Camera Alignment using Trajectory Intersections in Unsynchronized Videos

Thomas Kuo, Santhoshkumar Sunderrajan, and B.S. Manjunath
University of California, Santa Barbara
Santa Barbara, CA 93106
{thekuo,santhosh,manj}@ece.ucsb.edu

Abstract

This paper addresses the novel and challenging problem of aligning camera views that are unsynchronized by low and/or variable frame rates using object trajectories. Unlike existing trajectory-based alignment methods, our method does not require frame-to-frame synchronization. Instead, we propose using the intersections of corresponding object trajectories to match views. To find these intersections, we introduce a novel trajectory matching algorithm based on matching Spatio-Temporal Context Graphs (STCGs). These graphs represent the distances between trajectories in time and space within a view, and are matched to an STCG from another view to find the corresponding trajectories. To the best of our knowledge, this is one of the first attempts to align views that are unsynchronized with variable frame rates. The results on simulated and real-world datasets show trajectory intersections are a viable feature for camera alignment, and that the trajectory matching method performs well in real-world scenarios.

1. Introduction

Networks of wireless cameras are very useful in applications such as surveillance and intelligent environments [1] because they require less infrastructure. However, a wireless channel typically has less bandwidth and more disruptions such as dropped frames. These limits can be compensated for by varying both the quality of the image and the frame rate of the video according to the needed and available bandwidth.

These compromises make camera calibration and alignment more difficult. Low image quality hinders common approaches based on finding strong feature points, such as SIFT [8]. Variable frame rates affect methods that match points in trajectories, since these methods typically require frame-to-frame synchronization [19, 11, 16] and/or fixed frame rates [7, 3].

In this context, we contribute one of the first methods to align unsynchronized videos that have variable frame rates. We propose the novel use of the intersections of ground plane trajectories to find the homography between cameras. Additionally, to find corresponding intersections, we propose a new method for matching trajectories that represents trajectories in a Spatio-Temporal Context Graph (STCG).

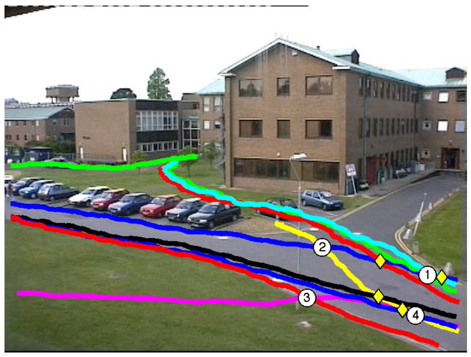
Our method, see Figure 1, starts with the ground plane trajectories of two views. Then, the spatial and temporal relationships between trajectories in one view are captured in an STCG, which we use to find the best matching trajectories in another view. Finally, the corresponding intersections of corresponding trajectories are used to compute a homography. Experiments show that our method performs as well as state-of-the-art methods on synchronized videos, and better on unsynchronized videos.

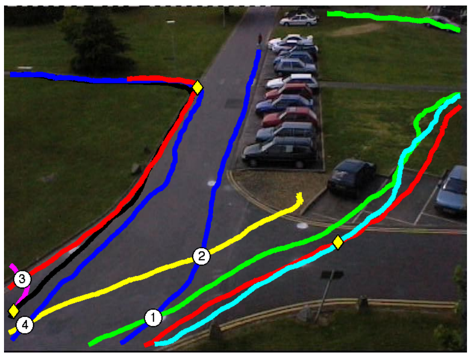
The paper is divided as follows: Section 2 overviews prior work in calibration, alignment, and synchronization. Section 3 describes STCGs and how to use them to match trajectories. Section 4 describes the procedure for using trajectory intersections to align camera views. In Section 5, we present results on both simulated and real-world datasets, and then discuss the advantages and disadvantages of the method in Section 6.

2. Prior Work

Camera calibration, image alignment, and synchronization has been well covered in the literature. Camera calibration and image alignment typically follows a pattern of finding a set of potential corresponding points across views, and then extracting a geometric model, e.g. a homography or fundamental matrix, using a RANSAC-based method. These corresponding points are usually found in static images, and matched using feature descriptors such as SIFT [8], SURF [2], and MSER [9]. Examples of this systems are explored by Mikolajczyk et al. [12] and Snavely et al. [18].

In videos with low image quality or wide-baselines, image features do not work well. Another class of methods finds corresponding points in the tracks of moving ob-


(a) Camera View 1


(b) Camera View 2

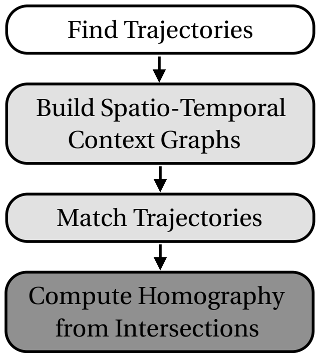

Find Trajectories
Build Spatio-Temporal Context Graphs
Match Trajectories
Compute Homography from Intersections
(c) Method flowchart

Figure 1: To find the homography between two camera views, such as these from the PETS 2001 Dataset [20], we find corresponding trajectory intersections across views by matching trajectories using Spatio-Temporal Context Graphs.

jects. Lee et al. [7] select object centroids across camera views that occur simultaneously in a small time window as potential corresponding points. Stauffer and Tieu [19] probabilistically match simultaneous points that have similar appearances. Both of these methods treat each object observation separately.

Other methods consider trajectories as a single unit. Caspi et al. [3] use a RANSAC variant to select corresponding trajectories and compute a homography or fundamental matrix from the observations in the trajectories. Meingast et al. [11] use bipartite graph matching to compute the epipolar constraints of synchronized cameras using the trajectories as features. Sheikh and Shah [16] also use a graph framework. They create a digraph that finds the maximum likelihood estimate of the inter-frame homographies.

Two of the these methods handle one form of synchronization: a constant time shift. Lee et al. [7] re-align the videos with different time shifts and select the shift with the least error. Caspi et al. [3] estimates the homography and time shift simultaneously in RANSAC.

The literature on synchronization overlaps with our goal of aligning unsynchronized videos. They also tend to assume videos with constant frame rate, and often require calibration. Whitehead et al. [21] formalize temporal synchronization in cases when different cameras have different constant frame rates. Their proposed synchronization method requires the camera geometry of 3 views, and first roughly synchronizes cameras using the points of maximum curvature. Then, they refine the synchronization to subframe accuracy using the epipolar lines of the inflection points. Pundik and Moses [15] also use epipolar lines from calibrated cameras by matching temporal signals along the epipolar lines. Wolf and Zomet [22] do not assume an existing calibration, but assume the videos have equal and constant frame rates. They synchronize views by rank constraints on matrices that capture either the linear combination between points in two views or the brightness measurements of image patches. Sinha and Pollefeys [17] simultaneously calibrate and synchronize cameras in a network, but require the silhouette of a person instead of the trajectory alone.

These methods are not designed for unsynchronized, variable frame rate videos. Most of the alignment methods require manual synchronization or only handle a constant time shift. Most of the synchronization methods assume constant frame rates or are dependent on an existing alignment.

This paper addresses unsynchronized cameras with variable frame rates. It proposes the novel use of trajectory intersections as corresponding points to align these views, and a method to match trajectories based on the the spatial and temporal context between neighboring trajectories. Unlike, the existing methods, these do not required constant frame rates or frame-to-frame synchronization.

3. Trajectory Matching using STCGs

Our approach requires as input ground plane object trajectories in each camera view. We express each trajectory j in camera i as a sequence of observations $T_j^i = \{(x_j^i[n], y_j^i[n], t_j^i[n])\}_{n\in\mathbb{N}}$, where $x_j^i[n]$ and $y_j^i[n]$ are the image coordinates of an observation and $t_j^i[n]$ is the timestamp of an observation. Typically, $t_j^i[n] - t_j^i[n-1]$ is not constant for all n, nor is it synchronized across views.

Methods for obtaining these trajectories are numerous and outside the scope of this work. In our experiments, we use Visual Tracking Decomposition (VTD) [6] and the Struck tracker [5].

Given the trajectories detected in each view, we match them across cameras in order to find corresponding tra-

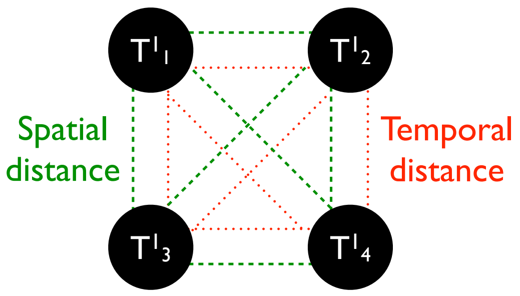


Figure 2: An STCG is an attributed, fully connected multi-graph with vertices that represent trajectories T_j^i and two edges between each vertex pair that represent the spatial distance (green, dashed lines) and the temporal distance (red, dotted lines) between the trajectories. We match trajectories across views by applying Balanced Graph Matching [4] on the graphs of the views.

jectory intersections that can be used to compute the homography. Our novel trajectory matching technique first codifies for each view the spatio-temporal relationships between the trajectories using a graph. Then we use existing graph matching techniques to match trajectories across views.

For each camera i, we build a fully-connected, attributed multi-graph $G^i = (V^i, E^i, A^i)$ with vertices $V^i = \{1, \ldots, N^i\}$, edges $E^i = \{(j,k)\}\ \forall j,k \in \{1,\ldots,N^i\}$, and attributes $A^i_{jk} = a \in \mathbb{R}$, like the example shown in Figure 2. The vertices are the trajectories in a camera, T^i_j, and each edge, (j,k), has an attribute that is the distance between trajectories j and k, $d(T^i_j, T^i_k)$. There are two edges between each pair of vertices. One represents a spatial distance, d^{spatial} and the other a temporal distance, d^{temporal}. The graphs are matched using the Balanced Graph Matching method by Cour et al. [4].

3.1. Spatial Trajectory Distances

Spatial trajectory distances measure the distance between trajectories in a view. Intuitively, trajectories that are near each other should remain nearby. However, different viewpoints may reveal different relationships between the trajectories, and some homographies may severely affect the arrangement of distances. In most surveillance networks where cameras are mounted with similar heights and orientations, these types of homographies are less likely.

We experimented with four commonly used spatial distance metrics [13]. They are defined in Table 1.

The Euclidean distance is computed as the average Euclidean distance between points on two trajectories. This requires that they have the same number of points, and thus the trajectory has to be resampled. In our experiments, we resample to 100 evenly-spaced points.

In the PCA+Euclidean distance, the (x,y) coordinates of the trajectories are transformed using PCA to capture 95% of the variation. The distance is the Euclidean distance between these coefficients. The trajectories must again be of equal length and so are resampled to 100 points.

Dynamic Time Warping (DTW) finds the optimal time warping that minimizes the total distance between matching points. Unlike the previous distances, DTW does not require that trajectories have the same length.

The Longest Common Subsequence (LCSS) distance determines the longest subsequence that is common to both trajectories. It also handles sequences of different lengths. The distance in Table 1 is based on the LCSS, which is found using the algorithm in Equation 1.

$$\text{LCSS}(A,B,\delta,\epsilon) = \tag{1}$$
$$\begin{cases} 0 & \text{if } A \text{ or } B \text{ is empty} \\ 1+\text{LCSS}(\text{Head}(A), & \text{if } \|a_n - b_m\| < \epsilon \\ \qquad \text{Head}(B),\delta,\epsilon) & \quad \text{and } |N-M| <= \delta \\ \max(\text{LCSS}(\text{Head}(A),B,\delta,\epsilon), & \text{otherwise} \\ \qquad \text{LCSS}(A,\text{Head}(B),\delta,\epsilon)) & \end{cases}$$

In this equation, A and B are trajectories, $\text{Head}(A) = A[0]$, the first element of the trajectory, and θ and ϵ are the temporal and spatial thresholds that denote when two trajectory points are close to each other.

3.2. Temporal Trajectory Distances

The temporal distance measures the time between trajectories, and, in the STCG, provides the temporal context between trajectories such as the approximate order. Intuitively, the trajectories should maintain a rough temporal order across views. The measurements may deviate, however, as different viewpoints may change the timing between trajectories.

The temporal distances are summarized in Table 1. The Start Time metric finds the absolute difference between the timestamps of the first observation in the trajectory. The Mean Time metric finds the absolute difference between the average times of each trajectory.

3.3. Balanced Graph Matching

To match the trajectories across views, we must find the one-to-one mapping between the trajectories that best preserves the spatial and temporal context between views. This is the goal of graph matching, and one of the start-of-the-art algorithms for graph matching is the Balanced Graph Matching method by Cour et al. [4]. It takes as input a compatibility matrix, W, that captures the similarity between edges.

Euclidean	$d_{\text{Euclidean}}(T_j^i, T_k^i) = \frac{1}{N^i}\sum_{n=1}^{N^i}\sqrt{(x_j^i[n]-x_k^i[n])^2+(y_j^i[n]-y_k^i[n])^2}$
PCA + Euclidean	$d_{\text{PCA}}(T_j^i, T_k^i) = \frac{1}{N^i}\sum_{n=1}^{N_\lambda^i} d_{\text{Euclidean}}(a_{i,n}, a_{j,n})$ where a_n is the PCA coefficients and N_λ is the number of eigenvalues retained
DTW	$d_{\text{DTW}}(T_j^i, T_k^i) = \frac{1}{N^i}\sum_{n=1}^{N^i} d_{\text{Euclidean}}(\phi_{i,n}, \phi_{j,n})\alpha_{n,\phi}$ where ϕ_i is a time warping function, and α is a weighting and normalization constant
LCSS	$d_{\text{LCSS}}(T_j^i, T_k^i, \delta, \epsilon) = 1 - \frac{\text{LCSS}(T_j^i, T_k^i, \delta, \epsilon)}{max(N^j, N^k)}$
Start Time	$d_{\text{Start Time}}(T_j^i, T_k^i) = \lvert t_j^i[0] - t_k^i[0]\rvert$
Mean Time	$d_{\text{Mean Time}}(T_j^i, T_k^i) = \lvert\text{mean}(t_j^i) - \text{mean}(t_k^i)\rvert$

Table 1: Spatial and temporal distances and their mathematical expressions

For cameras i and i' with M and M' number of edges respectively, W is an $MM' \times MM'$ matrix with each row representing the match j to j' and each column representing the match k to k'. In our method, W is defined by Equation 2, with each distance a value between 0 and 1.

$$\begin{aligned} W_{jj',kk'} &= S(E_{jk}^i, E_{j'k'}^{i'}) \\ &= f(A_{jk}^i, A_{j'k'}^{i'}) \\ &= \text{mean}\left(s^{\text{spatial}}, s^{\text{temporal}}\right) \end{aligned} \tag{2}$$

With s^{spatial} and s^{temporal} defined by Equations 3 and 4.

$$s^{\text{spatial}} = \min\left(\frac{d_i^{\text{spatial}}}{d_{i'}^{\text{spatial}}}, \frac{d_{i'}^{\text{spatial}}}{d_i^{\text{spatial}}}\right) \tag{3}$$

$$s^{\text{temporal}} = \min\left(\frac{d_i^{\text{temporal}}}{d_{i'}^{\text{temporal}}}, \frac{d_{i'}^{\text{temporal}}}{d_i^{\text{temporal}}}\right) \tag{4}$$

Balanced Graph Matching formulates the graph matching problem as the Integer Quadratic Program that is relaxed to a spectral matching with an affine constraint. It also normalizes the compatibility matrix such that scores for each edge sums to one, i.e. the bistochastic normalization of the edge similarity matrix.

4. Calibration using Trajectory Intersections

Using the trajectory matches determined from the previous section, we find the set of corresponding trajectory intersections across views. We use these intersections to find the homography using RANSAC.

To our knowledge, this is the first attempt to use the intersections of object trajectories to align cameras in a network. Intuitively, this method works since the ground plane intersections correspond to the same 3D point in different views, and can be found without perfectly synchronizing the observations of a trajectory.

To find the intersections of two ground plane trajectories, we use dynamic time warping to find the best distance between points in the trajectories. Next, we find local minima in the sequence of distances that are close to zero. These are potential intersections, that we then verify by fitting a third-degree polynomial to the matching points and their four neighbors in each trajectory. If the intersection of these polynomials is in this neighborhood, then this point is defined as an intersection.

Corresponding intersections are determined by the matched trajectories, i.e. if trajectories T_j^i and $T_{j'}^{i'}$ match and trajectories T_k^i and $T_{k'}^{i'}$ match, then the intersection of T_j^i and T_k^i matches the intersection of $T_{j'}^{i'}$ and $T_{k'}^{i'}$.

These intersections are inputted into a RANSAC-based method that finds the best homography. RANSAC takes 4 points, computes a homography, and then finds the set of inliers. The homography with the most inliers is the best estimate. Using this method helps to remove incorrect intersection matches that may have arisen from incorrect trajectory matches.

5. Results

Our methods are validated on several datasets. In the first dataset, we simulate trajectories in order to find the best combination of distance metrics. These metrics are applied to real world datasets. On synchronized datasets, (PETS 2001 and 2009), we hypothesize that our method should perform similar to existing methods. On unsynchronized datasets (wireless camera dataset), we hypothesize that our method will perform better than existing methods. Our method is compared against manually matched trajectory and Lee et al. [7] with given synchronization. Lee et al. is one of the state-of-the-art methods for camera alignment using object trajectories.

5.1. Simulated Dataset

Our initial experiments are on two simulated cameras that are pointed straight down. The camera views have a

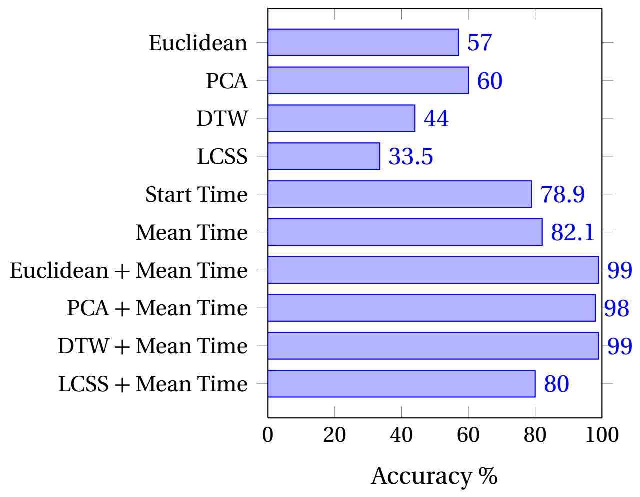


Figure 3: The accuracy of trajectory matching on simulated trajectories using different combinations of metrics.

4:3 aspect ratio and they overlap such that the right half of camera view 1 is the left half of camera view 2. Each experiment has a random set of 10 trajectories that move in a diagonal across the scene. The speeds and start times of each trajectory is also randomly assigned.

We tried different combinations of the distance metrics described in Section 3. Each combination is run 100 times with 10 trajectories each. We compute at the end the percent of trajectories that are correctly matched among the 1000 total trajectories.

The results are shown in Figure 3. They show that temporal metrics alone are much better than spatial metrics. This is likely because the trajectories are more spread out temporally than spatially. However, any combination of spatial and temporal metrics perform almost perfectly, except for the LCSS metric. The LCSS performs poorly because it is a measure of the percent of two trajectories that are near each other. Trajectories beyond this threshold have the same distance regardless of who far away they are. The DTW was an easier metric to compute, since Euclidean and PCA methods require resampling the trajectories. Therefore, we use the combination of DTW and Mean Time in the remaining experiments.

We also use this dataset to evaluate the effect of incorrect trajectory matches on the reprojection error of the final homography. When all trajectories are matched, the reprojection error of the homography is close to 0. Even when matching accuracy is at 40%, the homography can be recovered.

We also tried to enumerate all possible intersections pairs and apply RANSAC. This did not produce accurate results because the exponential explosion of possible matches hid the correct homography and correct matches.

5.2. PETS Datasets

The PETS 2001 Dataset, shown in Figure 1, consists of two views overlooking a roadway with 10 ground plane trajectories per camera view extracted by taking the bottom point of the bounding boxes of people. These trajectories are smoothed with a Kalman filter.

Vehicle trajectories are not used in this experiment. They skewed the homography results significantly, since points at the bottom of the bounding box did not correspond on the ground plane. This exemplifies a challenge of this method: the trajectories must accurately represent the ground plane point that correspond across views.

The trajectory matching method correctly identified 6 out of the 10 trajectories. It confused two pairs of trajectories. In both of these cases, the pair were walking together. Because they were traveling as a group, their trajectories were similar to other trajectories when compared spatially and temporally.

The approach found only 4 corresponding intersections between the two views. This is the minimum number necessary to compute a homography, and it reveals a challenge for using intersections. In some scenes, there are not a significant number of intersections, such as when viewing a corridor.

We verify the results by marking the five points marked on the ground and finding the error between the marked points and the homographic mapping from view 2 into view 1 and vice versa. The results are shown in Table 2.

The PETS 2009 S2-L1 Dataset, shown in Figure 4, consists of two views of the same roadway as PETS 2001, but from different viewpoints. From this dataset, we use views 1 and 8, which are far and near from the road intersection. We track 19 people through both views and smooth the results using a Kalman filter. The matching algorithm correctly matched 10 of the 19 trajectories. We choose as ground truth points the ends and center of the cross that are in the intersection. The reprojection errors of these points for the different methods are shown in Table 2.

5.3. Wireless Camera Dataset

The final dataset exemplifies a scenario that is challenging for existing methods. It has two 640-by-480 cameras that are wireless and view a walkway from opposite directions. As seen in Figure 5, the quality of video is such that there are few strong corners for traditional feature detectors. The video also has a variable frame rate that typically ranges from 15 to 21 frames per second, but occasionally dips to the single digits.

We extracted 24 trajectories from each camera using Visual Tracking Decomposition [6] and the Struck tracker

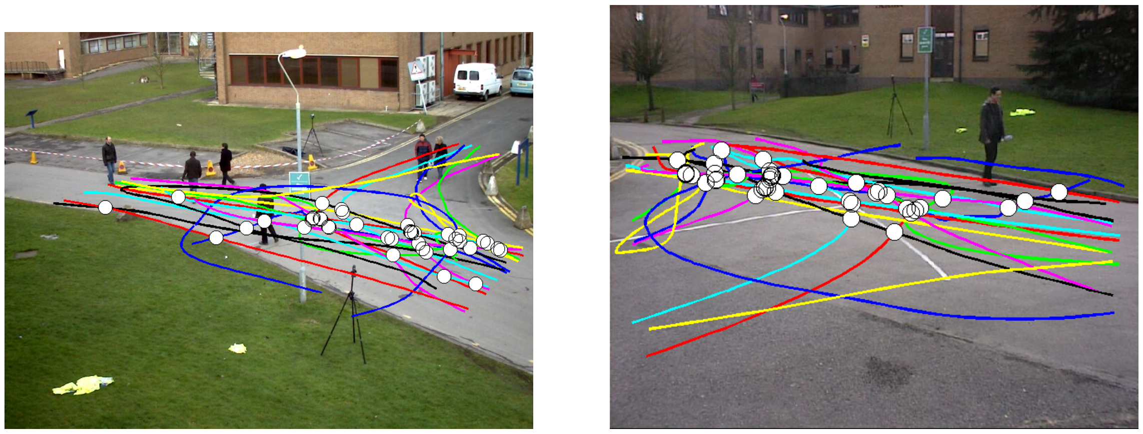

(a) Camera View 1 (b) Camera View 2

Figure 4: PETS 2009 Dataset. This dataset from PETS 2009 has 19 trajectories; 10 of which were matched correctly.

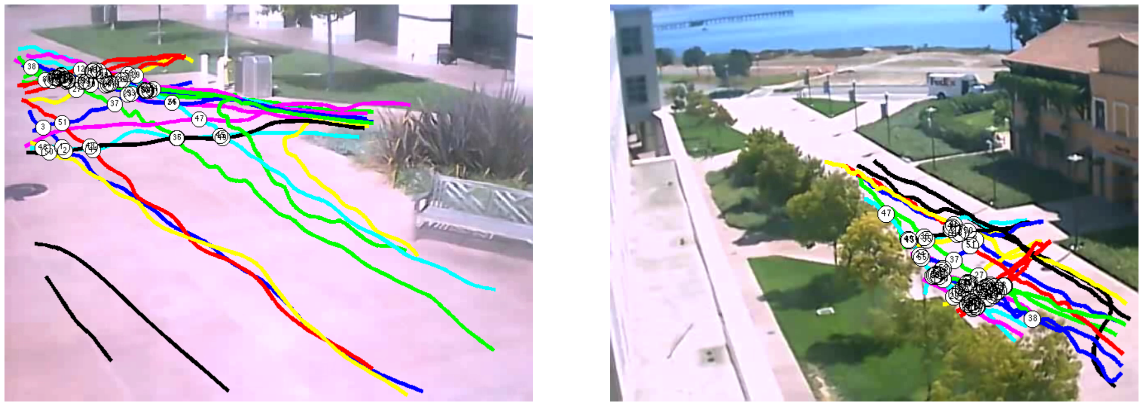

(a) Camera View 1 (b) Camera View 2

Figure 5: Wireless Camera Dataset. This dataset from wireless cameras have poor image quality and variable frame rates. Out of 24 trajectories, 22 were matched correctly and are color-coded in these views. The matched intersections are displayed as white circles and numbered.

[5]. These produce bounding boxes from which we use the central bottom position for the ground plane trajectory. These trajectories are smoothed by a Kalman filter.

Trajectory matching correctly matches 22 of the 24 trajectories. The two trajectories that were incorrectly matched were adjacent in time and start from the same area. We found 60 matching intersections across views. Three matching ground truth points from each image were selected. The reprojection errors for these points are shown in the final column of Table 2. These results show that calibration using intersections performs better than using the object points.

Thus matching trajectory intersections are more resilient to variable frame rate. We hypothesize that this should be true even as the frame rate decreases. Thus we compare the effect of decreasing frame rate with reprojection error between Lee et al. and our method in Table 3. We find that for Lee et al. the error increases slightly while the frame rate decreases, while for our method, the error stays about the same.

6. Discussion

The proposed methods of trajectory matching and alignment using trajectory intersections show promising results for the relatively new topic of aligning unsynchronized cameras with variable frame rates. The experiments suggest several conclusions and challenges that must be addressed in future work.

6.1. Advantages

Consistently and as hypothesized, alignment using manually matched trajectory intersections performs sim-

	PETS 2001		PETS 2009		Wireless Cameras	
	View 1 error (px)	View 2 error (px)	View 1 error	View 2 error	View 1 error	View 2 error
Lee et al.	12.92	20.41	9.14	8.52	90.72	8.78
Matched Intersections	13.45	21.67	9.19	8.38	19.27	7.2
Intersections w/ Matching	26.82	36.90	19.18	17.47	15.18	10.09

Table 2: Reprojection errors of the ground truth points for all datasets across three methods.

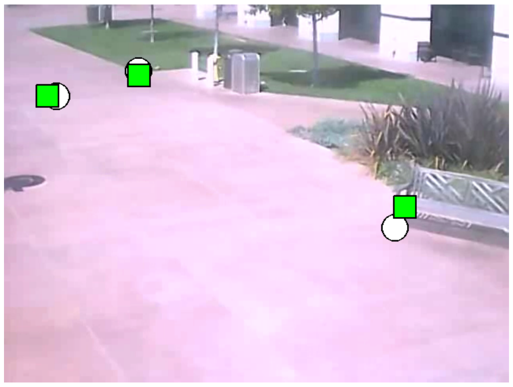


Figure 6: Camera View 1

Figure 7: The homographic projection of 3 points from camera view 2 (green squares). The are compared the ground truth points in camera view 1 (white circles).

Avg Frame Rate	View 1 error (px)	View 2 error (px)
Lee et al.		
19.1 Hz	90.72	8.78
9.5 Hz	96.65	9.2
6.3 Hz	123.83	9.8
Intersections w/ Matching		
19.1 Hz	31.09	12.08
9.5 Hz	21.26	9.59
6.3 Hz	29.93	13.54

Table 3: Reprojection error of ground truth points by decreasing the frame rate of the Wireless Camera Dataset.

ilar to Lee et al. in synchronized cameras, and better in unsynchronized cameras. Our method for matching trajectories matches 50% to 95% of the trajectories, but slightly affects the alignment errors.

In unsynchronized views, the proposed intersection-based approach performs better than Lee et al. because there are few truly simultaneous points across views. Nearly simultaneous points have slight differences in their actual 3D matches. We can see this effect when we decrease the frame rate, which increases the error for Lee et al. but does not effect the error of the intersection-based approach.

6.2. Challenges

The proposed method may not work for all video sequences. It is dependent upon a few critical factors such as reliable tracking and the existence and sufficient distribution of trajectory intersections.

Like all of the methods based on matching object trajectories [3, 11, 16], this method assumes that trajectories completely track the person across the view. Common tracking errors such as breaking one object track into two could lead to an incorrect trajectory match.

The corresponding 2D points in the trajectories must also match in 3D. For example, our tracks must be on the ground plane, which may be more difficult in views with odd angles. For example, in the PETS 2001 dataset, the vehicles were not used because the bottoms of the trajectories did not correspond to the center of gravity of the object, especially in view 2 where the camera is closer to the objects and points down.

This method also depends on the existence of trajectory intersections. In PETS 2001, there were only 4 intersection points, the minimum number of points needed for alignment. This hindered RANSAC's ability to compensate. Other potential examples include people in hallways and roadways whose trajectories are parallel and do not intersect. Some scenes may have intersections confined to a small region of the image, as is noted in each of the datasets.

Some of the inaccuracies produced by poor ground plane correspondence or confined intersections points may be controlled by using these results as an initial coarse alignment for a feature or region-based alignment scheme, as in Stauffer and Tieu [19]. Other issues, such as the reliable tracking, would require additional methods that would attempt to match broken trajectories or to compare the objects using appearance-based features.

When matching trajectories, the spatial and temporal distances do not completely capture the relationships between trajectories across views. Different viewpoints may reveal different parts of a trajectory and the spatial distances are affected by the homography, not invariant to them. This is a greater issue when the camera is closer to the trajectories, but in our experiments, such as PETS 2009, where cameras tend to be mounted farther away, the viewpoint change did not affect the contextual distances enough to severely compromise matching.

7. Acknowledgements

This work is supported by ONR/DURIP #N00014-08-1-0791, ONR #N00014-10-1-0478, and ONR #N00014- 12-1-0503.

8. Conclusion

We address the novel problem of aligning unsynchronized camera views that have low or variable frame rates. Our solution takes advantage of trajectory intersections, which can be found robustly, and a new trajectory matching method that does not require frame-to-frame synchronization.

First, trajectories across views are matched by matching Spatial-Temporal Context Graphs (STCG), which capture the distances between trajectories in time and space in a view. Through experimentation, we find that the Dynamic Time Warping spatial distance and Mean Time temporal distance produce the best results. Finally, camera views are aligned by finding and matching the intersections of trajectories.

Results show matching accuracy of using perfectly match intersection points to be similar to object-based alignment, and only slightly degraded by inaccurate matches using it scheme. The analysis of the results suggest a number of future extensions. The intersection points provide a rough homography that may be refined using the complete trajectory by, for example, modeling and match segments between intersection points. Other reference points, such as inflection points, could also be integrated into the framework. Other methods for matching the intersections or the trajectories, such as congealing [10] or cross ratios [14], may produce better results than the STCG described. Finally, this paper only explores the alignment of the camera views, but a natural extension would be to also synchronize the video using intersection points.

References

[1] H. K. Aghajan and A. Cavallaro, editors. *Multi-camera networks: principles and applications.* Academic Press, 1st edition, 2009.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006.

[3] Y. Caspi, D. Simakov, and M. Irani. Feature-Based Sequence-to-Sequence Matching. *IJCV*, 68(1):53–64, Mar. 2006.

[4] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. *NIPS*, 2007.

[5] S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270. IEEE, 2011.

[6] J. Kwon and K. M. Lee. Visual tracking decomposition. In *CVPR*, pages 1269–1276, 2010.

[7] L. Lee, R. Romano, and G. Stein. Monitoring activities from multiple video streams: Establishing a common coordinate frame. *PAMI*, 22(8):758–767, 2000.

[8] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, Nov. 2004.

[9] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, Sept. 2004.

[10] M. Mattar, M. Ross, and E. Learned-Miller. Nonparametric curve alignment. *ICASSP*, 2009.

[11] M. Meingast, S. Oh, and S. Sastry. Automatic Camera Network Localization using Object Image Tracks. In *ICCV*, pages 1–8. Ieee, 2007.

[12] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *IJCV*, 65(1-2):43–72, Oct. 2005.

[13] B. Morris and M. Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *CVPR*, pages 312–319. IEEE, 2009.

[14] W. Nunziati, S. Sclaroff, and A. Del Bimbo. Matching Trajectories between Video Sequences by Exploiting a Sparse Projective Invariant Representation. *PAMI*, 32(3):517–29, Mar. 2010.

[15] D. Pundik and Y. Moses. Video synchronization using temporal signals from Epipolar lines. In *ECCV*, 2010.

[16] Y. A. Sheikh and M. Shah. Trajectory association across multiple airborne cameras. *PAMI*, 30(2):361–7, Feb. 2008.

[17] S. N. Sinha and M. Pollefeys. Camera Network Calibration and Synchronization from Silhouettes in Archived Video. *IJCV*, 87(3):266–283, July 2009.

[18] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the World from Internet Photo Collections. *IJCV*, 80(2):189–210, Dec. 2007.

[19] C. Stauffer and K. Tieu. Automated multi-camera planar tracking correspondence modeling. In *CVPR*, pages 259–266. IEEE Comput. Soc, 2003.

[20] The University of Reading, UK. PETS datasets.

[21] A. Whitehead, R. Laganiere, and P. Bose. Formalization of the General Video Temporal Synchronization Problem. *Electronic Letters on Computer Vision and Image Analysis*, 9(1):1–17, 2010.

[22] L. Wolf and A. Zomet. Correspondence-free synchronization and reconstruction in a non-rigid scene. *Proc. Workshop on Vision and Modelling of Dynamic Scenes*, 2002.