Frame-level Temporal Calibration of Video Sequences from Unsynchronized Cameras by Using Projective Invariants

Senem Velipasalar
Electrical Engineering Department
Princeton University
Princeton, NJ 08544
svelipas@princeton.edu

Wayne Wolf
Electrical Engineering Department
Princeton University
Princeton, NJ 08544
wolf@princeton.edu

Abstract

This paper describes a new method for temporally calibrating multiple cameras by image processing operations. Existing multi-camera algorithms assume that the input sequences are synchronized either by genlock or by time stamp information and a centralized server. Yet, hardware-based synchronization increases installation cost. Hence, using image information is necessary to align frames from the cameras whose clocks are not synchronized. Our method uses image processing to find the frame offset between sequences so that they can be aligned. We track foreground objects, extract a point of interest for each object as its current location, and find the corresponding location of the object in the other sequence by using projective invariants in P^2. Our algorithm recovers the frame offset by matching the tracks in different views, and finding the most reliable match out of the possible track pairs. This method does not require information about intrinsic or extrinsic camera parameters, and thanks to information obtained from multiple tracks, is robust to possible errors in background subtraction or location extraction. We present results on different sequences from the PETS2001 database, which show the robustness of the algorithm in recovering the frame offset.

1. Introduction

There is an increasing interest in multi-camera systems, as one camera provides only limited amount of information. A single camera has limited field of view (FOV) and cannot cover a wide enough area. Certain applications, such as monitoring of areas for surveillance and statistics gathering purposes, require the coverage of larger areas and longer tracking times. In addition, occlusion, which can be an important problem for a single camera, may be overcome by using multiple cameras.

Having synchronized video inputs is very important for multi-camera systems. For example, a multi-object multi-camera tracking method is presented by Khan and Shah in [1], where the video sequences need to be synchronized to be able to perform consistent labeling, and to find the correspondences between the tracks in different views. Another multi-camera tracking system is presented by Cai and Aggarwal in [2] which needs synchronized video streams.

Temporal calibration identifies corresponding frames in several video sequences captured by different cameras. A low-level method for temporal calibration is synchronization which forces cameras to capture the corresponding frames at the same time by having a master clock. A generic temporal calibration method, based only on image information, provides a solution for cameras without a common clock as well and removes the need for special equipment and hardware.

Existing systems use either genlock or time stamp information and a centralized server to synchronize cameras. However, systems using centralized servers are too expensive to install in most environments, as they require overly long cables to run between cameras, and installation cost is a major concern in real-world surveillance systems. Hardware-based solutions require special equipment and place limits on the lengths of video cables. Using image information is a better way to align the frames from cameras whose clocks are not synchronized.

An approach in Fourier domain has been presented by Kuthirummal et al. [3], which, however, needs to compute weak calibration in the form of the trilinear tensor before alignment, thus requires at least 7 stationary corresponding points in three views. In addition, a point needs to be tracked over a number of frames in three views. Lee et al. [4], use geometric constraints to align the tracking data in time. This method requires the knowledge of the intrinsic camera parameters, and accuracy can be affected by the height of the objects, thus by their distance to the cameras.

A robust and efficient method, which uses image information and does not require camera calibration, camera parameter knowledge, or special equipment and wiring, is necessary for temporal calibration of videos from unsynchronized cameras. In order to fill this gap, we propose a novel method in which foreground objects are tracked, a point of interest is extracted for each object as its current location, and the corresponding location of the object in the other sequence is obtained by using projective invariants in P^2. This calculated corresponding location, the location extracted in the current view, and the current frame number are saved together with the label of the tracker for each track. Fig. 1 illustrates the problem, and shows the data that is saved. $F_i^{L_a^1}$ denotes the frame number at which the i^{th} entry of the a^{th} track with label L_a^1 is obtained. Thus, for the first camera $F_i^{L_a^1}$ is equal to $F_k^{L_b^1}$, meaning there are two tracks coexisting at this frame. The other camera captures the same event, and the goal is to recover the frame offset $F_j^{L_{a'}^2} - F_i^{L_a^1}$ between the corresponding frames.

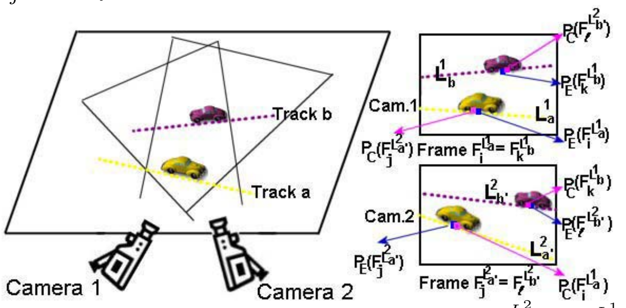


Figure 1: The goal is to recover the frame offset $F_j^{L_{a'}^2} - F_i^{L_a^1}$ between the corresponding frames captured by different cameras with unsynchronized clocks.

Using information from multiple tracks provides robustness to possible errors or spurious parts caused by background subtraction (BGS). The frame offset is recovered by matching the tracks in different views, and finding the most reliable match out of the possible track pairs. The initial matching of the tracks in different views is performed by minimizing a distance measure between the actual locations and calculated locations of the objects received from the other sequence. This method does not require any information about the intrinsic or extrinsic camera parameters.

The algorithm has four parts: 1) tracking each object, extracting its location in the current sequence, and calculating its corresponding location in the other view; 2) for each track obtained from the current sequence, finding a possible match in the saved track list of the other sequence and recovering an initial frame offset value for this match; 3) performing a confidence check for each matched track pair by using the recovered offset value from this match to find the most reliable matching track pair and candidate offset; 4) refining the result by using the pair of tracks obtained from the previous step to check the preciseness of the initial frame offset given by this pair, and to find an offset value aligning this pair of tracks optimally. The last two steps make the algorithm very robust against poor BGS performance and/or possible location detection errors.

We make the following assumptions: the cameras are static and have the same frame rate; objects move on a planar surface; and even if not always, during a short period of time, the bottom part of the objects are visible. The assumption about the same frame rate can be removed by having a different matching measure, which will be implemented as future work.

If camera calibration information is available, points to use can be picked in different ways, their corresponding locations in the other views can be found by using the calibration information, and the remaining steps of our method can still be applied to these points. If cameras are not calibrated, then points and their corresponding locations can be obtained as described in Section 3.1. Thus, the presented method is generic and can be used in different scenarios.

2 Projective Invariants and Computing Corresponding Locations

Let's denote the two cameras, which have been used to capture the two video sequences, by C^i and C^j, and a point on the ground plane of C^j by $p_g^{(j)}$. The corresponding location of $p_g^{(j)}$ in the view of C^i will be computed by using the projective invariants.

A projective invariant is a measurement that does not change under the projective transformations. On the projective plane P^2, five points in general position, i.e. no three of them are collinear, have two independent projective invariants which are defined as follows [6]:

$$I_1 = \frac{|M_{421}^{(1)}||M_{532}^{(1)}|}{|M_{432}^{(1)}||M_{521}^{(1)}|} = \frac{|M_{421}^{(2)}||M_{532}^{(2)}|}{|M_{432}^{(2)}||M_{521}^{(2)}|}, \tag{1}$$

$$I_2 = \frac{|M_{421}^{(1)}||M_{531}^{(1)}|}{|M_{431}^{(1)}||M_{521}^{(1)}|} = \frac{|M_{421}^{(2)}||M_{531}^{(2)}|}{|M_{431}^{(2)}||M_{521}^{(2)}|}, \tag{2}$$

where $|M_{abc}^{(i)}|$, $\{a,b,c\} \in \{1,\dots,5\}$, denotes the determinant of the matrix $M_{abc}^{(i)}$ for image i, whose columns are the homogeneous coordinates of the points $p_a^{(i)}$, $p_b^{(i)}$ and $p_c^{(i)}$.

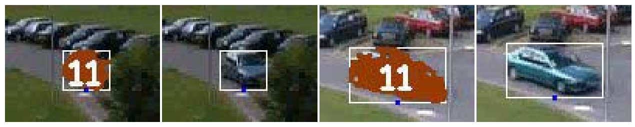
The inputs to our system are four pairs of corresponding points (chosen off-line on the ground plane) in two different sequences. These points in the views of C^i and C^j are denoted by $P^{(i)} = \{p_1^{(i)}, ..., p_4^{(i)}\}$ and $P^{(j)} = \{p_1^{(j)}, ..., p_4^{(j)}\}$ respectively. The points in $P^{(j)}$ and another point $p_g^{(j)}$, which is on the ground plane of the scene of C^j, and not collinear with any of the two points in $P^{(j)}$, form the five points from which 2 projective invariant values are calculated by using (1) and (2). Then, the points in $P^{(i)}$ are inserted in (1) and (2), and these equations are rewritten to solve for the corresponding point of the $p_g^{(j)}$ in the view of C^i.

3 Temporal Calibration Algorithm

As the focus of this paper is temporal calibration, we assume that the foreground objects have been obtained in the current frame by using a background subtraction (BGS) algorithm of choice, such as mixture of Gaussians method described in [5] or a derivative of it.

3.1 Obtaining the Calibration Data

Our first step is to identify representative points for the targets. For each foreground object the aspect ratio is calculated. The aspect ratio (AR) is defined as the height of the boundary box divided by its width. As shown in Fig. 2, if $AR \leq 1$, then depending on the difference in the viewing angles of the cameras, the midpoints of the bottom lines of the boundary boxes in different views will not correspond to the same point in the scene, and the distance between a midpoint and the calculated location received from the other view will be high with high probability. Even if there is no frame offset between the sequences, when the corresponding location of the blue midpoint (Fig.2(d)) in the first view is calculated, the distance between this calculated point and the midpoint in the first view (Fig. 2(b)) may not be small as they do not correspond to the same ground point. If the difference in the viewing angles increases, this problem becomes even more pronounced. This may make the algorithm error-prone, and cause the frame offset not being detected correctly.

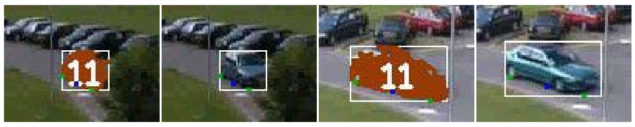


(a) (b) (c) (d)

Figure 2: Using the midpoint of the bottom line of the bounding box can cause errors. The blue points in (b) and (d) do not correspond to the same ground location.

In order to avoid the above problem and make the algorithm more precise, we do not use the midpoint of the bottom line of the boundary box as the current location of the object. Depending on the value of the AR, we follow two different sets of steps to detect the point of interest.

If $AR \leq 1$, the direction vector of the moving object obtained from the tracking data is used to extract the current ground location of the object. If the slope of the direction is negative, the algorithm starts at the left boundary of the foreground mask and searches, in a small width w, the point with the smallest y-coordinate, which is picked as the first anchor point. Then the second anchor point is calculated so that it has the same y-coordinate as the bottom line of the boundary box, and the line between the two anchor points has the same slope as the direction vector. In rare cases, when direction vector cannot be obtained correctly, second anchor point may not lie on the boundary box, and if this happens second anchor point is set to be the lower right corner of the boundary box. The means of the coordinates of the anchor points are set to be the coordinates of the location of the object. Fig. 3 shows the points obtained by our algorithm. If the slope of the direction is positive, this time the algorithm starts at the right boundary of the mask to find the first anchor point, then the second anchor point and the location of the object are obtained similarly.

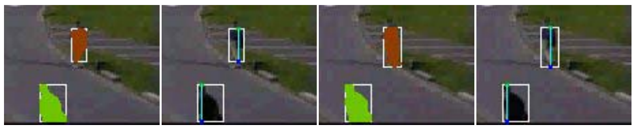


(a) (b) (c) (d)

Figure 3: Points obtained by our algorithm. The green points are the anchor points, and the blue point is the point used as the current location of the object.

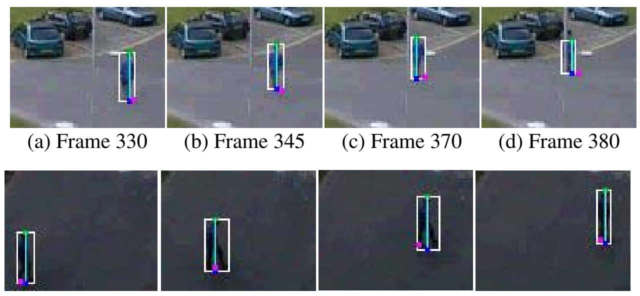
Corner detection may be another way to find the anchor points, but it is not preferred for several reasons: 1) it is not guaranteed that these points will be detected, as one corner threshold value may not work for all videos; 2) the detected corners may not be localized correctly; 3) there will still be a need for the direction vector to be able to pick the right corners; 4) when it is necessary to perform the temporal calibration in real-time, a computationally more expensive method to finely localize these points is not preferable. The proposed method results in reliable results in recovering the frame offsets which will be discussed in Section 4.

If $AR > 1$, then the foreground object is likely to be a person. Using the midpoint of the bottom line of the boundary box can still be a problem, as a person can carry an item such as a bag. In this case, detecting the top of the head, drawing a perpendicular line from it, and using the intersection of this line with the bottom of the boundary box as the location of the object will result in better results. However, for the detection of the top of the head, relying on corner detection on the silhouette of the object is not always reliable. Depending on the object size, and the performance of the BGS, we may not always have the head or the exact silhouette. Thus, we take the mean of the coordinates of the points with the highest y-coordinate to find the point of interest on the top. As these steps are repeated during the period the object is tracked, there will be information from multiple frames which will increase robustness to errors caused by BGS. Figures 4 and 5 illustrate the extracted location points. As seen in Fig. 4, the silhouette of a foreground mask does not always have a corner for the head, and the proposed method can still be used in these cases.



(a)Frame 470 (b)Frame 470 (c)Frame 474 (d)Frame 474

Figure 4: Extracted points of interest when $AR > 1$.


(a) Frame 330 (b) Frame 345 (c) Frame 370 (d) Frame 380

(e)Frame 330 (f)Frame 345 (g)Frame 370 (h)Frame 380

Figure 5: Blue points are the extracted points of interest. The pink point in (a)[(e)] is the calculated corresponding location of the blue point in (e)[(a)]. Similarly for all columns.

Target classification can be used as well, instead of AR, to decide which set of steps to follow for point extraction. After the location of the foreground object is extracted in the current sequence, its corresponding location in the other sequence is calculated as described in Section 2. Fig. 5 shows both the extracted locations, and the calculated locations received from the other view. For each track, the frame number, the coordinates of the current location, and the calculated coordinates of its corresponding location are saved as the track data in a data structure where the label of the tracker is mapped to the track data. These steps are repeated in parallel for the other sequence as well, and the saved track information for each sequence constitutes the temporal calibration data.

If the angle between the two cameras is equal to or larger than 90°, then neither the midpoint of the bottom line of the boundary box nor the scheme here can give the same location point in two different views for the objects with $AR \leq 1$. In this case, the points obtained from the people in the scene can still be used. Another solution is to use more than two cameras.

As will be discussed in Section 4, the algorithm is robust to possible BGS and location extraction errors, and similar motion patterns, and performs reliably in case of divided tracks which may be due to possible occlusions or merges.

3.2 Matching the Tracks

After obtaining the track data for each sequence, each track in the first sequence is matched with a track in the other by minimizing a distance measure between the points forming the tracks and their calculated locations received from the other sequence.

Let L_a^c be the label of the a^{th} track in the c^{th} camera view. Thus, $c \in \{1,2\}$ and $a \in \{1,2,\ldots,N_c\}$ where N_c is the number of tracks in the sequence captured by the c^{th} camera (Fig. 1). The track data for L_a^c is in the format displayed in (3), where $F_i^{L_a^c}$ is the frame number for the i^{th} point in the track, $P_E(F_i^{L_a^c}) = (x_{E_i}^{L_a^c}, y_{E_i}^{L_a^c})$ is the extracted location of the foreground object at frame $F_i^{L_a^c}$ in the current view, and $P_C(F_i^{L_a^c}) = (x_{C_i}^{L_a^c}, y_{C_i}^{L_a^c})$ is the corresponding location of $P_E(F_i^{L_a^c})$ in the other view, calculated at frame $F_i^{L_a^c}$ by using projective invariants. The extracted and calculated locations are shown by blue and pink respectively in Fig. 1. $|L_a^c| = n$ denotes the length of the track.

$$L_a^c \rightarrow \left\{ \begin{array}{c} \left(F_1^{L_a^c}, x_{E_1}^{L_a^c}, y_{E_1}^{L_a^c}, x_{C_1}^{L_a^c}, y_{C_1}^{L_a^c}\right) \\ \left(F_2^{L_a^c}, x_{E_2}^{L_a^c}, y_{E_2}^{L_a^c}, x_{C_2}^{L_a^c}, y_{C_2}^{L_a^c}\right) \\ \vdots \\ \left(F_n^{L_a^c}, x_{E_n}^{L_a^c}, y_{E_n}^{L_a^c}, x_{C_n}^{L_a^c}, y_{C_n}^{L_a^c}\right) \end{array} \right\} \tag{3}$$

Let's denote the Euclidean distance between the points $P_C(F_i^{L_a^1})$ and $P_E(F_j^{L_t^2})$ by $d(P_C(F_i^{L_a^1}), P_E(F_j^{L_t^2}))$, where $a \in \{1,2,\ldots,N_1\}$ and $t \in \{1,2,\ldots,N_2\}$. $D(F_i^{L_a^1}, F_j^{L_t^2})$ in (4), which is the sum of two Euclidean distances, will be called the *point-wise distance measure* between the points of tracks in different cameras at frames $F_i^{L_a^1}$ and $F_j^{L_t^2}$.

$$\begin{aligned} D(F_i^{L_a^1}, F_j^{L_t^2}) &= d(P_C(F_i^{L_a^1}), P_E(F_j^{L_t^2})) + \\ &+ d(P_E(F_i^{L_a^1}), P_C(F_j^{L_t^2})) \end{aligned} \tag{4}$$

$$\begin{aligned} D(F_{i+\Delta}^{L_a^1}, F_{j+\Delta}^{L_t^2}) &= d(P_C(F_{i+\Delta}^{L_a^1}), P_E(F_{j+\Delta}^{L_t^2})) + \\ &+ d(P_E(F_{i+\Delta}^{L_a^1}), P_C(F_{j+\Delta}^{L_t^2})) \end{aligned} \tag{5}$$

We formulate the initial track matching problem as:

$$\{t^*, i^*, j^*\} = \underset{\substack{t\in\{1,2,\ldots,N_2\} \\ i\in\{1,2,\ldots,|L_a^1|\} \\ j\in\{1,2,\ldots,|L_t^2|\}}}{\operatorname{argmin}} \left[D(F_i^{L_a^1}, F_j^{L_t^2}) + D(F_{i+\Delta}^{L_a^1}, F_{j+\Delta}^{L_t^2})\right] \tag{6}$$

where, for a track L_a^1 we find a track $L_{t^*}^2$, $t^* \in \{1,2,\ldots,N_2\}$, and indices i^* and j^* in the track data of L_a^1 and $L_{t^*}^2$ respectively, so that the total distance measure defined on the right hand side is minimized over all possible t, i and j. In (6), $\Delta \geq 10$. Henceforth, for clarity, t^* is replaced by a' and $L_{a'}^2$ is used to denote the match of the tracker L_a^1, i.e $a' = t^*$, and the tracker L_a^1 is said to be matched to tracker $L_{a'}^2$. Moreover, $D_{min}^{L_a^1, L_{a'}^2} = D(F_{i^*}^{L_a^1}, F_{j^*}^{L_{a'}^2}) + D(F_{i^*+\Delta}^{L_a^1}, F_{j^*+\Delta}^{L_{a'}^2})$, obtained from this match, will be called the *match distance measure*. After the tracks are matched, the following data is saved for the pair: $(L_a^1, L_{a'}^2, F_{i^*}^{L_a^1}, F_{j^*}^{L_{a'}^2}, O^{L_a^1}, D_{min}^{L_a^1, L_{a'}^2})$. The initial frame offset obtained from this match is $O^{L_a^1} = (F_{j^*}^{L_{a'}^2} - F_{i^*}^{L_a^1})$. The same steps are repeated to find the match of each track.

If, only the $D(F_i^{L_a^1}, F_j^{L_t^2})$ term in (6) were minimized, then any object passing through that one location in any direction at another time could cause a wrong match. Assuming that the frame rates are the same, and imposing the condition that the object should go from one point to the other in Δ frames, this problem is avoided. In our scheme, as will be explained in more detail in Section 3.3, a problem can occur only if *all* the objects in the scene move along the same line with the same speed, i.e. if there is only periodic motion, which is a highly unlikely case. The Δ is required to be ≥ 10 so that there is enough number of frames between the passes of an object through two points.

3.3 Confidence Check

The confidence check is performed to find the most reliable pair of matched tracks. Using the pair for which *the match distance measure* is minimum, among the other track pairs, as the most reliable match may be an option. However, there is always a possibility of another object moving along the same line, i.e. having a very similar motion pattern, at another time. Thus, relying only on the track pair resulting in the minimum *match distance measure* may give the wrong frame offset value. Using the initial offset values obtained for each match and performing a confidence check through the tracks for consistency will avoid this problem and increase robustness and reliability.

Depending on the performance of the BGS, and accuracy of the location extraction, some track matches will be more reliable than the others. Considering this, and with the goal of saving computing power, only the track pairs for which the *match distance measure* is less than or equal to the median match distance measure are kept. This reduces the number of tracks pairs to be used in the confidence check to half. We also performed experiments by keeping different number of tracks which showed the effectiveness of using information from multiple tracks. The results of these experiments will be summarized in Section 4.

Let T^{kept} in (7) be the list of the saved data for the matched tracks that are kept. As stated previously, each tracker label L_a^c, $c \in \{1,2\}$ is mapped to its track data which has the format shown in (3).

$$T^{kept} = \left\{ \begin{matrix} \left(L_a^1, L_{a'}^2, F_{i^*}^{L_a^1}, F_{j^*}^{L_{a'}^2}, O^{L_a^1}, D_{min}^{L_a^1, L_{a'}^2} \right) \\ \left(L_b^1, L_{b'}^2, F_{k^*}^{L_b^1}, F_{l^*}^{L_{b'}^2}, O^{L_b^1}, D_{min}^{L_b^1, L_{b'}^2} \right) \\ \vdots \\ \left(L_r^1, L_{r'}^2, F_{y^*}^{L_r^1}, F_{z^*}^{L_{r'}^2}, O^{L_r^1}, D_{min}^{L_r^1, L_{r'}^2} \right) \end{matrix} \right\} \quad (7)$$

We formulate the confidence check as follows:

$$O^* = \underset{O \in \{O^{L_a^1}, \dots, O^{L_r^1}\}}{\operatorname{argmin}} \frac{1}{|T^{kept}|} \sum_{t=L_a^1}^{L_r^1} \left(\frac{1}{|t|} \sum_{e=1}^{|t|} D(F_e^t, F_e^t + O) \right). \quad (8)$$

First, $O^{L_a^1} = (F_{j^*}^{L_{a'}^2} - F_{i^*}^{L_a^1})$ is obtained from the first matched pair, and used as the candidate offset value. For all the track points of L_a^1, this candidate offset value is added to their frame number (which is the first element of each entry of the track data). Then, the points of a track which exist at the resulting frames in the other sequence are found, the point-wise distance measure in (4) is calculated for each point pair, and the mean of the point-wise distance measures over the point pairs is found. If there are multiple tracks existing at the resulting frames in the other sequence, the minimum of the mean point-wise distance measures obtained from these tracks is used. The same is repeated, again using $O^{L_a^1}$ as the candidate offset, for the track points of $L_b^1, \dots, L_r^1$ and the overall mean of the point-wise distance measure over different tracks is obtained for the offset $O^{L_a^1}$. Then candidate offset values from the other entries in T^{kept} are tried as well. The entry whose frame offset value is equal to the O^* in (8), is picked as the most reliable track match, and, unless the last refinement step is preferred to be performed, the system output is set to be this offset value.

3.4 Refinement of the Frame Offset

The frame offset value obtained at the end of the confidence check step may be refined as a design choice. Refinement can be applied if a finer and more exhaustive search is tolerated.

The most reliable entry of T^{kept} in (7), obtained after the confidence check, is the input to the refinement step. Let this entry be $\left(L_b^1, L_{b'}^2, F_k^{L_b^1}, F_l^{L_{b'}^2}, O^{L_b^1}, D_{min}^{L_b^1, L_{b'}^2} \right)$. Indices i and j are found in the track data lists of L_b^1 and $L_{b'}^2$ respectively, so that

$$\frac{1}{m} \sum_{k=i}^{i+m-1} D(F_k^{L_b^1}, F_k^{L_b^1} + F_j^{L_{b'}^2} - F_i^{L_b^1}) \quad (9)$$

is minimized, where $m = min(|L_b^1|, |L_{b'}^2|)/2$ is the number of track points over which we perform the point-wise distance measure calculation and summation. Then the output of the system for the frame offset value is set to be $F_j^{L_{b'}^2} - F_i^{L_b^1}$.

4 Experimental Results

The proposed algorithm was tested on video sequences from PETS2001 database. Each set of video is captured by two cameras, and the sequences are provided as synchronized. An interface was implemented for the experiments by which the user can enter a frame offset value, and delay one of the sequences by the entered amount. This way, the ground truth for the frame offset is known for each experiment. In all the experiments, the value of Δ was 10.

The results obtained without the final refinement step are summarized in Table 1. Using multiple matched track pairs, and performing a confidence check among all of them for consistency provides robustness against the possibility of different objects following the same motion pattern and direction, and hence increases accuracy. The algorithm is also robust against divided tracks (due to occlusion and merges), as when a track is divided into two, the pieces will be treated as two different tracks, and will still be considered in track matching step, and potentially in confidence check. As an example; when the ground truth for the frame offset value was 400, the initial frame offsets, from the entries of T^{kept} in (7), were 432, 401, 395, 384, 423 and 393. When all these offset values were used in confidence check, the offset value 401 was obtained from (8), and the rest were eliminated.

		FRAME OFFSETS						
Video 1	Ground Truth	100	200	300	400	500	800	1000
	System Output	99	199	301	401	501	795	993
	Accuracy	99%	99.5%	99.67%	99.75%	99.8%	99.37%	99.3%
Video 2	Ground Truth	100	200	300	400	500	800	1000
	System Output	100	200	300	398	498	798	1008
	Accuracy	100%	100%	100%	99.5%	99.6%	99.75%	99.2%

Table 1: The results obtained after evaluating the synchronization algorithm with different videos and frame offsets.

Table 2 summarizes the results obtained before and after applying the refinement step, together with the ground truth. As can be seen, although refinement provides improvement to the results, depending on the stability of BGS during the track, and the accuracy of the location extraction, it may not always result in a high enough improvement compared to the computational cost. Hence, as stated previously, this step becomes a design decision.

	Frame Offset Output Before Refinement	**Frame Offset Output After Refinement**	**Ground Truth**
Video 1	199	199	200
	501	501	500
	795	800	800
	993	995	1000
Video 2	798	798	800
	1008	1007	1000

Table 2: Comparison of the frame offset values obtained before and after the refinement step.

We also performed experiments by keeping different number of the track pairs after the matching step, i.e. we performed experiments with different number of tracks in T^{kept} of (7). The results obtained without the refinement step are displayed in Fig. 6. Using multiple number of matched track pairs, and performing the confidence check increases accuracy.

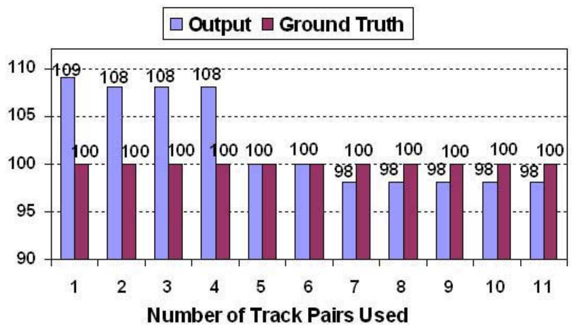


Figure 6: Using multiple number of matched tracks, and performing the confidence check increase accuracy.

Better BGS outputs, and better localization of the four input point pairs can improve the results even more. As people become really small in these sets, the BGS is not always very reliable, and even in this case, the algorithm shows robustness and high accuracy.

5 Conclusions

We presented a novel and robust algorithm for automatic frame-level temporal calibration of video sequences from unsynchronized cameras, which does not require the knowledge of intrinsic or extrinsic camera parameters. The proposed method achieved 99.6% accuracy in the experiments performed with different videos and frame offsets.

We designed our system as a temporal calibration module built upon background subtraction (BGS). The algorithm is robust to errors in BGS unless they are continuous. Different state of the art approaches can be used to improve the output of BGS. For example, in the case of shadows, the input to the proposed algorithm can be fed after applying shadow removal, as shadows can be a problem when extracting the ground location of the object.

By performing a confidence check with the initial offset values obtained from the matched tracks, the algorithm shows robustness against possible BGS and location extraction errors, and similar motion patterns. As multiple tracks are used, and the most reliable match among them is obtained by confidence check, the algorithm also performs reliably in the case of divided tracks which may be due to possible occlusion and merge cases.

References

[1] S. Khan and M. Shah, "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view," *IEEE Trans. on PAMI*, vol. 25, no.10, pp. 1355–1360, Oct 2003.

[2] Q. Cai and J.K. Aggarwal, "Automatic tracking of human motion in indoor scenes across multiple synchronized video streams ," *Int'l Conf. on Computer Vision*, Jan. 1998.

[3] S. Kuthirummal, C.V. Jawahar and P.J. Narayanan, "Video frame alignment in multiple views", *IEEE Int'l Conf. on Image Processing*, vol. 3, pp. 357–360, June 2002.

[4] L. Lee, R. Romano, and G. Stein, "Monitoring activities from multiple video streams: Establishing a common coordinate frame," *IEEE Trans. on PAMI*, pp. 758–768, Aug. 2000.

[5] C. Stauffer and W.E.L. Grimson, "Adaptive background mixture models for real-time tracking", *IEEE Int'l Conf. on CVPR*, vol. 2, June 1999.

[6] C.A. Rothwell, *Object Recognition Through Invariant Indexing*, Oxford Science Publications, 1995.