Projective Space Temporal Synchronization of Multiple Video Sequences

Anthony Whitehead, Robert Laganiere, Prosenjit Bose

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 of a screenplay. However, in some situations such a 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.

Figure 1: Camera Sync Slate, a.k.a. clapper board[1]

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*. 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, i.e.

$$T_i(f) : N \rightarrow R \tag{1}$$

The synchronization problem can now be expressed as finding a set of frames numbers, $f_1, f_2, \ldots, f_k$, one from each video sequence, such that the *synchronization equality* $T_1(f_1) = T_2(f_2) = \ldots = T_k(f_k)$ holds. Such a set of frames that exactly solves the synchronization equality is said to be in *perfect integral synchronization*.

[1] Picture from Filmtools, Burbank CA. http://www.filmtools.com/ Used with permission.

However, due to possible minute variations in camera start times and variations in frame rates, perfect integral synchronization does not generally exist. In such a case we search for a set of frames whose pair-wise difference with respect to the synchronization equality is minimized. We provide exact bounds on the value of this pair-wise difference in the next section. If we remove the restriction of integral frame numbers, the frame-time function maps frame values (integral and non-integral) to a point in time, i.e:

$$Ti(f) : R \rightarrow R \tag{2}$$

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, $f_1, f_2, \ldots, f_k$, one from each video sequence such that synchronization equality $T_1(f_1) = T_2(f_2) = \ldots = T_k(f_k)$ holds.

In summary the synchronization problem is:

1. …referred to as the *full frame synchronization problem* when restricted to integral frame numbers, and seeks to minimize the pair-wise differences of the *synchronization equality*. i.e. $|T_i(f_i) - T_j(f_j)|$ is minimal for all sequence pairs i,j.
2. …referred to as the *exact synchronization problem* when unrestricted, and seeks to exactly solve the *synchronization equality*.

We 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 [2], however, since the processing of large volumes of video data is becoming tractable, recent work has investigated the problem of synchronizing 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. 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|. 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 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.

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. Furthermore, as synchronization is simply a means to an end, they examine the full frame synchronization problem rather than the exact synchronization problem. 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 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 characteris-

tic such as a pendulum, nor can the motion be directly along the optical axis of one of the cameras. Any camera count over three can be handled by our method on an overlapping basis.

This work is organized as follows: Section 2 formalizes the problem of synchronizing video sequences and introduces terminology. Section 3 outlines our proposed method that includes 1) Computing camera geometries, 2) Generating trajectories, 3) Using inflection points to grossly approximate the synchronization, and 4) using the computed geometries to compute the exact synchronization. Section 3 ends with an adaptation to handle errors in the computed geometries. In Section 4, we perform experiments with our proposed method and present results. In Section 5, we examine some practical issues and finally we draw some conclusions.

2 Problem Formalization

We begin by formalizing the synchronization problem, and follow up by introducing terminology used in the description of the proposed solution.

The synchronization problem

We first examine some properties of the relationship between multiple video sequences and specify terminology. We let $F_i: R \rightarrow \{0,1\}$ be the *frame-capture function*. $F_i(x) = 1$ if at time x, a frame in video sequence i is being captured and $F_i(x) = 0$ otherwise. Close examination reveals that the frame-capture function is periodic in nature and therefore the model for video capture and synchronization we use is wave based, not linear as one might expect. The time between peaks in the function F_i is known as the *period* (in wave mechanics terminology) and what is commonly referred to as the *frame rate* (ρ_i), is actually the *frequency*. Recall that frequency and the period are inversely related. Figures 2 and 3 plot the function F_i. The peaks (value 1) occur when a frame is captured 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.

Defintion: A *primary synchronization point* is a single point in time that minimizes the difference between the exact synchronization function (2) and the full frame synchronization function (1) for all sequences. i.e. The frame numbers that satisfy the synchronization equality and minimize the difference given by (3).

Formally, the difference between full frame and exact synchronization is given by:

$$\left|Ti(fi) - Ti(\lfloor fi + 0.5 \rfloor)\right| \tag{3}$$

Any synchronization time that does not minimize the difference (3) is termed a *secondary synchronization point* i.e. any non primary synchronization point. As we see in Figure 2, 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. i.e the difference given by (6.3) is 0, and as we shall discuss next, perfectly synchronized video sequences are rare in practice.

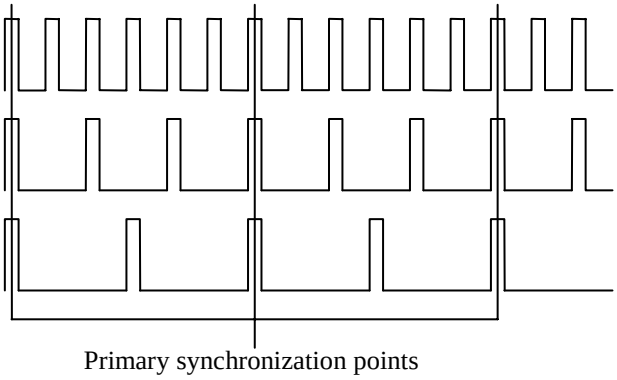


Figure 2: Perfect integral synchronized video sequences with varying frame rates showing primary synchronization points. .

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 sync points for two sequences is a function of the frame rates given by:

$$frames(\rho_1, \rho_2) = \frac{\max\{\rho_1, \rho_2\}}{\min\{\rho_1, \rho_2\}} \quad (4)$$

The time between two primary synchronization points (λ) is determined by the maximum frame rate and the least common multiplier of (4) for all pairs of sequences.

$$\lambda = \rho_{\max} \bullet LCM\{frames(\rho_i, \rho_j) \mid \forall ij \text{ s.t. } 1 < i < j < N\} \quad (5)$$

We coin the term *primary synchronization period,* denoted by the symbol λ, to be the time between these events. If $(f_1,f_2,\ldots,f_k)$ is a primary synchronization point from *k* video sequences, then $(f_1+ d\lambda, f_2+ d\lambda, \ldots, f_k+ d\lambda)$ for all *d* such that $f_i + d\lambda$ falls within the individual sequences local time line.

In practice however, we do not always have perfectly synchronized video sequences as shown in Figure 2. Instead, we have a slight synchronization offset. We can see in Figure 3, for sequences that are slightly out of sync, that the offset is minimal at the primary synchronization points. Furthermore, this offset has a maximum bound for any secondary synchronization point. We explore this bound next.

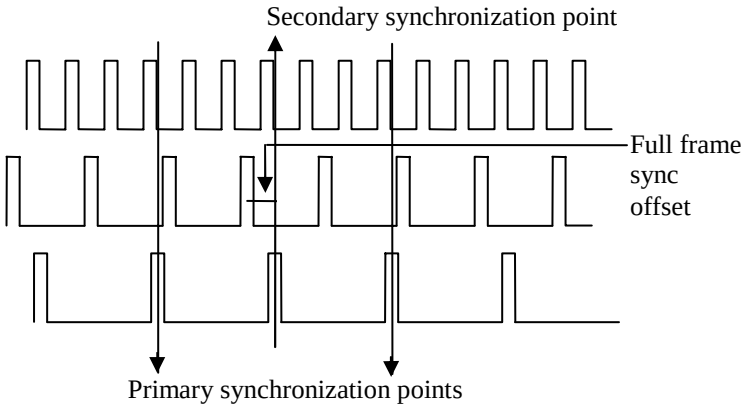


Figure 3: Imperfectly synchronized video sequences with varying frame rates showing primary and secondary synchronization points.

Given two imperfectly synchronized video sequences, the maximum full frame sync offset is half the maximum difference between two frame captures of the higher frame rate sample. As we can see in Figure 3 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 syn-

chronization will be with the closest frame, in time, of the higher rate sample. For two video sequences, the quality of full frame synchronization is bounded by:

$$offset(V_1, V_2) = \frac{\min\left\{\frac{1}{\rho_1}, \frac{1}{\rho_2}\right\}}{2} \tag{6}$$

For N video sequences, the error is bounded to a maximum error defined by (6) for all camera pairs and is characterized by:

$$\Delta = \max\{offset(V_i, Vj)\} \ \forall ij \text{ s.t. } 1 < i < j < N \tag{7}$$

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 (6) and the two slowest frame rates.

Lemma 2.1 *For N 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 N sequences such that: $\rho_N < \ldots < \rho_3 < \rho_2 < \rho_1$. For all sequence pairs, the offsets defined by (5) are $\rho_2/2$, $\rho_3/2$, and $\rho_3/2$ respectively. Since ρ_2 is greater than ρ_3, $\rho_2/2$ is greater than $\rho_3/2$. As ρ_3, ρ_2 and ρ_1 are the three slowest rates, any other frame rate ρ_i (i>3) from the N sequence is less than ρ_3 resulting in an application of (5) resulting in $\rho_i/2$ which is less than our largest value $\rho_2/2$. Therefore, the error is bounded by $\rho_2/2$, half of the 2nd slowest frame rate. □

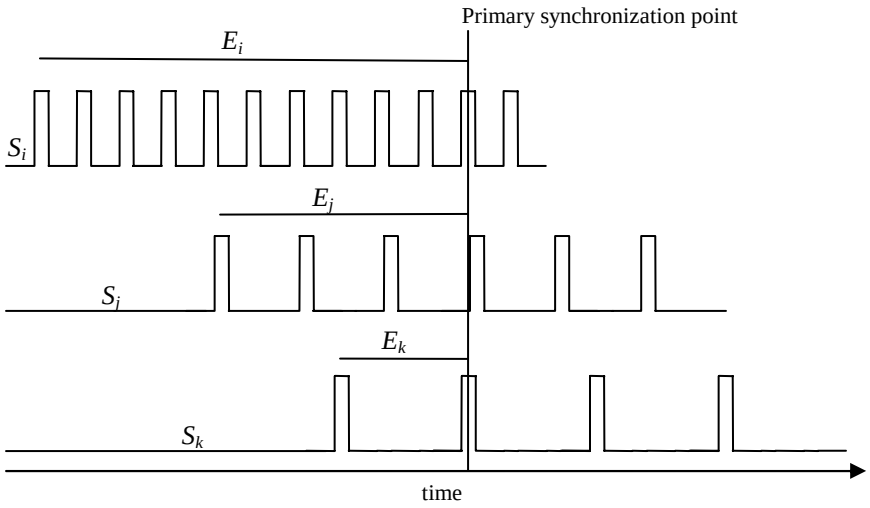


Figure 4: Video sequences with respect to a universal time line.

Given two frames from a single video sequence i, the amount of time that elapses between frame f_1 and f_2 is $(f_2\text{-}f_1)*\rho_i$. We let E_i represent the amount of time that has elapsed between the first frame and the f^{th} frame (denoted f_i) in sequence i. Specifically, the f^{th} frame (f_i) in sequence i will be taken at elapsed time E_i and is given by the following equation:

$$E_i = f_i \cdot \rho_i \tag{8}$$

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 (2) by by:

$$Ti(n_i) = E_i + S_i \tag{9}$$

where the start time of the sequence *i*, is at some offset S_i from the universal start time. This offset in the universal time line represents a *phase shift* in wave mechanics terminology. As we see in Figure 4, 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.

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 (7). For 3 video sequences *(i, j, k)*, the universal time line obeys the synchronization equality:

$$Ti(n_i) = E_i + S_i = E_j + S_j = E_k + S_k \tag{10}$$

The problem of synchronization is now, in fact, two fold. 1) finding the inter-sequence times (*Ei*) where a synchronization point occurs and 2) solving for the universal time phase shifts (S_i). 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 for the synchronization equality (10). This can be done on an integral frame basis, knowing that we can only be accurate to within the time frame given by (7), or it can be solved exactly by allowing sub frame accuracy determination in equation (10). If we choose to only support integral frame numbers, equation (10) has constraints on the determination of the inter-sequence times that account for the maximum error Δ.

$$\begin{aligned} \left|(E_i + S_i) - (E_j + S_j)\right| &\le \Delta \\ \left|(E_j + S_j) - (E_k + S_k)\right| &\le \Delta \\ \left|(E_i + Si) - (E_k + S_k)\right| &\le \Delta \end{aligned} \tag{11}$$

Additional cameras are a simple extension of the equality from (10) and the constraints from (11). Primary synchronization points minimize the constraints given by (11), 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 3D 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.

2.1 Terminology

We continue by specifying terminology that defines the core concepts behind the proposed solution. A *camera sequence* (CS) 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 N images, where each image in the subset comes uniquely from one of the N cameras. A *synchronized CCS* is a cross camera subset where each frame of the set is full frame synchronized as outlined in equation (1).

A cross camera subset is not necessarily aligned in time, we denote a CCS to be simply a selection of N frames, one from each of N camera sequences. 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.

We further sub-classify cross camera sets into dynamic-CCS and static-CCS. As their names elude, a static-CCS is comprised of those images that have the same static content (or in practice, a majority of static content). There are multiple static-CCS candidates among a set of video camera sequences. The term static-CCS is not to say that there is no dynamic motion within the frames, but rather we are interested only in the static content of each frame so that we can compute the camera geometries. A dynamic-CCS is the set of images in which we are utilizing the dynamic aspects of the cross camera set. Again this is not to say that all pixels within a set of candidate frames are moving, but rather they contain the same moving objects.

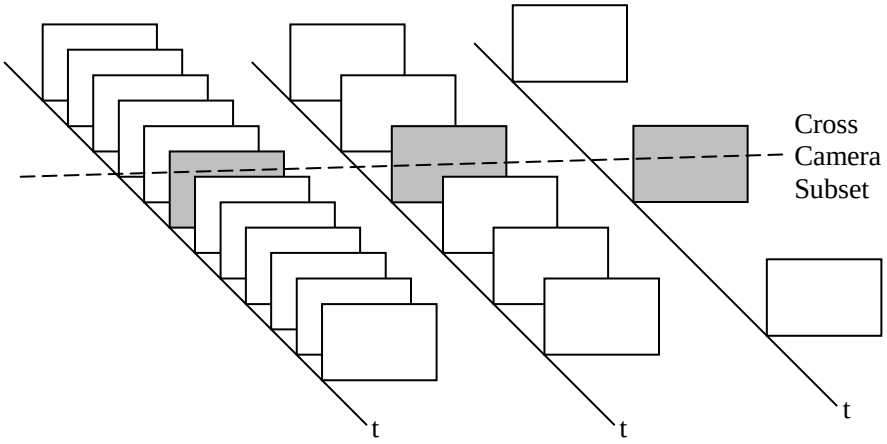


Figure 5: 3 camera sequences in a three video camera setup with varying frame rates, with a synchronized cross camera set in gray.

3 Recovery of the synchronization

Because we are utilizing multiple non-moving 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 fea-

tures will have perfect consensus on the camera geometries, and this is when the moving objects are captured at the same instant in time. When dynamic-CCS objects concur with the geometry computed with static-CCS, the frames that comprise the dynamic-CSS are full frame synchronized. Moreover, the best geometric support will come from a primary synchronization point. It is also important to note that more than one dynamic-CSS will support the geometry computed from the static-CSS; those being any dynamic-CSS that contain synchronized frames.

Clearly a pure brute force method (using all frame permutations) of finding the dynamic-CCS that supports our computed camera geometry is not an option since the combinations are exponential to the number of cameras. This would require M^N combinations to be examined, where N is the number of cameras and M is the frame count. One way to reduce the cost of the brute force method is to align groups of 3 adjacent cameras. With a maximum synchronization offset of just 30 frames (±15), a three camera system will yield 27,000 combinations to be tested. 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, would be:

$$(N-2)\cdot max\_offset^{\,3} \tag{12}$$

We alleviate the need to perform these brute force computations (even smarter brute force) by creating a virtual image that embeds the dynamic object trajectories in 2D. We utilize these virtual trajectory images to quickly determine the synchronization. The core idea we utilize is that the camera geometry and the object trajectories will concur, allowing us to quickly compute the frames with maximal geometric consensus and therefore implicitly generate the synchronization offsets. We use a basic 4 step system: 1) compute the camera geometry from a static-CCS, 2) generate trajectory images for each sequence, 3) Narrow down trajectory images via inflection points, 4) Refine the selection via consensus to single frame accuracy and via pure geometric support to sub frame accuracy.

3.1 Computing Camera Geometry from the Static-CCS

Because the cameras are static, and the features considered in a static-CCS are also stationary and 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.

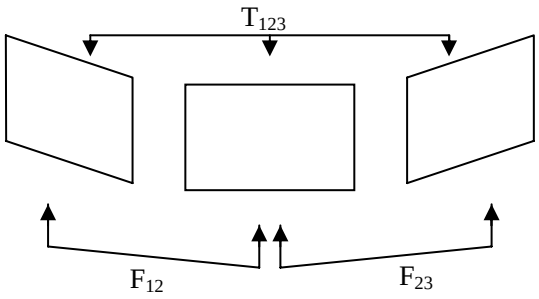


Figure 6: Required geometries for a 3 camera synchronization

Once the static-CCS has been selected, it is used to first compute information about the camera geometry. In Figure 6, we see the required geometries to compute the synchronization of three video sequences. The fundamental matrices F_{12}, 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} and F_{23} can be derived from T_{123}. We use the software presented in [11] for our experiments.

3.1.1 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 tra-

jectory image. Feature tracking is performed on the luminance channel (grey map) for the video frames. The luminance channel is computed as follows:

$$\text{Luminance} = \text{Red}*0.299 + \text{Green}*0.587 + \text{Blue}*0.114 \tag{13}$$

The feature tracker we use is based on the work of Lucas and Kanade in [12]. This work was further developed by Tomasi and Kanade in [13] of which Shi and Tomasi provide a complete description in [14].

Briefly, features are located by examining the minimum eigenvalue of a 2x2 image gradient matrix. The features are tracked using a Newton-Raphson method of minimizing the difference between the two windows around the feature points. We continue by presenting a very brief outline of the work by Tomasi et al [12,13,14].

Given a point p in an image I, and its corresponding point *q* in an image J, the displacement vector δ between *p* and *q* is best described using an affine motion field:

$$\delta = Dp + t \tag{14}$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \tag{15}$$

is a deformation matrix and t is the translation vector of the centre point of the tracked feature window. The translation vector t is measured with respect to the feature in question. Tracking feature *p* to feature *q* is simply the problem of determining the six parameters that comprise the deformation matrix D and the translation vector t. In the case of pure translation, D will be the identity matrix and thus

$$\delta = p + t \tag{16}$$

Because of this, the case of pure translation is computationally simpler and thus preferable due the higher frame rates typically found in video data. Since the motion between adjacent frames of standard video is generally quite small, it turns out that setting the deformation matrix to identity is a safe computation [14], leaving us with the translation vector being exactly the displacement vector.

The displacement vector is computed using a pyramid of resolutions because processing a high resolution image is computationally intense. The multi-resolution pyramid within the feature tracker reduces the resolution of the entire image, say by a factor of 2. Tracking occurs by tracking a features general area in the lowest resolution and upgrading the search for the exact location as it progresses back up the pyramid to the highest resolution.

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 inflection points and help to ensure that trajectory correspondences exist. In Figure 7, we have a trajectory image for 3 seconds of a video sequence along with the first and last frames from the from the trajectory sequence.

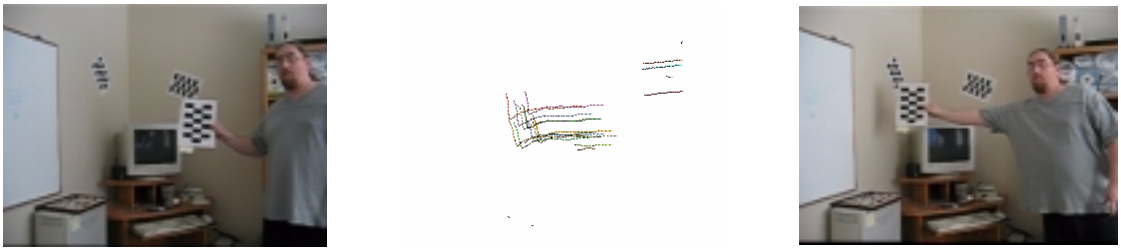


Figure 7: Initial image, trajectory image (target and head features) tracked over 45 frames, final image

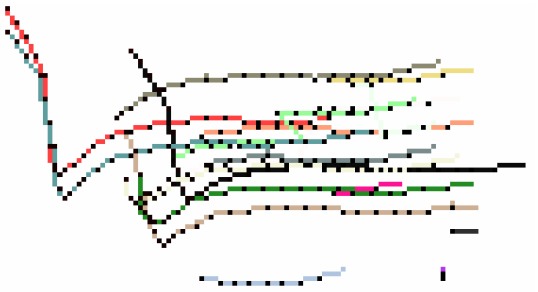


Figure 8: Trajectory image (enlarged) for a 3 second interval.

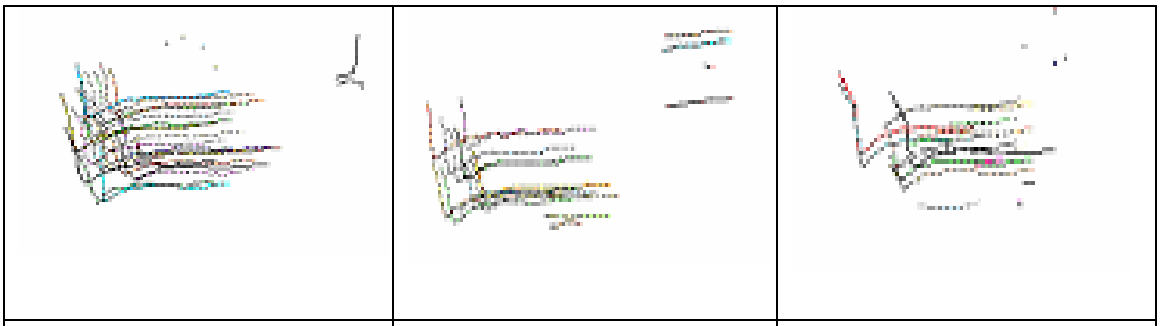
In Figure 8 we present an enlarged version of the trajectory image. The image maintains a separate colour for each point feature tracked and specifies the exact point feature position in black for each frame.

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 inflection points.

An inflection point is found 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 9. 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 inflection points in the trajectory images.

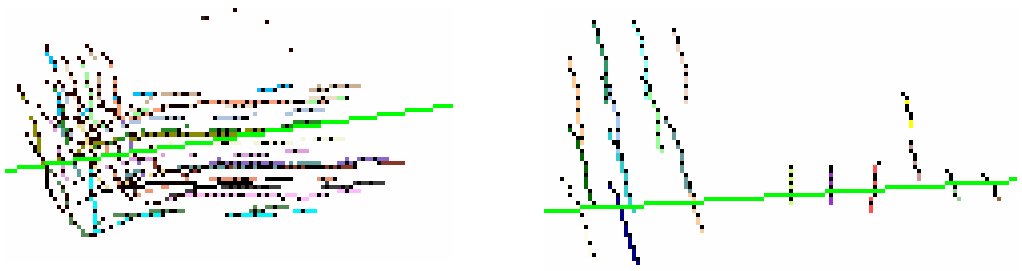
In practice, the presence of obvious inflection points may be quite difficult to find, especially when the motions of the dynamic objects are not under control of the application. Furthermore, the trajectories of non rigid objects have different times in which the change of motion represents itself. For example, the loose fitting clothing of a basketball player making a jump shot continues moving upwards after the player has reached the apex of the jump causing the abrupt change in motion of the player's head and clothing to occur in different features at different points in time. In order to resolve these discrepancies, it would be necessary to first solve the trajectory correspondence problem.



a b c

Figure 9: 3 trajectory images with obviously coincident points of inflection

A closer examination of motion trajectories and the corresponding frames in the video sequences helps to show how the gross approximation errors 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. 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 localization of the frames impossible.



a b

Figure 10: Epipolar line and trajectories. The intersection of trajectories and the epipolar line make up the candidate set of points. (a) large trajectory (b) reduced trajectory

Once we have identified the inflection points, and implicitly the gross approximation of the synchronization, it is further refined by creating a reduced trajectory image around the point of inflection and a geometric consensus stage is applied. In cases where inflection points cannot be reliably found, the gross approximation stage can be omitted and we use

the larger trajectory images in the consensus stage. This will result in many more consensus trials being performed, as we see in Figure 10a, because the epipolar line will potentially intersect with many more trajectories causing the candidate set to be larger. In Figure 10b we show a reduced trajectory image (enlarged for viewing) for an 8 frame track after the point of inflection.

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 11, 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 3^{rd} 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 11.

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. The putative synchronized frame set with the highest consensus overall is selected as the synchronized CCS.

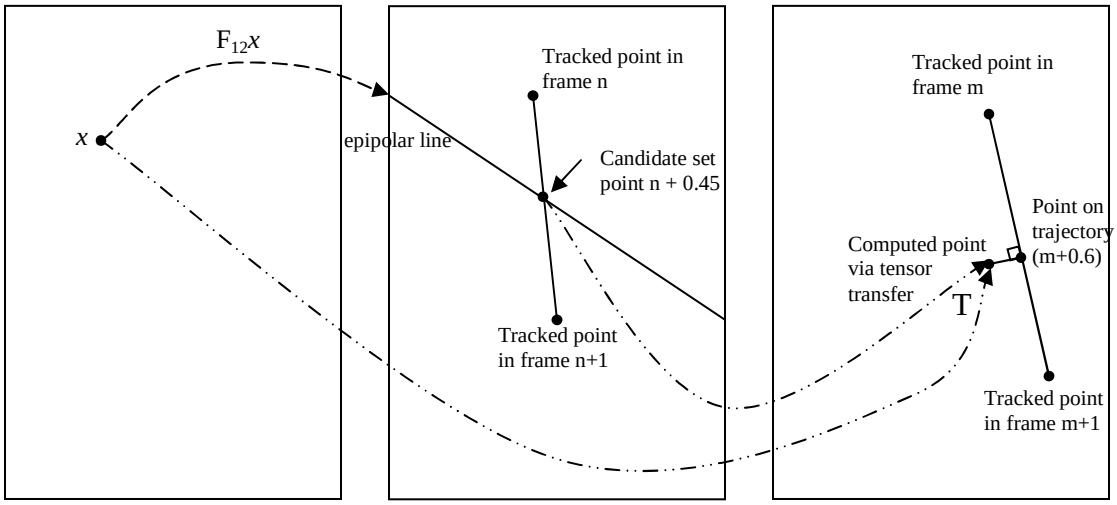


Figure 11: Computed frame values to sub-frame accuracy via epipolar and tensor transfer.

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 (N-2)max_offset^3 which is the number of trials for a smart brute force method as discussed in section 3.

3.5 Synchronization in the face of erroneous geometries.

In the presence of error in the computed geometries, an exact answer cannot be trusted. 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 an 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 gen-

eration 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 value for ε is dictated by our confidence in the computed geometry, its error and the distance between tracked points. This will increase the number of consensus trials by a factor of $(2\varepsilon+1)^2$ times, the number of trials where we have absolute confidence in the computed geometries. The optimal epsilon is function of the frame rates and guarantees us to search at least one primary synchronization point. For each sequence, *i*, epsilon is:

$$\varepsilon = \left\lceil \frac{\lambda}{2\rho_i} \right\rceil \tag{17}$$

This will result in $(N-2)(2\varepsilon+1)^2$ consensus trials. For our experimental trials, we set ε to be 2 as computed from (17).

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(F12 and F23) & triliner tensor(T123)
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 f1 on a trajectory image 1
7  l = F12x (compute Epipolar line)
8  Compute the candidate set of points (xci) i.e. the intersections
   of the trajectories and the epipolar line l
9  For each point pair (x, xci) do
10   xt = Projective_Transfer(x, xci)
11   Compute frame number f2 from xci to sub-frame accuracy (Fig 13)
12   Compute frame number f3 from xt and the nearest trajectory to
          sub-frame accuracy (Fig 13)
13   If (Xcorr( f1,x, ⌊f2+0.5⌋,xci, ⌊f3+0.5⌋, xt) < threshold)
14      Perform guided matching using frames ⌊f1+0.5⌋⌊f2+0.5⌋⌊f3+0.5⌋ and
           the geometry computed in (2)
15       If consensus feature count is maximal then
           Save f1, f2, and f3
16 Endfor(9)
17 Return f1, f2, and f3
```

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 inflection 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 3. 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 capabilities. The cameras had different capture capabilities such as 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 ½ a frame.

In Figure 12, we show the synchronized frames in rows and sequences in columns for this example. As you can see the events (specifically the hand) are well matched in time

Camera	Gross Approximation	Exact Sync	First Primary Sync Point	Exact Time *Ei* (s)	Universal Time Shift *Si* (s)	Exact First Full Frame	Ground Truth
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

In our next set of experiments, we utilize 3 off the shelf digital cameras with video capture capabilities. The cameras were of varying quality and frame rates. The camera limitations for this set of experiments are presented in Table 2. This reflects the situation depicted in Figure 3. Given the frame rates, we can expect full frame synchronization to fall within 0.033 seconds of the perfectly accurate synchronization.

Camera	Frame Rate	Noise level	Resolution (Sharpness)
1	15	typical	low
2	15	typical	standard
3	10	high	standard

Table 2: Camera Characteristics

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	127	8.467	7.533	126.75	8.45	7.45	127	8.467
2	1/15	228	15.2	0.80	229.33	15.289	0.611	229	15.267
3	1/10	160	16	0	159	15.9	0	159	15.9

Table 3: Synchronization Results

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 (inflection points) occurred. These very sharp inflection points allow the gross approximation method to achieve very close results to the synchronized frames with maximal consensus. We can see in Table 3 that the gross approximation in the presence of inflection points are accurate to within a few frames of the exact full frame synchronization. In Table 4, we confirm that the time difference falls within the expected values given the ground truth.

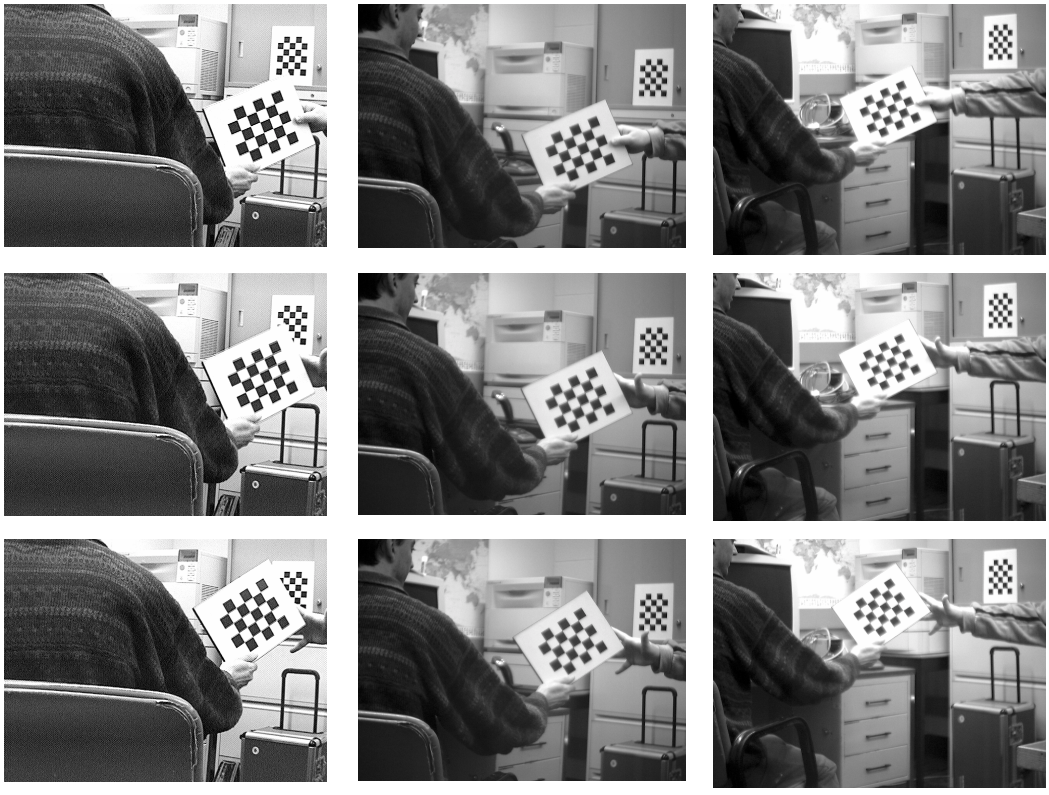


Figure 12: 3 synchronized (rows), consecutive frames from 3 video sequences (columns)

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.

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

Table 4: Synchronization Results

A second example using the same cameras outlined in Table 2 also bring us to the same conclusion. The results, outlined in Table 5, support the previous experiments findings that the gross approximation, in the presence of inflection points is accurate to within a few frames.

Figure 13: Selected Synchronized 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 5: 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 13 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.

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.

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 6: Synchronization Results

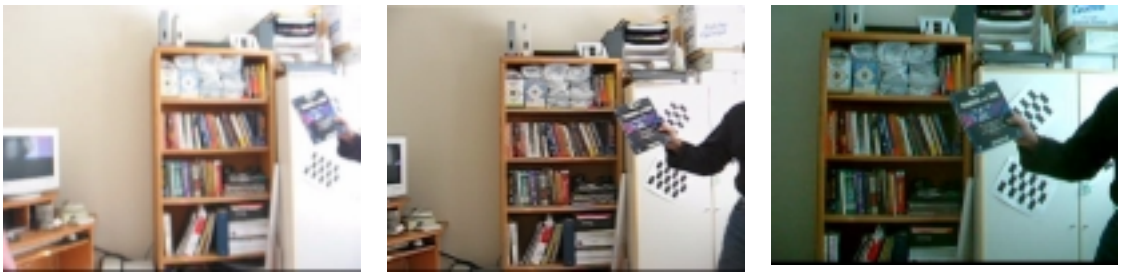


Figure 14: Selected Synchronized Frames

Again, the results shown in Figure 14, listed in Table 6 and Table 7 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 7: Synchronization Results

Example	Camera	Exact with inaccurate geometry	Robust	Ground Truth	Nearest full frame, with accurate geometry
1	1	125.33	127	127	127
	2	227	229	229	229
	3	159	159	159	159
2	1	166.25	167	167	167
	2	211.5	213	214	214
	3	170	170	170	171
3	1	240.66	243	243	243
	2	304.25	302	301	302
	3	274	274	274	274

Table 8: Comparison of methods in the face of geometric inaccuracy

In our final set of experiments, we artificially added error to the computed geometries to simulate degenerate geometries. We then ran the examples again using the robust strategy outlined in section 3.5 for dealing with erroneous geometries. As we can see in Table 8, the strategy of sliding up and down the trajectories and performing more consensus trails on all the local combinations is helpful in the face of inexact geometries. However,

it requires substantially many more consensus trials, and therefore requiring more computing time. By using the robust method, we are able to achieve results that are very close to ground truth.

5 Practical Considerations

During the investigation, several practical issues were raised. Instrumental to the success of the accurate computation of the synchronization is an accurate computation of the camera geometries. The problem of computing an accurate set of camera geometries is considered difficult; and inaccurate geometries, even within a few pixels, can result in an incorrect selection of synchronized frames that are no better than the gross approximation stage. In practice, SIFT features [15] helped to generate accurate geometries in difficult circumstances.

The primary area of concern for the tracking aspect is the avoidance of multiple frame associations to a single feature location and that corresponding features are tracked for some period of time. In order to avoid multiple frame associations, the maximum length of the trajectories should be less than the tracked features intra frame pixel disparity when generating the reduced trajectory images. It is difficult to ensure that corresponding moving object features are tracked in all views, but in practice, well defined textures (such as a chessboard pattern) ensured that many corresponding trajectories existed. However, in situations where targets were not used, simply selecting areas to automatically select features to track helped to improve the situation at the cost of taking away from the hands off approach that the targets allowed. Restricting the area for detection of features to track in an automated method would help to make the algorithm more practical.

Open problems include automatic detection of corresponding inflection points 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) is required.

6 Conclusions

In this work we present a novel method for multiple video temporal synchronization 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 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.

7 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.
[13] Jianbo Shi and Carlo Tomasi. "Good Features to Track". *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593-600, 1994.
[14] Stan Birchfield. "Appendix C, Depth and Motion Discontinuities". *Ph.D. Thesis, Stanford University*, June 1999.
[15] David G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision,* 2004. to appear