synchronize around, we are reducing the amount of work necessary in finding a putative synchronization frame set and increasing our ability to find a primary synchronization point. Furthermore, this offset has a maximum bound for any secondary synchronization point. The *full frame synchronization offset* is the time difference between the actual frame capture and the exact synchronization time as shown in Figure 2. There is a maximum bound for this value that is based on the camera configuration and we explore this bound next.

2.2 Bounds of Secondary Synchronization Points

Given two imperfectly synchronized video sequences S_i and S_j with frames rates ρ_i and ρ_j, the maximum full frame synchronization offset is half the maximum difference between two frame captures of the higher frame rate sample. As we can see in Figure 2 for any pair of sequences, a frame in the slower frame rate sample straddles two frames of the higher frame rate sample, and thus full frame synchronization will be with the closest frame, in time, of the higher rate sample. For two video sequences (S_i and S_j), the quality of full frame synchronization is bounded by:

$$offset(S_i, S_j) = \frac{\min\left\{\frac{1}{\rho_i}, \frac{1}{\rho_j}\right\}}{2} \tag{7}$$

For K video sequences, the error is bound to a maximum error defined by (7) for all camera pairs and is characterized by:

$$\Delta = \max\{offset(S_i, Sj)\}\ \forall ij \text{ s.t. } 1 < i < j < K \tag{8}$$

It turns out that the maximum error will always be between the slowest and the 2nd slowest video frame rates, and thus for N cameras, Δ can be easily determined using (7) and the two slowest frame rates, or ½ the 2nd slowest frame rate.

Lemma 2.1 *For K video sequences, the maximum full frame offset is bound by half the 2nd slowest camera period.*

Proof: Let ρ_1, ρ_2 and ρ_3 be the three slowest frame rates from K sequences such that: $\rho_K < \ldots < \rho_3 < \rho_2 < \rho_1$. For all sequence pairs (1,2)(1,3),(2,3), the offsets defined by (7) are $½\rho_2$, $½\rho_3$, and $½\rho_3$ respectively. Since ρ_2 is greater than ρ_3, $½\rho_2$ is greater than $½\rho_3$. As ρ_3, ρ_2 and ρ_1 are the three slowest rates, any other frame rate ρ_i (i>3) from the sequence is less than ρ_3 and in an application of (7) resulting in $½\rho_i$ which is less than our largest value $½\rho_2$. Therefore, the error is bounded by $½\rho_2$, half of the 2nd slowest frame rate. □

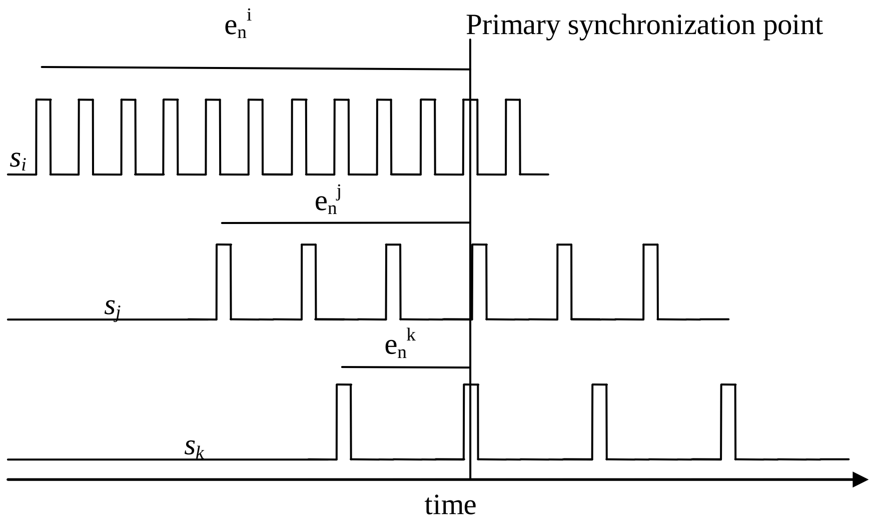

Figure 3: Video sequences with respect to a universal time line exhibiting phase shift offsets.

2.3 Universal Time and Synchronization

The functions given by (1) and (3) are generalizations that transform a video frame number into a universal time reference. Given two frames from a single video sequence S_i, the amount of time that elapses between frame f_m and f_n, n>m, is $(f_n-f_m)*\rho_i$. We let e_n^i represent the amount of time that has elapsed between the first frame and the nth frame in sequence *i*. Specifically, the *n*th frame (f_n) in sequence *i* (denoted f_n^i) will be taken at elapsed time e_n^i and is given by the following equation:

$$e_n^i = f_n^i \cdot \rho_i \tag{9}$$

which should be recognizable as the typical linear time, frame, frame rate relationship.

Because the sequence start time is the beginning of the sequence, we have a simple linear relationship between the frame rate and the frame number. However, since we want to synchronize video cameras that were not necessarily started at the same point in time, it is necessary to determine the elapsed time within the context of a universal timeline, and not simply within the time line of the single sequence itself. We can now specify the exact nature of the function described in (1) and (3) for the *nth* frame of the ith sequence on a universal time line by:

$$T_i(f_n^i)=e_n^i+s_i \tag{10}$$

where the start time of the ith sequence, is at some offset s_i from the universal start time. This offset in the universal time line represents a *phase shift*. As we see in Figure 3, three cameras started at different points in time have different phase shifts with respect to the universal time line. We see that there are phase shifts s_i, s_j, and s_k that correspond to the differences in time for which the cameras started capturing video sequences i, j, and k.

Given multiple video sequences, the synchronization consists of the frame numbers that were taken at the same instant in universal time, within the known bounded error Δ given in (8). For 3 video sequences *(i, j, k)*, the universal time line obeys the synchronization equality:

$$T_{universal} = e_n^i+s_i = e_n^j+s_j = e_n^k+s_i \tag{11}$$

The problem of synchronization is now, in fact, two fold. 1) finding the inter-sequence times (*e*) where a synchronization point occurs and 2) solving for the universal time phase shifts (*s*). In practice, we can impose the constraint that s_1 be set to universal time 0 and base our phase shift values on a time frame dictated by camera start events. We can determine the camera start order by examining the elapsed sequence times in the order of highest to lowest.

The goal of synchronization is now to solve (or minimize the error) for the synchronization equality (11). This can be done on an integral frame basis, knowing that we can only be accurate to within the time frame given by (8), or it can be solved exactly by allowing sub frame accuracy determination in equation (11). If we choose to only support integral frame numbers, equation (11) has constraints on the determination of the inter-sequence times that account for the maximum error Δ.

$$\begin{aligned}
\left|(e_i + s_i) - (e_j + s_j)\right| &\le \Delta \\
\left|(e_j + s_j) - (e_k + s_k)\right| &\le \Delta \\
\left|(e_i + si) - (e_k + s_k)\right| &\le \Delta
\end{aligned} \tag{12}$$

Additional cameras are a simple extension of the equality from (11) and the constraints from (12). Primary synchronization points minimize the constraints given by (12), and in practice should be sought.

The e’s are solved by finding a primary synchronization point where each frame was taken at the same moment in time of the same scene, and the s’s by setting s_1 to be zero, therefore becoming the universal start time, and using algebraic manipulation to solve for the remaining.

3 Recovery of the synchronization

We next examine the recovery of the synchronization from a set of cameras. We exploit the core idea that the camera geometries can be computed from static background features that are relatively constant throughout the capture process and the moving objects will only agree with the camera geometries at one point in time. Because we are utilizing multiple stationary video cameras, we can use the fundamental matrices [9] and the trifocal tensor [10] of the three views to determine the synchronization offsets. There is only one instant in time where all moving and non-moving features will have perfect consensus on the camera geometries, and this is when the moving objects are captured at the same instant in time. When foreground objects agree with the geometry computed from background features, these frames are full frame synchronized. Moreover, the best geometric support will come from a primary synchronization point.

Clearly a pure brute force method (using all frame permutations) of finding the foreground objects that support our computed camera geometry is not an option since the combinations are exponential to the number of cameras. With a maximum synchronization offset of just 30 frames (±15), a three camera system will yield 27,000 combinations to be tested unless we take advantage of anchoring around a frame in the slowest frame rate, we end up with 900 combinations to try. This still remains computationally intense. In general, the number of combinations required to perform such a computation for an N camera sequence with a maximum offset of *max_offset* frames, anchored around 1 frame from the slowest sequence, would be:

$$max\_offset^{\,N-1} \tag{13}$$

We alleviate the need to perform these brute force computations by working with camera triples and creating a virtual image that embeds the dynamic object trajectories for each camera view. We track features from each of the camera views and use the pre-computed camera geometries to determine the synchronization. When the camera geometry and the object trajectories agree, we can quickly compute the frames with maximal geometric consensus; and therefore generate the synchronization offsets. We use a four step system: 1) compute the camera geometry from static background features, 2) generate trajectory images for each sequence from moving foreground objects, 3) Narrow down trajectory images via curvature points and lemma 2.1, 4) Refine the selection via consensus to single frame accuracy and via pure geometric support to sub frame accuracy. We detail each of these steps next.

3.1 Computing Camera Geometry Using Background Features

Because the cameras are static, and the features considered in the background are also stationary, we can select any frames as candidate frames so long as they minimize the effect of the moving objects. There are a variety of ways to achieve this, from a brute force examination of the frame data using some difference metric, to a user selected set of frames. Selecting static features that do not change from frame to frame, i.e. background subtraction, or utilizing optical flow methods to remove pixels that are not static, simply adds computational overhead that is practically not necessary.

Selecting frames that are relatively close to the synchronized frames should be avoided in this step to prevent outliers from being included, resulting in a degenerate computation of the camera geometry. However, due to the large ratio of frames to cameras, and this will be true in most practical cases, we can simply sample frames from each camera sequence so that they are well distanced in time.

Once the background features have been selected, they are used to first compute information about the camera geometry. The fundamental matrices F_{12}, $F_{13,}$ and F_{23}, and the trilinear tensor, T_{123}, are required from a 3 camera system and can be computed robustly using techniques outlined in [9][10]. Furthermore, one can simply use the tensor alone since F_{12}, F_{13} and F_{23} can be derived from T_{123}. We use the software presented in [11] for our experiments which presents a work flow to compute both the fundamental matrix and the trilinear tensor.

3.2 Generating the Trajectory Images

We utilize a feature tracking mechanism to generate the trajectory images. As we track features over time, we associate the current frame number to the position within the trajectory image. The feature tracker we use is based on the work of Lucas and Kanade in [12].

While tracking features it is possible that an extremely large object motion between frames does occur and features cannot be tracked any further resulting in smaller object trajectories. This is especially true in the case of slower frame rates. So long as the object trajectories overlap in time, the length of the trajectory bears little relevance, although longer sequences help in finding points of maximum curvature to help to ensure that trajectory correspondences exist.

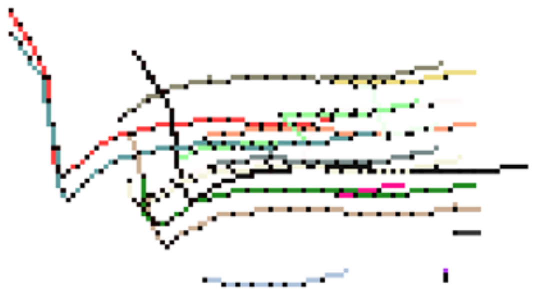

Figure 4: Trajectory image (enlarged) of a set of feature points for a 3 second interval.

In Figure 4 we present an enlarged version of the trajectory image. The image maintains a separate color for each point tracked, specifies the exact point feature position in black for each frame and linearly interpolates between the point locations.

3.3 Gross approximation of synchronization via trajectory images

We begin by performing a gross approximation of the full frame synchronization. Because we are not assuming trajectory correspondence, we must have enough interest points tracked to ensure correspondence between the 3 views. This will result in cluttered trajectory images; however we can reduce the trajectory images in the presence of points of maximum curvature.

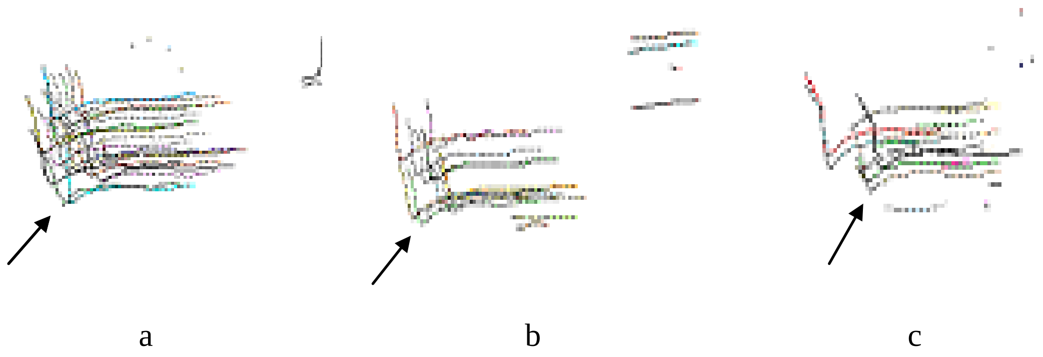

Figure 5: 3 trajectory images with coincident points of maximum curvature indicated by the arrows, notice that the shapes of the trajectories varies because motion direction is not view invariant but large changes in directions are more obvious.

A point of maximum curvature is found by examining the trajectories for similarities in overall shape. In the presence of object motion where direction is changed suddenly; the trajectories show this change at a very obvious point shown in Figure 5. In practice this allows us to get within a few frames of correct synchronization, but is never guaranteed to be exact. The reason for this is due to differing frames rates combined with perspective distortions of the fluidly moving objects causing a many-to-one, frame-to-pixel location of points of large curvature in the trajectory images.

We determine points of maximum curvature in the following way: We take the tracked points in five consecutive frames and extend the line segments formed by points in frames 1,2 and 4,5. These line segments are extended and their intersection is determined. The angles formed by the intersecting line segments will have two pairs of vertical angles that are equal and supplementary angles on each line segment. Large curvature occurs when the 4 vertical angles are roughly equal or when the two line segments are as close to perpendicular as possible. We look for curvature points by subtracting the two supplementary angles and keeping track of the global min across the entire set of trajectory points.

Once we have identified the points of maximum curvature, and implicitly the gross approximation of the synchronization, it is further refined by creating a reduced trajectory image around the point of maximum curvature. We then apply a geometric consensus stage. In cases where these large curvature points cannot be reliably found, the gross approximation stage can be omitted and we use the dense trajectory images in the geometric consensus stage. This will result in many more consensus trials being performed, as we see in Figure 6a, because the epipolar line will potentially intersect with many more trajectories causing the candidate set to be larger. In Figure 6b we show a reduced trajectory image (enlarged for viewing) for an 8 frame track after the point of maximum curvature.

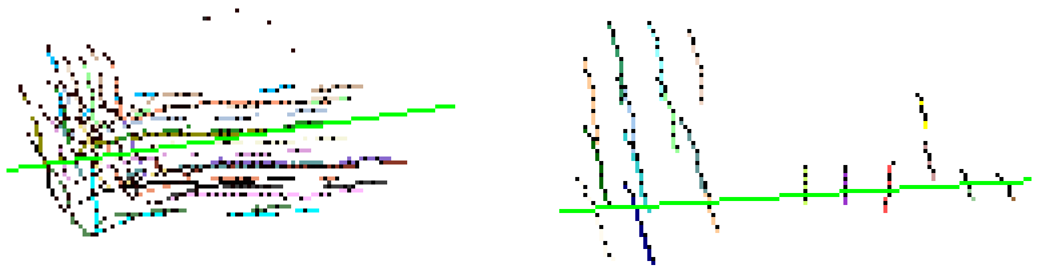

Figure 6: Epipolar line (green) and trajectories. The intersection of trajectories and the epipolar line make up the candidate set of points. (a) dense trajectory image (b) reduced trajectory image

In practice, the presence of obvious large curvature points may be quite difficult to find, especially when the motions of the dynamic objects are not under control of the application. Furthermore, the trajectories of non rigid objects have different times in which the change of motion presents itself. Although the practice of reducing the trajectories for computational efficiency is useful, it is not strictly necessary for the algorithm outlined here to successfully determine the synchronization. A closer examination of motion trajectories and the corresponding frames in the video sequences helps to show how gross approximation errors can occur. When object motion changes, especially if it is in the direction of the optical axis, there are several frames associated to a single pixel location. This situation can also occur when an objects motion remains in the epipolar plane of a camera pair presenting a many to one frame to pixel association. Furthermore, should the object motion be extremely slow, there are multiple frames associated with the feature location and therefore a higher error in the gross approximation will result. To confirm, we tracked a target over 15 frames as it moved towards a camera along the optical axis. The result was a many to one, frame to pixel location association that made exact synchronization of the frames impossible using that objects trajectory.

3.4 Refinement of synchronization via maximal geometric consensus

During the creation of the trajectory images, we associate a list of frame numbers to each tracked pixel position of the dynamic objects in the trajectory image. We can now effectively compute the synchronization to sub-frame accuracy using the camera geometry and the trajectory images. We do this by selecting a point *x* in any trajectory in the first image. We then compute the epipolar line that will intersect the corresponding trajectory in the second trajectory image. The epipolar line will also cross other trajectories in the second image, and we use the intersections of the epipolar line and the trajectories to create a candidate set of matching points. As shown in Figure 7, these candidate frames can be computed to sub-frame accuracy as the intersection of the trajectory line joining two point positions in adjacent frames. For each point in the candidate set, tensor transfer is applied along with the first point to compute a third point in the third trajectory image. The computed 3rd point (via tensor transfer) is used to find the closest trajectory point. This closest point is also computed to sub frame accuracy as shown in Figure 7.

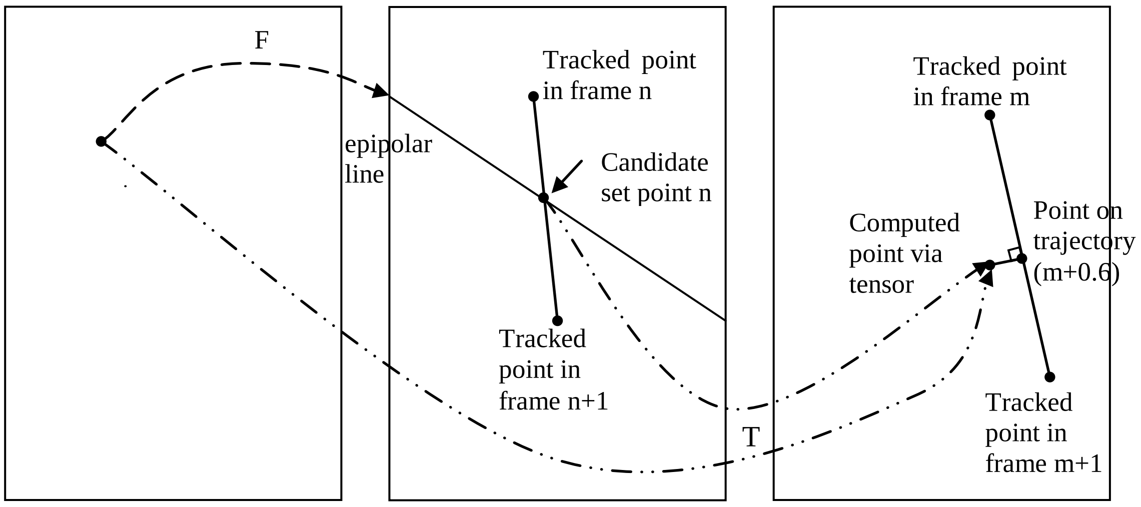
F
epipolar line
Tracked point in frame n
Candidate set point n
Tracked point in frame n+1
T
Tracked point in frame m
Computed point via tensor
Point on trajectory (m+0.6)
Tracked point in frame m+1

Figure 7: Computed frame values to sub-frame accuracy via epipolar and tensor transfer for 3 images. F represents the epipolar transfer and T represents the Tensor transfer functions.

Once the 3 points have been associated to their respective trajectories, the nearest full frames are selected for a consensus trial. We compare a window around the exact tracked point position for each of points using normalized cross correlation to determine whether or not the three points are similar enough to perform the consensus trial. When the points agree, we then verify that a variety of features in the selected frames support the given geometry by generating corner features and performing matching that is guided by the pre-computed geometries [11]. The putative synchronized frame set with the highest consensus overall is selected as the synchronized CCS.

In practice, one should always start with the slowest camera when selecting the first point *x* because it will help to reduce the number of potential consensus trials as the two slowest cameras define the largest possible error in synchronization time as discussed in section 2. Moreover, attempts should be made to use a reduced trajectory image so that the number of many-to-one frame-to-pixel associations is minimized and therefore the size of the candidate set of matching points will be smaller. If any of the computed points lie on overlapping tracks, we test all the possible combinations of frames. For a point *x* whose epipolar line intersects X trajectories and the subsequent tensor transfer is equidistant to Y trajectories results in a consensus trial for X·Y frame triplets, a drastic reduction in consensus trials compared to (13) which is the number of trials for a smart brute force method as discussed previously.

3.5 Synchronization in the face of erroneous geometries.

Invariably there is error. The two main sources of error in our system occur from erroneous computation of geometries and erroneous localization during tracking. Even a single pixel displacement of the epipolar line will result in an incorrect location of the intersection of trajectories and the epipolar line, which will result in inexact time localization. In the presence of larger inaccuracies, it is beneficial to examine a broader range of frames when operating our consensus trials. We do this by modifying the generation of the candidate set to include multiple frames from each track that intersect with the epipolar line. We examine a number (ε) of complete frames on either side of the epipolar line. The optimal epsilon is a function of the frame rates and guarantees us to search at least one primary synchronization point due to (6), (7) and lemma 2.1. For each sequence, S_i, epsilon is:

$$\varepsilon = \left\lceil \frac{\lambda}{2\rho_i} \right\rceil \qquad (14)$$

This will result in $(2\varepsilon+1)^{N-1}$ consensus trials for N cameras. For our experimental trials, we set ε to be 2 as computed from (14). In practice, epsilon should be globally set to the largest computed epsilon for each sequence S_i.

3.6 Algorithm Review: Synchronize

Input: 3 video camera view sequences
Output: frame numbers for 3 synchronized frames
External Functions:
Projective_Transfer(p1,p2) performs tensor transfer between p1,p2 returns point p3
Xcorr(f1,p1,f2,p2,f3,p3) performs pair-wise normalized cross correlation for 3 points (*px*) in 3 frames (*fx*)and returns the minimum correlation of all 3 pairings.

```
1  Select frames that a form static-CCS
2  Compute fundamental matrices($F_{12}$ and $F_{23}$) & triliner tensor($T_{123}$)
3  Generate trajectory images at 3 second intervals
4  Find and match inflection points in trajectory images
5  Generate reduced trajectory images around inflection points
6  Select a point $x$ from $f_1$ on a trajectory image 1
7  l = $F_{12}x$ (compute Epipolar line)
8  Compute the candidate set of points ($x_{ci}$) i.e. the intersections of the
   trajectories and the epipolar line l
9  For each point pair ($x$, $x_{ci}$) do
10   $x_t$ = Projective_Transfer($x$, $x_{ci}$)
11   Compute frame number $f_2$ from $x_{ci}$ to sub-frame accuracy (Fig 13)
12   Compute frame number $f_3$ from $x_t$ and the nearest trajectory to sub-frame
          accuracy (Fig 13)
13   If (Xcorr( $f_1$,x, $\lfloor f_2+0.5 \rfloor$,$x_{ci}$, $\lfloor f_3+0.5 \rfloor$, $x_t$) < threshold)
14      Perform guided matching using frames $\lfloor f_1+0.5 \rfloor$,$\lfloor f_2+0.5 \rfloor$,$\lfloor f_3+0.5 \rfloor$ and
             the geometry computed in (2)
15      If consensus feature count is maximal then
          Save $f_1$, $f_2$, and $f_3$
16 Endfor(9)
17 Return $f_1$, $f_2$, and $f_3$
```

4 Experimental Results

We have applied the algorithm to various sequences, both synthetic and those captured by a variety of different cameras of varying quality and frame rates. We compare to known ground truth where possible, while in the other cases, we compare to our hand selected ground truth.

4.1 Synthetic Data

In our synthetic data set, we have a series of static 3D points in a variety of positions. Our dynamic 3D points are the vertices of a cube which we move before constructing each frame in our sequence. The frame rates are the same and constant for each generated sequence, and the sequences are perfectly synchronized. This scenario represents a system of cameras with identical frame rates that are full frame synchronized. The offsets were set to be 0, 5 and 10 frames respectively. In this case, the motion of the cube was arranged so that the vertices of the cube projected to a unique pixel after each motion. This resulted in a trajectory image with a one-to-one pixel/frame number association. The trifocal tensor was derived from the projection matrices and the fundamental matrices were subsequently derived from the tensor. The trajectory images were generated using the projected positions of the 3D vertices of the cube. Due to the simplistic motion, there were no curvature points in the trajectory image, thus application of the consensus algorithm was all that was necessary. Under these ideal conditions, the synchronization was computed exactly to be frame deltas 0, 5 and 10 respectively.

In our next synthetic example, we configured the system to have the same constant frame rates for each generated sequence. In this example, the sequences are not perfectly synchronized and frame deltas of 0, 5.25 and 10.75 were used to represent a system similar to Figure 2. In this case, the motion of the cube was arranged so that the vertices of the cube projected to a unique pixel after each motion and that for each full frame tick, the points moved exactly 4 pixels. This resulted in a trajectory image with a one-to-one pixel/frame number association and allowed us to easily determine the sub-frame offsets. Under these conditions, the synchronization was computed exactly to be frame deltas 0, 5.25 and 10.75 respectively.

4.2 Real Data

In the following experiments, we used various digital cameras with video capture capabilitiesand varying frame rates and resolutions. In our first experiment, we used a system of 3 cameras that grabbed frames on a synchronized basis, we then offset the video sequences by 5 and 10 frames for the second and third cameras respectively to be used as our ground truth. This scenario represents a system of cameras with identical frame rates that are full frame synchronized and the capture process was started simultaneously (as in Figure 1). As we can see in Table 1 the computed full frame synchronization is correct, however due to minor errors in computed geometry, the exact synchronization exhibits the minor errors. The error falls well within the expected maximum error of half a frame.

Camera	Gross Approx. (frame #)	Exact Sync (frame #)	1st Primary Sync Point (sec)	Exact Time *ei* (sec.)	Universal Time Shift *si* (sec.)	First Full Sync Frame (frame #)	Ground Truth (frame #)
1	141	141	0	0	1.994	0	0
2	146	146.05	5.05	1.010	0.984	5	5
3	151	150.97	9.97	1.994	0	10	10

Table 1: Synchronization Results using real cameras with known synchronization parameters

In the first two examples, we used a target as the moving object in order to assure corresponding points would produce corresponding trajectories. In both of these examples, the target was moved such that an obvious change of direction (curvature points) occurred. These very sharp points allow the gross approximation method to achieve very close results to the synchronized frames with maximal consensus. In Table 2 we see that the gross approximation in the presence of points of maximum curvature is accurate within a few frames. In Table 3, we confirm that the time difference falls within the expected bounds given the ground truth.

Camera	Frame Rate (sec)	Gross Approx. (frame #)	Gross Aprox. Time (sec)	Gross Aprox. Universal Time Shift (sec)	Exact Sync (frame #)	Exact Time (sec)	Exact Universal Time Shift (sec)	Nearest Full Frame (frame #)
1	1/15	127	8.467	7.533	126.75	8.45	7.45	127
2	1/15	228	15.2	0.80	229.33	15.289	0.611	229
3	1/10	160	16	0	159	15.9	0	159

Table 2: Synchronization Results using real cameras with hand selected ground truth.

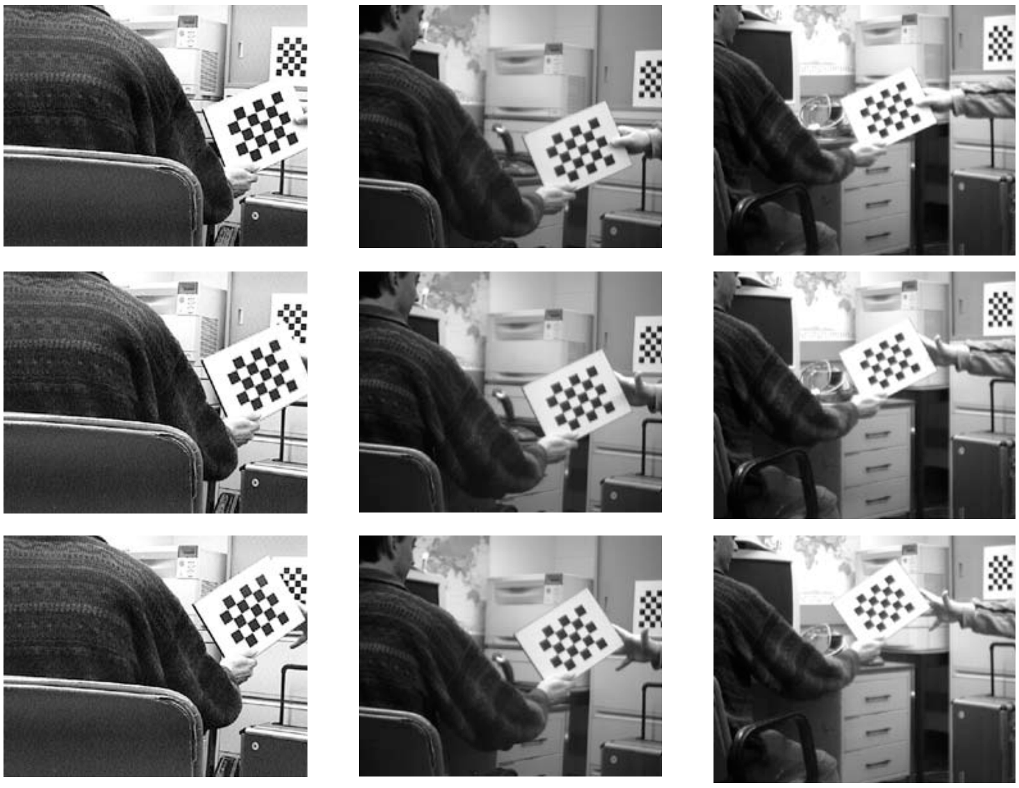

Figure 8: 3 synchronized (rows), consecutive frames from 3 video sequences (columns)

Ground Truth (user selected)	Time	Difference from computed time
127	8.467	0.017
229	15.267	0.022
159	15.9	0

Table 3: Synchronization Results compared to user selected synchronizations

The targets, while helpful in ensuring the accuracy by allowing multiple corresponding trajectories causes the algorithm to force many more consensus trials because of the feature proximity. With fewer corresponding features being tracked, there are fewer points in the candidate sets and thus fewer geometric consensus trials. However, in order to ensure accurate synchronization, we need to ensure that at least one corresponding feature is sufficiently tracked in all 3 cameras sequences.

More results from another experiment outlined in Table 4, support the previous findings that the gross approximation, in the presence of curvature points is accurate to within a few frames.

	Gross Approximation	Exact Synchronization	Full Frame Sync.	Full Frame Ground Truth
Camera 1 frame	165	167	167	167
Camera 2 frame	212	213.66	214	214
Camera 3 frame	170	170.80	171	170
Total Frame Error	4	1.13	1	N/A

Table 4: Synchronization Results

In order for the exact computation of synchronization via geometric consensus to be effective, there is the requirement of corresponding features being successfully tracked. The targets, seen in Figure 8 help to ensure that corresponding features are indeed tracked. As a result, the trajectory images are quite feature-rich. In contrast, without the use of targets, the trajectory images may be quite sparse due to the static background features being selected automatically over the moving object features. In our final example, we abandon the use of targets and look to automatically track features on dynamic objects.

Camera	Frame Rate	Gross Approx - imation	Time (s)	Universal Time Shift (s)	Exact Sync	Exact Time (s)	Universal Time Shift (s)	Nearest Full Frame	Time
1	1/15	244	16.267	11.133	242.80	16.187	11.213	243	16.20
2	1/15	302	20.133	7.267	301.50	20.100	7.300	302	20.13
3	1/10	274	27.400	0	274	27.400	0	274	27.40

Table 5: Synchronization Results for the 2nd experiment

In our next experiment, we set up 3 different cameras and manually started each of the cameras. The ground truth was manually selected as there was no computerized synchronization mechanism. The ground truth was selected as frames 243, 301, and 274 for cameras 1, 2, and 3 respectively. This experiment also removes the use of obvious targets and uses textured objects for tracking. Unlike our target based approach, the gross approximation became more difficult and required minor manual interventions. Because the features on the moving objects lack the contrast of the target, it was often the case that features on the moving objects would not be automatically selected into the tracked features list. A minor manual step to force feature selection in selected areas helped to generate better trajectory images. Background subtraction techniques would help to remove the need for this manual requirement.

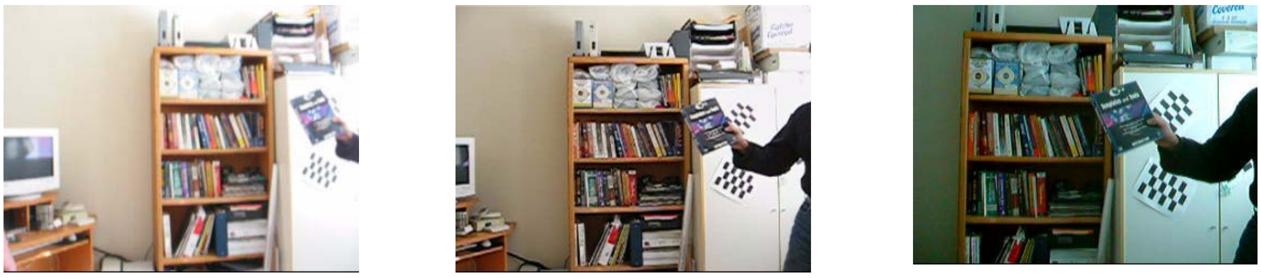

Figure 9: Selected Synchronized Frames

Again, the results shown in Figure 9, listed in Table 5 and Table 6 fall within the expected maximum error. However, in this specific case, the epipolar line fell exactly halfway between the two points in sequence 2. Our decision to move up, rather than down affected the accuracy and caused a single frame error in the computed full frame synchronization, but does not change the accuracy of the exact computation. A minor anomaly worth noting, is that in this example, the hand selected ground truth value for sequence 2 (301) may be incorrect. All three methods, gross approximation, exact and robust all agree with a value of 302. When selecting the ground truth, we had several people examine the frames and come to a consensus of the frame numbers that they believed were synchronized.

Ground Truth (user selected)	Time	Difference from exact computed time
243	16.200	0
301	20.067	0.033
274	27.400	0

Table 6: Synchronization Results for 2nd experiment

6 Conclusions

In this work we presented a complete theoretical formalization of the general problem of temporal synchronization with heterogeneous camera setups and without scene constraint in the continuous and discrete time domains. As well, we detailed a novel method for synchronizing multiple video sequence using feature tracking and geometric consensus. The proposed method allows for the least constraints being placed on the camera setup and the scene being viewed. The method provides two levels of accuracy by using a two step process of grossly approximating the frame synchronization followed by a refinement step that examines the selected frames for their consensus with the camera geometry. The method has been successfully used on both synthetic data and real data with substantial noise, differing frame rates, differing resolutions, and varying levels of initial synchronization. Even in the presence of erroneous geometries, it is possible to get very close synchronization results at the cost of performing more consensus trials to account for the geometric inaccuracies. Open problems include accurate detection of corresponding curvature points that are not so grossly evident and the automatic detection of corresponding trajectories apriori. While these values are implicitly computed by the method, thus known apostori; knowing them apriori would result in a reduction of the number of times the consensus step (the largest consumption of time). As well, dealing with situations where the trajectories and the epipolar lines approach coincidence, i.e. do not have unique intersections points is an interesting area that needs further investigation.

References

[1] G. Xu, and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition.* Kluwer Academic Publishers. 1996.

[2] H. Huang, C. Kao, Y. Lin, Y. Hung, Yi-Ping, "Disparity-based view interpolation for multiple-perspective stereoscopic displays", *Proceedings of SPIE Vol. 3957, Stereoscopic Displays and Virtual Reality Systems VII,* p. 102-113, 2000

[3] Lily Lee, Raquel Romano, Gideon Stein, "Monitoring Activities from Multiple Video

Streams: Establishing a Common Coordinate Frame", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Special Section on Video Surveillance and Monitoring, 22(8), 2000

[4] J. Kang, I. Cohen, G. Medioni. "Continuous multi-views tracking using tensor voting", *Proceedings of Workshop on Motion and Video Computing, 2002.* pp 181- 186

[5] Y. Caspi and M. Irani. "Alignment of non-overlapping sequences". *Proceedings of International Conference on Computer Vision*, Vancouver, BC, pp 76-83, 2001.

[6] S. Kuthirumal, C.V. Jawahar, and P.J. Narayanan. "Video frame alignment in multiple views". *Proceedings of International Conference on Image Processing*, Rochester, NY, 2002.

[7] C. Rao, A.Gritai, M. Shah. "View-invariant Alignment and Matching of Video Sequences", *In Proceedings of International Conference on Computer Vision*, pp 939-945, 2003.

[8] P. Tresadern and I. Reid. "Synchronizing Image Sequences of Non-Rigid Objects", *In Proceedings of British Machine Vision Conference*, 2003

[9] P.H.S. Torr and D.W. Murray, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix", *International Journal of Computer Vision*, vol 24 pp 271-300, 1997

[10] P.H.S. Torr and A. Zisserman. "Robust Parameterization and Computation of the Trifocal Tensor". P*roc. British Machine Vision Conference,* pp 655-664. 1996.

[11] A. Whitehead and G. Roth. "The Projective Vision Toolkit", *In Proceedings of Modelling and Simulation*, pp 204-209, May 2000

[12] Carlo Tomasi and Takeo Kanade. "Detection and Tracking of Point Features". *Carnegie Mellon University Technical Report CMU-CS-91-132*, 1991.

[14] B. Zitova and J. Flusser. "Image registration methods: a survey". Image and Vision Computing, vol 21, pp 977-1000, 2003

[15] R. Carceroni, F. Padua, G. Santos, and K. Kutulakos. "Linear sequence-to-sequence alignment". In Proc. IEEE CVPR, pp 746-753, 2004

[16] T. Tuytelaars and L. Van Gool, "Synchronizing Video Sequences", In Proc. IEEE CVPR, pp 762-768, 2004

[17] P. Sand and S. Teller, "Video Matching". ACM Trans. Graph., vol. 23(3), pp 592-599, 2004

[18] I. Laptev, S. Belongie, P. Perez, and J. Wills. "Periodic motion detection and segmentation via approximate sequence alignment". In Proc. IEEE ICCV, pp 816-823, 2005

[19] Y. Wexler and D. Simakov. "Space-time scene manifolds". In Proc IEEE ICCV, pp 858-863, 2005.

[20] S. Sinha, M. Pollefeys, "Synchronization and Calibration of Camera Networks from Silhouettes", International Conference on Pattern Recognition (ICPR), 2004

[21] Y. Caspi, D. Simakov, and M. Irani. Feature-based sequence to sequence matching. Int. Journal of Computer Vision, 68(1):53–64, 2006