FRAME-LEVEL TEMPORAL CALIBRATION OF UNSYNCHRONIZED CAMERAS BY USING LONGEST CONSECUTIVE COMMON SUBSEQUENCE

Youlu Wang, Senem Velipasalar *

University of Nebraska-Lincoln
Dept. of Electrical Engineering, 209N WSEC, Lincoln, NE, 68588
youlu.wang@huskers.unl.edu,velipasa@engr.unl.edu

ABSTRACT

We present a computationally efficient and robust method for temporally calibrating video sequences from unsynchronized cameras by using object trajectories. Existing methods remain restricted in terms of their assumptions, and/or they are computationally expensive. To match and align the object trajectories, and thus to recover the frame offset between video sequences, we present an algorithm that is based on the Longest Consecutive Common Subsequence. The candidate frame offsets are obtained from each matched trajectory pair, and then a confidence check is performed. The algorithm is robust against possible errors due to background subtraction and location extraction, and can handle large frame offsets. We present experimental results for different frame offset values on different video sequences, which show the robustness of the algorithm in recovering the frame offsets. We also compare the presented algorithm with our previous work to demonstrate the computational efficiency provided.

Index Terms— Temporal calibration, unsynchronized cameras, frame offset, longest consecutive common subsequence, trajectory alignment

1. INTRODUCTION

We present a method for temporal calibration of video sequences from unsynchronized cameras by using object trajectories. Temporal calibration identifies corresponding frames in video sequences captured by different cameras. A low-level method for temporal calibration is synchronization that forces cameras to capture the corresponding frames at the same time by having a master clock. A generic temporal calibration method that is 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.

Temporal calibration is very important for multi-camera systems, because the transfer of relevant data between cameras is essential. Hardware-based synchronization increases installation cost. An alternative way is to use image/video processing to align frames from the cameras and retrieve the frame offset.

*This work was supported by NSF under Grant CNS–0834753.

Kuthirummal et al. [1] presented an approach in Fourier Domain, which requires at least seven stationary corresponding points in three views. Also, a point needs to be tracked over a number of frames in three views. Lee et al. [2] introduced a method to align the centroids of moving objects. However, centroid points are treated individually rather than as a part of a trajectory, which increases the combinatorial complexity. Moreover, accuracy can be affected by the height of the objects, thus by their distance to the cameras. Caspi et al. [3] also introduced a trajectory-based algorithm. It is assumed that the temporal offset between the two sequences is at most 25 frames. Tuytelaars and VanGool [4] proposed a method that can deal with moving cameras and general $3D$ scenes. However, this method requires tracking five corresponding points in two sequences, which are selected manually as a subset of a feature point set tracked through the video sequence. Velipasalar and Wolf [5] introduced a search algorithm to match and align trajectories obtained from different sequences. This method is robust to errors caused by background subtraction or location extraction. Yet, it performs an exhaustive type of search.

In this paper, we describe a method based on finding the longest consecutive common subsequence (LCCS). Longest Common Subsequence (LCS) was proposed by Vlachos et al. [6] to find similar multi-dimensional trajectories, and was used by Buzan et al. [7] and Cheriyadat and Radke [8] for finding similar trajectories in video sequences. Both [7] and [8] focus on trajectory clustering in a single camera view. We present an LCCS-based algorithm with a customized similarity criteria, and employ it in a multi-camera application to find consecutive matching points as a part of our method. The proposed algorithm provides significant improvement in terms of computational complexity, and has comparable or better results with respect to the previous work described in [5].

2. TRAJECTORY ALIGNMENT USING LCCS

We detect, track and extract the location of each moving object, as described in more detail in the previous work [5], to form trajectory data. Let $L^c_{t_c}$ be the label of the ${t_c}^{th}$ trajectory on the view of camera c. Thus, $c \in \{1, 2\}$ and $t_c \in$

$\{1, 2, \ldots, N_c\}$ where N_c is the number of trajectories in the sequence captured by the c^{th} camera. The trajectory data for label $L^c_{t_c}$ is in the following format:

$$L^c_{t_c} \rightarrow \left\{ \begin{array}{c} \left(F_1^{L^c_{t_c}}, x_{E_1}^{L^c_{t_c}}, y_{E_1}^{L^c_{t_c}}, x_{C_1}^{L^c_{t_c}}, y_{C_1}^{L^c_{t_c}}\right) \\ \left(F_2^{L^c_{t_c}}, x_{E_2}^{L^c_{t_c}}, y_{E_2}^{L^c_{t_c}}, x_{C_2}^{L^c_{t_c}}, y_{C_2}^{L^c_{t_c}}\right) \\ \vdots \\ \left(F_n^{L^c_{t_c}}, x_{E_n}^{L^c_{t_c}}, y_{E_n}^{L^c_{t_c}}, x_{C_n}^{L^c_{t_c}}, y_{C_n}^{L^c_{t_c}}\right) \end{array} \right\} \quad (1)$$

where $F_i^{L^c_{t_c}}$ is the frame number for the i^{th} point in the trajectory, $P_E(F_i^{L^c_{t_c}}) = (x_{E_i}^{L^c_{t_c}}, y_{E_i}^{L^c_{t_c}})$ is the extracted location of the foreground object at frame $F_i^{L^c_{t_c}}$ in the current view, and $P_C(F_i^{L^c_{t_c}}) = (x_{C_i}^{L^c_{t_c}}, y_{C_i}^{L^c_{t_c}})$ is the corresponding location of $P_E(F_i^{L^c_{t_c}})$ in the other view, calculated at frame $F_i^{L^c_{t_c}}$ by using an estimated homography [5].

Longest Common Subsequence (LCS) is an algorithm originally designed for finding similar strings in computer science applications. A two-dimensional trajectory matching application for LCS is proposed in [6]. In this paper, we modify LCS to LCCS, which only finds the longest consecutive common subsequence. We present an LCCS-based algorithm with a customized similarity criteria to find matching trajectories on different camera views.

2.1. LCCS-based algorithm

Let $L^1_{t_1}$ and $L^2_{t_2}$ denote two trajectories containing n and m points, respectively, which are expressed as

$$L^1_{t_1} = \left\{ (F_1^{L^1_{t_1}}, x_{E_1}^{L^1_{t_1}}, y_{E_1}^{L^1_{t_1}}, x_{C_1}^{L^1_{t_1}}, y_{C_1}^{L^1_{t_1}}), \ldots, (F_n^{L^1_{t_1}}, x_{E_n}^{L^1_{t_1}}, y_{E_n}^{L^1_{t_1}}, x_{C_n}^{L^1_{t_1}}, y_{C_n}^{L^1_{t_1}}) \right\}$$

$$L^2_{t_2} = \left\{ (F_1^{L^2_{t_2}}, x_{E_1}^{L^2_{t_2}}, y_{E_1}^{L^2_{t_2}}, x_{C_1}^{L^2_{t_2}}, y_{C_1}^{L^2_{t_2}}), \ldots, (F_m^{L^2_{t_2}}, x_{E_m}^{L^2_{t_2}}, y_{E_m}^{L^2_{t_2}}, x_{C_m}^{L^2_{t_2}}, y_{C_m}^{L^2_{t_2}}) \right\}$$

We define $Head\left(L^1_{t_1}\right)$ and $Head\left(L^2_{t_2}\right)$ as

$$Head\left(L^1_{t_1}\right) = \left\{ (F_1^{L^1_{t_1}}, x_{E_1}^{L^1_{t_1}}, y_{E_1}^{L^1_{t_1}}, x_{C_1}^{L^1_{t_1}}, y_{C_1}^{L^1_{t_1}}), \ldots, (F_{n-1}^{L^1_{t_1}}, x_{E_{n-1}}^{L^1_{t_1}}, y_{E_{n-1}}^{L^1_{t_1}}, x_{C_{n-1}}^{L^1_{t_1}}, y_{C_{n-1}}^{L^1_{t_1}}) \right\}$$

$$Head\left(L^2_{t_2}\right) = \left\{ (F_1^{L^2_{t_2}}, x_{E_1}^{L^2_{t_2}}, y_{E_1}^{L^2_{t_2}}, x_{C_1}^{L^2_{t_2}}, y_{C_1}^{L^2_{t_2}}), \ldots, (F_{m-1}^{L^2_{t_2}}, x_{E_{m-1}}^{L^2_{t_2}}, y_{E_{m-1}}^{L^2_{t_2}}, x_{C_{m-1}}^{L^2_{t_2}}, y_{C_{m-1}}^{L^2_{t_2}}) \right\}$$

The Euclidean distance between the n^{th} extracted point in the first trajectory, and the m^{th} calculated point in the second trajectory is denoted by $d_{E_nC_m}$.

Definition 1 *Given a positive number ϵ, we define $LCCS_\epsilon\left(L^1_{t_1}, L^2_{t_2}\right)$ as follows:*

$$LCCS_\epsilon\left(L^1_{t_1}, L^2_{t_2}\right) = \left\{ \begin{array}{l} 1 + LCCS_\epsilon\left(Head\left(L^1_{t_1}\right), Head\left(L^2_{t_2}\right)\right) \\ \quad if\ d_{E_nC_m} < \epsilon\ and\ d_{C_nE_m} < \epsilon \\ \\ 0 \quad if\ L^1_{t_1}\ or\ L^2_{t_2}\ is\ empty\ or \\ d_{E_nC_m} \geq \epsilon\ or\ d_{C_nE_m} \geq \epsilon \end{array} \right.$$

The constant ϵ is the distance matching threshold. The points that are close in space are regarded as matching points. If the extracted location in the first view and the calculated corresponding location from the second view are close enough *and* the extracted location in the second view and the calculated corresponding location from the first view are close enough, then the matching score is increased by 1. LCCS searches all points in two trajectories sequentially and collects the LCCS score in a recursive way. LCCS only saves the number of consecutive matching points by resetting the LCCS score to 0 once the search meets an unmatched pair.

Since we assume that cameras have the same frame rate, we can find all matching points by LCCS without time stretching. However, there may be possible errors due to background subtraction and/or location extraction. Thus, there may be points in a trajectory, which make LCCS comparison fail, and reset the similarity score to 0. To avoid this, we introduce a positive integer, M, as a threshold for the number of matched points. We only need to find the first M matching points between the two trajectories. Then, we stop searching for matching points by LCCS once we have $LCCS_\epsilon\left(L^1_{t_1}, L^2_{t_2}\right) = M$. We continue to search the trajectory from the last matched point pair. For example, if i^{th} and j^{th} points in two trajectories are the M^{th} matched point pairs, then we stop LCCS at these points. We denote the number of matched points as N_{match}, and N_{match} is set to be M when LCCS-based search is stopped. Then, we compare the $(i+1)^{th}$ and $(j+1)^{th}$ points from the two trajectories, respectively. If they match, N_{match} will be increased by 1, and it will be $M+1$; if they do not match, we move on to the points $(i+2)$ and $(j+2)$ without increasing N_{match}. We continue this search until we reach the end of one of the trajectories.

Definition 2 *We define the similarity function S between two trajectories $L^1_{t_1}$ and $L^2_{t_2}$, given ϵ and M, as follows:*

$$S\left(\epsilon, M, L^1_{t_1}, L^2_{t_2}\right) = \frac{N_{match}}{min\left(n, m\right)}$$

We define the similarity function S by normalizing N_{match} by the minimum length of the two trajectories, thus $0 \leq S \leq 1$. This similarity function S is used as the main criteria to find the best matching trajectories.

Thus, for a trajectory $L^1_{t_1}$ in the first camera view, we calculate the value of S with every trajectory from the second camera. In other words, if there are N_2 trajectories in the second camera view, we perform N_2 many similarity computations. As described above, we have two groups of location co-

ordinates for every point in each trajectory: extracted location and its calculated location in the other view. The distance between the points $P_C\left(F_i^{L_{t_1}^1}\right)$ and $P_E\left(F_j^{L_{t_2}^2}\right)$ is denoted by $d_{C_iE_j}$, where $t_1 \in \{1, 2, \ldots, N_1\}$ and $t_2 \in \{1, 2, \ldots, N_2\}$. With the given distance threshold ϵ, we consider two points matching with each other when $d_{C_iE_j} < \epsilon$ and $d_{E_iC_j} < \epsilon$ are both satisfied.

After computing the similarity scores between the $L_{t_1}^1$ and all the trajectories in the second camera, we pick the trajectory in second camera view, which gives the highest S value, as the match of the trajectory $L_{t_1}^1$. Then, we can easily obtain the frame offset from these two trajectories, since all matching point pairs have the same frame offset. The frame offset from $L_{t_1}^1$ is denoted by $O^{L_{t_1}^1}$, and is obtained by subtracting the frame numbers of any matched pair of points. Then, the two matched trajectories and their corresponding frame offset value are saved as the input of the confidence check step.

We perform the above steps for every trajectory in the first camera view to find their matching trajectory in the second view. The pseudo code for the proposed LCCS-based trajectory matching is presented in Table 1.

for every $L_{t_1}^1, t_1 \in \{1 \ldots N_1\}$
  S_{max} = 0;
 for every $L_{t_2}^2, t_2 \in \{1 \ldots N_2\}$
  n = length($L_{t_1}^1$);
  m = length($L_{t_2}^2$);
  set $table_{match} = [n+1]\,[m+1]$ all 0; k=1;
  while $k \leq n * m$
  $i = floor\left((k-1)/m\right) + 1$;
  $j = k - (i-1) * m$;
  if $d_{C_jE_i} < \epsilon$ and $d_{E_jC_i} < \epsilon$
   $table_{match}\,[i+1]\,[j+1] = 1 + table_{match}\,[i]\,[j]$;
   if $table_{match}\,[i+1]\,[j+1] == M$
    $i_{stop} = i$;
    $j_{stop} = j$;
    $N_{match} = M$;
    break;
   else $k++$;
  set $i = i_{stop}$;
  set $j = j_{stop}$;
  while $i < n$ and $j < m$
   $i++$;
   $j++$;
  if $d_{C_jE_i} < \epsilon$ and $d_{E_jC_i} < \epsilon$
   $N_{match} = N_{match} + 1$;
  else $continue$;
 $S = N_{match}/min\,(n, m)$;
 $O_{L_{t_2}^2}^{L_{t_1}^1} = F_{j_{stop}}^{L_{t_2}^2} - F_{i_{stop}}^{L_{t_1}^1}$;
 if $S > S_{max}$
  $S_{max}^{L_{t_1}^1} = S;\ t_1^{'} = t_2;\ O^{L_{t_1}^1} = O_{L_{t_2}^2}^{L_{t_1}^1}$;
 save $\left(L_{t_1}^1, L_{t_1'}^2, S_{max}^{L_{t_1}^1}, O^{L_{t_1}^1}\right)$

Table 1. Pseudo code for the LCCS-based trajectory matching

In Table 1, t_1' denotes the matching trajectory found for t_1. After we obtain candidate trajectory pairs, we find the median value S_{med} of their similarity scores. We keep the trajectory pairs whose similarity score is greater than S_{med}. This decreases the number of possible matches by half by removing the pairs with low scores. In addition to the computational aspects, this step is useful since a trajectory may not have a real match in the other camera view. This trajectory will have a low score, and will be removed with this step.

2.2. Confidence check for the frame offsets

In this step, we perform a confidence check to find the most reliable frame offset value among the different offset values obtained from the matched trajectories. The confidence check is inherited from [5] and described below. Let Λ denote the set of the trajectory numbers on the current camera view, that are kept with their matched trajectories from the other view. In other words, the set Λ is built from the elements of $\{1, 2, ...N_1\}$ such that the S value calculated for the trajectories with labels $\{L_{t_1}^1 : t_1 \in \Lambda\}$ and their matched trajectories is greater than S_{med}. Let T^{match} be the saved data for the matched trajectories that are kept. T^{match} has the following format:

$$T^{match} = \left\{ \left(L_{t_1}^1, L_{t_1'}^2, O^{L_{t_1}^1}\right) :\ t_1 \in \Lambda \right\}$$

The confidence check is formulated as follows:

$$O^* = \underset{O \in \{O^{L_{t_1}^1} : t_1 \in \Lambda\}}{\arg\min} \frac{1}{|T^{match}|} \sum_{\substack{\tau \in \{L_{t_1}^1 : \\ t_1 \in \Lambda\}}} \left(\frac{1}{|\tau|} \sum_{e=1}^{|\tau|} D(F_e^\tau, F_e^\tau + O) \right) \quad (2)$$

The confidence check starts with a $L_{t_1}^1$, where $t_1 \in \Lambda$, and $O^{L_{t_1}^1}$ which is the offset candidate obtained from the corresponding trajectory pair. For all the track points of $L_{t_1}^1$, this offset candidate is added to their frame numbers. Then the points of a trajectory, which exist at the resulting frames, in the other camera are found. The point-wise distance, $D(F_e^\tau, F_e^\tau + O)$ [5], is calculated for each point pair, and the mean of the point-wise distance measures over the number of trajectory points is found. If there are multiple trajectories existing at the resulting frames in the other camera, the minimum of the mean point-wise distance measures obtained from these trajectories is used. The same process is repeated, again using $O^{L_{t_1}^1}$, for the track points of the next trajectory in Λ, and the overall mean of the point-wise distance measure over different trajectories is obtained for the offset $O^{L_{t_1}^1}$. All offset candidates are tried in this way, and the offset candidate that has the minimum overall mean of point-wise distance over all different trajectories is the best frame offset that we recover.

2.3. Comparison of the proposed method with the previous work

The previous method presented in [5] calculates the distance of *each* point in *each* trajectory of the first camera to the

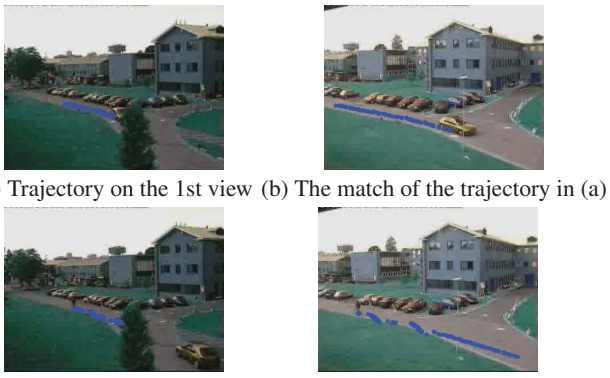
each point in *each* trajectory of the second camera. This exhaustive search involves four main loops, which results in $O(N_1 * N_2 * n * m * C)$ operations. N_1, N_2, n, m are the sizes of each nested loop. C is the number of operations inside the innermost loop.

As seen in Table 1, we set up three loops at the beginning, and the initial sizes of these loops are also $N_1, N_2, n * m$. However, in most cases, the loops are not executed completely. Once we find M many matching point pairs, the LCCS searching loop is broken. If the M matching points are at the beginning of the trajectories, we only need $M * m$ steps to find the first M matching pairs. There will be $L = min(n - i_{stop}, m - j_{stop})$ more steps after the LCCS stops. Thus, the number of operations becomes $O(N_1 * N_2 * (M * m * C + L))$. M is normally much smaller than n, which reduces the total number of operations approximately by M/n. In our experiments, $(M/n_{avg}) < 0.2$. Thus, the running time and complexity is reduced significantly compared to the previous work in [5].

3. EXPERIMENTAL RESULTS

The proposed algorithm is tested on the trajectory data obtained from the video sequences in the PETS2001 database. One of the two sequences of each video set is delayed by a known offset. In this way, the ground truth for the frame offset is known for each experiment. In our experiments, we use $\epsilon = 20$ and $M = 5$.

The examples of the matched trajectories from two cameras are shown in Figure 1. As seen in Fig. 1(d), the algorithm is robust to errors of the background subtraction algorithm, which caused a zigzag-like trajectory. Table 2 shows the results obtained after the confidence check step together with the ground truth. Results obtained with the proposed algorithm and the previous method in [5] are displayed together. The proposed method, which provides significant improvement in terms of computational complexity, has comparable or better results with respect to our previous work. If background subtraction and location extraction results are more accurate, better results can be achieved with the proposed method.



(a) Trajectory on the 1st view (b) The match of the trajectory in (a)

(c) Trajectory on the 1st view (d) The match of the trajectory in (c)

Fig. 1. Examples of matched trajectories in two cameras

		Frame Offsets			
Vid. 1	Ground Tr.	300	500	800	1000
	Prev. Meth.	301	499	792	989
	Prop. Meth.	301	498	800	998
	Accuracy	99.67%	99.6%	100%	99.8%
Vid. 2	Ground Truth	300	500	800	1000
	Prev. Meth.	300	500	807	1000
	Prop. Meth.	299	495	795	999
	Accuracy	99.67%	99%	99.37%	99.9%

Table 2. The frame offsets obtained after the confidence check with the proposed method and the previous work.

4. CONCLUSIONS

We presented a computationally efficient and robust algorithm to match and align object trajectories from unsynchronized cameras, and thus to recover the frame offset. This method employs LCCS during the trajectory matching. Compared to the previous work in [5], which performs an exhaustive search, the proposed algorithm reduces the operation time by a factor of M/n_{avg}, where $M = 5$ in the experiments, and n_{avg} is the average trajectory length. While providing significant improvement in terms of computational complexity, the proposed algorithm has comparable or better results with respect to our previous work. It is reliable and robust to possible errors due to background subtraction or location extraction. After performing the experiments with different frame offsets and different video sequences, an average accuracy rate of 99.63% is achieved.

5. REFERENCES

[1] 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.

[2] L. Lee, R. Romano, and G. Stein, "Monitoring activities from multiple video streams: Establishing a common coordinate frame," *IEEE Trans. on PAMI*, pp. 758–767, August 2000.

[3] Y. Caspi, D. Simakov, and M. Irani, "Feature-based sequence-to-sequence matching," *VAMODS workshop*, 2002.

[4] T. Tuytelaars and L. Van Gool, "Synchronizing video sequences," *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. 762–768, 2004.

[5] S. Velipasalar and W. Wolf, "Frame-level temporal calibration of video sequences from unsynchronized cameras," *Machine Vision and Applications*, p. Online first version available, January 2008.

[6] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," *IEEE 18th Int'l Conf. on Data Engineering*, pp. 673–684, 2002.

[7] D. Buzan, S. Sclaroff, and G. Kollios, "Extraction and clustering of motion trajectories in video," *IEEE 17th Int'l Conf. on Pattern Recognition*, vol. 2, pp. 521–524, Aug. 2004.

[8] A. M. Cheriyadat and R. Radke, "Automatically determining dominant motions in crowded scenes by clustering partial feature trajectories," *First ACM/IEEE Int'l Conf. on Distributed Smart Cameras*, pp. 52–58, Sept. 2007.