# Temporal Synchronization of Video Sequences in Theory and in Practice

**Anthony Whitehead, Robert Laganiere, Prosenjit Bose**

***Abstract*— In this work, we present a formalization of the video synchronization problem that exposes new variants of the problem that have been left unexplored to date. We also present a novel method to temporally synchronize multiple stationary video cameras with overlapping views that: 1) does not rely on certain scene properties, 2) suffices for all variants of the synchronization problem exposed by the theoretical disseration, and 3) does not rely on the trajectory correspondence problem to be solved apriori. The method uses a two stage approach that first approximates the synchronization by tracking moving objects and identifying inflection points. The method then proceeds to refine the estimate using a consensus based matching heuristic to find moving features that best agree with the pre-computed camera geometries from stationary image features. By using the fundamental matrix and the trifocal tensor in the second refinement step we are able to improve the estimation of the first step and handle a broader range of input scenarios and camera conditions.**

***Index Terms*—Video synchronization, video processing, computer vision**

## I. 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 filmmaking that utilize multiple cameras when a screenplay, and 3D reconstruction of dynamic scenes. There is the fundamental problem of sequence synchronization that needs initial resolution before tasks such as those listed above can begin.

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.

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

Manuscript received July 31, 2004. This work was supported in part by NSERC and Nortel Networks scholarship.

A. D. Whtiehead is with the School of Computer Science, Carleton University, Ottawa, Ontario, Canada (Phone: 613-520-2600; fax: 613-520-3434; e-mail: awhitehe@scs.carleton.ca).

R. Laganiere is with the School of Information Technology Engineering, University of Ottawa, Ottawa, Ontario, Canada (e-mail: laganier@site.uottawa.ca).

P. Bose is with the School of Computer Science, Carleton University, Ottawa, Ontario, Canada (email: jit@scs.carleton.ca)

The synchronization problem can now be expressed as finding a set of frames numbers, $f_1, f_2, \ldots, f_k$, one from each 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*.

However, due to possible minute variations in camera start times and variations in frame rates, perfect integral synchronization does not always exist. In such a case we search for a set of frames whose pair-wise difference with respect to the synchronization equality is minimized. 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-time 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, $f_1, f_2, \ldots, f_k$, such that synchronization equality $T_1(f_1) = F_2(f_2) = \ldots = T_k(f_k)$ 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 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.

Synchronization is often assumed however, since the processing of large volumes of video data is becoming tractable, recent work has investigated the problem of synchronizing video sequences. In [1], the method will require that the frame rates be that same for all views in order for the correlation phase to properly be modeled. In [2] the synchronization problem is constrained to having a large planar surface present. The method also suffers under certain 3D motions such as similar objects moving in a line with constant velocity. In [3], the method is also constrained by a large ground plane being present, but further requires intrinsic camera parameters. Furthermore, the method assumes a homogenous camera system. In [4] 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. In [5], 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. In [6,7], 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.

Generally speaking these methods are restrictive because of the requirements of identical frame rates, large planar surfaces being present, or the requirement that the camera system be partially, if not fully, calibrated. 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 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, and that moving objects have sufficient texture for feature tracking. The motion of the moving objects is also slightly constrained in that they cannot have a periodic characteristic such as a pendulum, nor can the motion be directly along the optical axis of one of the cameras.

This work is organized as follows: Sections 2 and 3 formalizes the problem of synchronizing video sequences and introduces terminology. Section 4 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 4 ends with an adaptation to handle errors in the computed geometries. In Section 5, we perform experiments with our proposed method and present results. In section 6, we draw conclusions.

## II. Problem Formalization

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* ($\rho$), is actually the *frequency*. Recall that frequency and the period are inversely related. Figure 1 plots the function $F_i$. The peaks occur when a frame is captured and the valleys occur 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 time and the full frame synchronization.

***Defintion:*** A *primary synchronization point* is a 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 of (3).

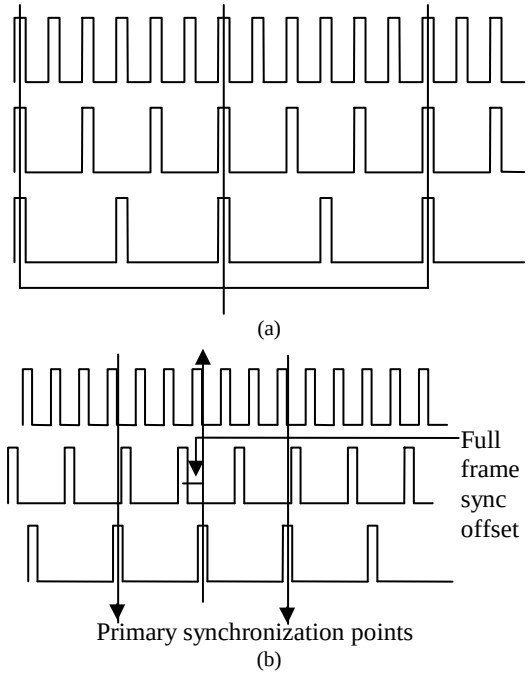


**Figure 1**: a) Perfectly synchronized video sequences with varying frame rates showing primary synchronization points. b) Imperfectly synchronized video sequences with varying frame rates showing primary and secondary synchronization points

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 1a, 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.

Primary synchronization points occur at regular intervals that are a function of the frame rates of the individual sequences.

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

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

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

We coin the term *primary synchronization period*, denoted by the symbol $\lambda$, to be the time between these events.

In practice however, we do not always have perfectly synchronized video sequences as shown in Figure 1a. Instead, we have a slight synchronization offset. We can see in Figure 1b,

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.

Given two imperfectly synchronized video sequences, the maximum full frame sync offset is half the maximum difference between two frame captures (period) of the higher frame rate sample. As we can see in Figure 1b 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, the quality of full frame synchronization is bound by:

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

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

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

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 $\gamma_1$, $\gamma_2$ and $\gamma_3$ be the three slowest frame periods ($\gamma_i = 1/\rho_i$) from N sequences such that: $\gamma_N < \ldots < \gamma_3 < \gamma_2 < \gamma_1$. For all sequence pairs, the offsets defined by (6) are $\gamma_2/2$, $\gamma_3/2$, and $\gamma_3/2$ respectively. Since $\gamma_2$ is greater than $\gamma_3$, $\gamma_2/2$ is greater than $\gamma_3/2$. As $\gamma_3$, $\gamma_2$ and $\gamma_1$ are the three slowest rates, any other frame rate $\gamma_i$ (i>3) from the N sequence is less than $\gamma_3$ resulting in an application of (6) resulting in $\gamma_i/2$ which is less than our largest value $\gamma_2/2$. Therefore, the error is bounded by $\gamma_2/2$, half of the 2nd slowest frame period. □

Given two frames from a single video sequence i, the amount of time that elapses between frame $f_1$ and $f_2$ is $(f_2 - f_1)*\rho_i$. 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 ($n_i$) in sequence i will be taken at elapsed time $E_i$ and is given by the following equation:

$$E_i = f_i \cdot \rho_i \quad (8)$$

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 now specify the exact nature of the functions given by (1) and (2):

$$T_i(f_i) = E_i + S_i \quad (9)$$

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 2, 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.

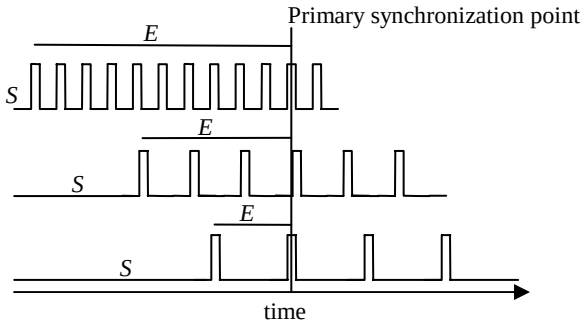


**Figure 2:** Video sequences relative to a universal time line.

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:

$$T = E_i + S_i = E_j + S_j = E_k + S_k \quad (10)$$

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 that account for the maximum error Δ.

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

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, 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.

## III. 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 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. A *synchronized CCS* is a cross camera subset where each frame of the set is full frame synchronized as outlined in equation (1). The problem of camera synchronization is that of determining 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). 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. A dynamic-CCS is the set of images in which we are utilizing the dynamic objects of the cross camera set. Again this is not to say that all pixels within a set are moving, but rather they contain the same moving objects.

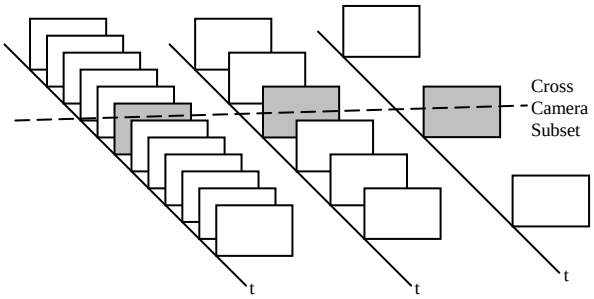


**Figure 3**: 3 camera sequences in a three video camera setup with varying frame rates, with a synchronized cross camera set in gray.

## IV. Recovery of Synchronization

Because we are utilizing multiple stationary video cameras, we can use the fundamental matrices [8] and the trifocal tensor [9] of the three views to determine the synchronization. There is 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 that same instant in time. When dynamic-CCS objects concur with the geometry computed from the 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. Space constraints prevent us from providing complete details, but a full and complete technical report is available from the authors.

### *A. Computing Camera Geometry from the Static-CCS*

Because the cameras are stationary, and the features considered in a static-CCS are also stationary and we can select any frames, so long as they minimize the effect of the moving objects. 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, we can simply sample frames from each camera sequence so that they are well distanced in time. Once the static-CCS has been selected, it is used to first compute information about the camera geometry. The fundamental matrix and the trifocal tensor, are required from a 3 camera system and can be computed robustly using techniques outlined in [8][9].

### *B. Generating the Trajectory Images*

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. The feature tracker we use is based on the work of Lucas and Kanade in [10]. We generate the trajectory images by tracking features from frame-to-frame, and plotting the feature location into the trajectory image. 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. An example trajectory image is given in Figure 4 where an object was moved purposely to show a point of inflection. In order to effectively track moving objects, they need a distinguishable texture.

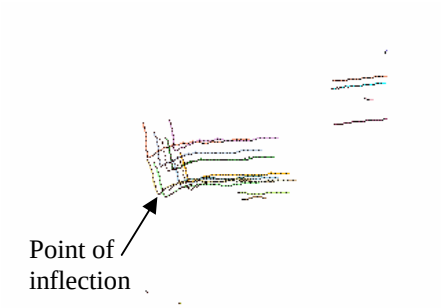


**Figure 4**: a trajectory image generated by tracking features on moving objects.

### *C. Gross approximation of synchronization*

We begin by performing a gross approximation of the full frame synchronization by looking for inflection points. An inflection point is found examining the trajectories for changes in direction. In the presence of object motion where direction is changed suddenly; the trajectories show this change at a very obvious as point shown in Figure 4. 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 frame 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 or on non-rigid objects. However, in the absence of inflection points, or 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 because the epipolar lines will potentially intersect with many more trajectories causing the candidate set to be larger.

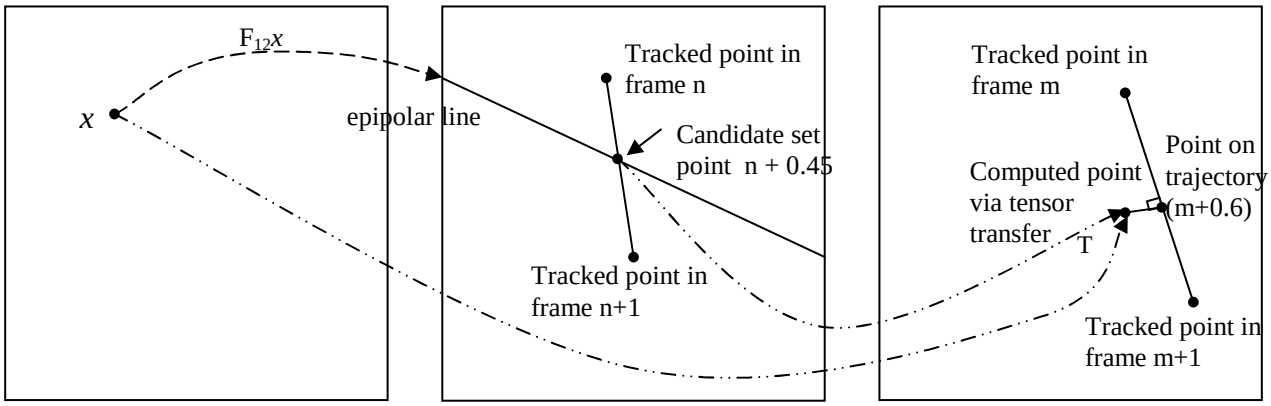


**Figure 5:** Frame vales to sub-frame accuracy via epipolar and tensor transfer

Once we have identified an inflection point, the synchronization is grossly determined by frames associated to the inflection point. The synchronization can then be further refined by creating a reduced trajectory image around the point of inflection and a geometric consensus stage is applied. To improve efficiency, we then generate a "reduced" trajectory image around the inflection. The reduced trajectory image consists of trajectory data for ± four frames around the inflection point.

*D. Refinement of synchronization*

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 5, 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 5.

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 each point 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 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 CCS with the highest consensus overall is selected as the synchronized CCS.

*E. 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 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 examine a number (ε) of complete frames on either side of the epipolar line. 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 rate and the primary synchronization period (5) and guarantees us to search at least one primary synchronization point. For each sequence, *i*, epsilon is:

$$\varepsilon = \left\lceil \frac{\lambda}{2\rho_i} \right\rceil \tag{12}$$

## V. 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 hand selected ground truth. Due to space constraints, we present the results in condensed form.

In our synthetic data set, the frame rates are the same and constant for each generated sequence. The offsets were set to be 0, 5 and 10 frames respectively. 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, 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 1b. Under these conditions, the synchronization was computed exactly to be frame deltas 0, 5.25 and 10.75 respectively.

In our next 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 1a). As we can see in Table 1 the computed full frame synchronization is correct.

| 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

| Camera | Frame Rate | Gross Appr. | Time (s) | Universal Time Shift (s) | Exact Sync | Exact Time (s) | Universal Time Shift (s) | Nearest Full Frame | Time |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 15 | 127 | 8.467 | 7.533 | 126.75 | 8.450 | 7.450 | 127 | 8.467 |
| 2 | 15 | 228 | 15.20 | 0.800 | 229.33 | 15.289 | 0.611 | 229 | 15.267 |
| 3 | 10 | 160 | 16.00 | 0 | 159 | 15.900 | 0 | 159 | 15.900 |

Table 2: Synchronization Results

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 period.

In our next experiment, we utilize 3 off the shelf digital cameras with video capture capabilities. The cameras were of varying quality and frame rates. This reflects the situation depicted in Figure 1b. Ground truth was user selected to be frames 127, 229, and 159 respectively. In this example, an object was moved such that an obvious change of direction (inflection points) occurred (figure 4). 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 2 that the gross approximation in the presence of inflection points are accurate to within a few frames of the exact full frame synchronization.

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. The results all fell within 1 frame of the ground truth.

## VI. Conclusions

In this work we introduce the theoretical groundwork that formalizes the problem of video synchronization and introduces several variants of the problem not previously exposed in the literature. We elucidate the necessary concerns around the utilization of a non-homogenous set of cameras and show that, in the case of varying frame rates, there are areas of synchronization that are superior to others and dub these *primary synchronization points*. This is important for applications that use computer based USB cameras for video capture as the driver software generally will make the frame rates variable through frame dropping. Furthermore, we present a novel method for synchronizing multiple video sequences using feature tracking and geometric consensus solely in projective 2D space. The proposed method has the least constraints placed on the camera setup, the scene being viewed, and does not require the trajectory correspondence problem to be solved apriori. The method also provides two levels of accuracy by using a two step process of grossly approximating the frame synchronization followed by a refinement step that examines 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 accurate synchronization results at the cost of performing more consensus trials to account for the geometric inaccuracies. Space constraints prevented a full disclosure of our experiments; however, a full and complete technical report is available from the authors.

## References


[1] J. Yan and M. Pollefeys, “Video Synchronization Via Space-time Interest Point Distribution”, ACIVS, 2004

[2] L. Lee, R. Romano, G. Stein, “Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame”, IEEE PAMI, Special Section on Video Surveillance and Monitoring, 22(8), 2000

[3] J. Kang, I. Cohen, G. Medioni. “Continuous mutiview tracking using tensor voting”, Proc. of Workshop on Motion and Video Computing, 2002. pp 181-6

[4] Y. Caspi and M. Irani. “Alignment of non-overlapping sequences”. Proc. ICCV, Vancouver, pp 76-83, 2001.

[5] S. Kuthirumal, C. Jawahar, and P.J. Narayanan. ”Video frame alignment in multiple views”. Proc. ICIP, 2002.

[6] C. Rao, A.Gritai, M. Shah. “View-invariant Alignment and Matching of Video Sequences”, In Proc. of ICCV, pp 939-945, 2003.

[7] P. Tresadern and I. Reid. “Synchronizing Image Sequences of Non-Rigid Objects”, In Proc. BMVC, 2003

[8] P.H.S. Torr and D.W. Murray, “The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix”, IJCV, v. 24 pp 271-300, 97

[9] P.H.S. Torr and A. Zisserman. “Robust Parameterization and Computation of the Trifocal Tensor”. Proc. BMVC, pp 655-664. 1996.

[10] C. Tomasi and T. Kanade. “Detection and Tracking of Point Features”. Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.