Tracking Mean Shift Clustered Point Clouds for 3D Surveillance

Mark A. Keck Jr.
Dept. of Computer Science and Engineering
Ohio State University
Columbus, OH 43210 USA
keck@cse.ohio-state.edu

James W. Davis
Dept. of Computer Science and Engineering
Ohio State University
Columbus, OH 43210 USA
jwdavis@cse.ohio-state.edu

Ambrish Tyagi
Dept. of Computer Science and Engineering
Ohio State University
Columbus, OH 43210 USA
tyagia@cse.ohio-state.edu

ABSTRACT

We present in this paper a method of tracking multiple objects (people) in 3D for application in video surveillance. The tracking method is designed to work on images with objects at low resolution and has two major contributions. First we propose a way to generate 3D point clouds that imposes multiple constraints (both geometric and appearance-based) to ensure minimal noise in the 3D data. Second, we incorporate a method to group the points into clouds (or clusters) that correspond to objects in the environment being imaged. We show that this method is more powerful than current 3D tracking techniques that try to fuse 2D tracking information into 3D tracks. A comparison to competing 3D tracking methods are shown, and performance and limitations are discussed.

Categories and Subject Descriptors

I.4.5 [**Image Processing and Computer Vision**]: Reconstruction; I.5.3 [**Pattern Recognition**]: Clustering

General Terms

Algorithms

Keywords

3D tracking, reconstruction, mean shift clustering.

1. INTRODUCTION

The human visual system has the striking ability to temporally associate, or *track* objects. However, this has proven difficult in computer vision, and as such tracking has become a classic problem in the field. Tracking in the surveillance domain is often solved with one of a few existing methods.

Kalman filters [3,7,13,14] assume the object being tracked is driven by a linear process and the filter, based on observations, finds the optimal state sequence of the hidden process. Often this scheme is extended to estimate state sequences of nonlinear processes with Extended Kalman filters (EKFs) at the cost of suboptimal estimation of state sequences. Kalman filters and EKFs have one simplifying assumption: they assume that the posterior is Gaussian distributed. This assumption does not hold for all processes, and motivated the creation of particle filters [9] which have been used extensively in tracking of contours and shapes. More recently, kernel-based techniques for tracking have also become popular. Mean shift tracking [5,6] is very attractive to the vision community because of its computational efficiency compared to the other techniques described, however, it lacks some of the robustness that the previous approaches have.

All of the aforementioned algorithms are more often than not applied only in the image plane (in 2D). There have been steps toward extracting 3D information from existing frameworks or using multiple cameras to improve tracking results (e.g. [2, 11, 19]), but these frameworks focus on combining results from 2D trackers into 3D information and therefore still suffer from all the problems of 2D tracking.

To get around these problems we propose a 3D reconstruction of the foreground objects and track those objects in three dimensions, circumventing the problems inherent in combining multiple 2D trackers. The work proposes a novel methodology on reasoning about 3D reconstruction and the grouping of these point clouds. Experimental results are shown and compared to other approaches.

The remainder of the paper is organized as follows: Sect. 2 will discuss related work in more detail. In Sect. 3 we will describe our method of matching points in two camera views and generating 3D point clouds by employing multiple geometric constraints and continues with Sect. 4 discussing how we group the point clouds generated. We will go over experimental methods in Sect. 5 and discuss the findings. Concluding remarks are given in Sect. 6.

1.1 Notation

In this paper, we will use the following notational standards. Non-image matrices will be denoted with capital boldface (e.g. $\mathbf{A}$) and vectors will be denoted with lowercase boldface (e.g. $\mathbf{b}$). The i^{th} row of a matrix $\mathbf{A}$ will be denoted $\mathbf{a}_i$. Image matrices will be denoted with calligraphic font, like $\mathcal{I}$.

Often in this paper we will refer to two cameras, and there-

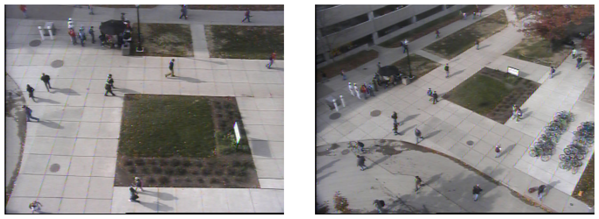

(a) (b)

Figure 1: (a) View from first camera. (b) View from second camera.

fore two images. We will refer to objects in the first camera/image with no accentuation, for instance image $\mathcal{I}$ or image point $\mathbf{m}$. However, we will denote objects related to the second camera/image with the $'$ symbol, like image $\mathcal{I}'$ or image point $\mathbf{m}'$.

We will denote homogeneous coordinates with the tilde symbol, i.e. if $\mathbf{x} = [x \quad y]^\top$, then $\tilde{\mathbf{x}} = [x \quad y \quad 1]^\top$.

2. RELATED WORK

As discussed in Sect. 1 there have been many proposed tracking frameworks, and most are based on either the Bayesian approach to dynamic state estimation [1,3,7,9,13,14] or are based on kernel density estimation [5,6]. As these techniques are all fairly well known in the community, we would only like to comment on the fact that in almost all cases they are only used in single view.

More recently 3D trackers have started to appear. Some trackers focus more on the finer level of tracking with high resolution targets [12]. However, the primary focus of this system is surveillance and therefore we are interested in a coarser level of tracking.

There have been some systems focused in this domain that have used multiple sensors to improve results. In [19] shape features were extracted from blobs in a single view and these features were associated from frame to frame. The tracking algorithm was extended in the paper using an EKF and multiple cameras to help with occlusion. In [2] a similar approach was proposed, using epipolar transfer to match centroids of foreground objects in two and three views and found the corresponding 3D point for the match sets. This 3D data was then pushed through a Kalman filter for tracking. Both of these systems work in many cases, but when two foreground objects are a single blob in all images, these systems fail (although it sounds like a special case, this is actually quite common).

Other work exists that locates the positions of people on the ground plane using homographies [11]. This can, with a large number of cameras, give accurate counts of the number of pedestrians in a scene and reliably attain the position of their feet. However, this approach suffers greatly when shadows are abundant (typical in natural scenes).

We propose a semi-dense 3D reconstruction (only reconstruct the surfaces of foreground objects) followed by an intelligent clustering of the resulting 3D points, and then using the centroids of these 3D clusters as the observation input to a Kalman filter for tracking.

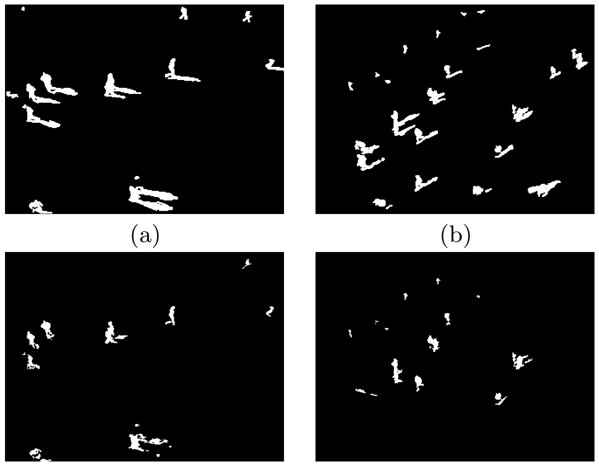

(a) (b) (c) (d)

Figure 2: (a) Background subtraction from first view. (b) Background subtraction from second view. (c) Shadow removal from first view. (d) Shadow removal from second view.

3. 3D POINT CLOUD GENERATION

In this section we will discuss the steps necessary to generate a 3D reconstruction. From here forward, we assume that we have *only* two cameras.

3.1 Calibration & Structure from Motion

The first step in the system is to calibrate the cameras offline. We accomplish this using the algorithm from [15]. This results in knowledge of the intrinsic camera matrices which we denote as $\mathbf{K}$ and $\mathbf{K}'$ for cameras one and two respectively. Once the cameras are calibrated, we deploy them in an outdoor area. Example viewpoints from two cameras with overlapping views are displayed in Fig. 1.

The next step is to determine the relative orientation of the cameras. We use a basic structure and motion technique similar to [17]. However, because the cameras are assumed to be wide baseline an automatic matching approach like that of [18] cannot be robustly employed to recover the epipolar geometry. One could implement an approach similar to [16] where features invariant to photometric and geometric changes are extracted in the image and then matched, but we manually selected anchor points in the scene to avoid noise in this stage affecting our results. This process extracts the relative rotation and translation between the two cameras, which we will denote $\mathbf{R}$ and $\mathbf{t}$ respectively.

To take advantage of more geometric information, we also assume that at least four of the correspondences among the views are known to be on the ground plane, so that the homography between ground planes, $\mathbf{H}$, can be estimated such that for all homogeneous points $\tilde{\mathbf{x}}$ (where $\mathbf{x} = [x \quad y]^\top$ are coordinates from the first image)

$$s\tilde{\mathbf{x}}' = \mathbf{H}\tilde{\mathbf{x}} \tag{1}$$

where s is an arbitrary scale factor. We estimate $\mathbf{H}$ using the standard least squares technique.

3.2 Shadow Removal

For each camera, a mean background model is estimated

Algorithm 1 Projective Shadow Removal

```
1: procedure ShadowRemoval($\mathbf{H}$,$\mathcal{I}$, $\mathcal{I}'$,$\mathcal{F}$, $\mathcal{F}'$, $T$)
2:     Let $\mathcal{L}_r$,$\mathcal{L}_g$,$\mathcal{L}_b$ denote the red, green, and blue planes of image $\mathcal{L}$
3:     $\hat{\mathcal{I}} \leftarrow$ Warp($\mathcal{I}$, $\mathbf{H}$)            ▷ Warp view 1 to view 2
4:     $\hat{\mathcal{F}} \leftarrow$ Warp($\mathcal{F}$, $\mathbf{H}$)    ▷ Warp foreground 1 to view 2
5:     $\mathcal{D} \leftarrow \mathcal{I}' - \hat{\mathcal{I}}$
6:     $\mathcal{M} \leftarrow \mathcal{F}' \vee \hat{\mathcal{F}}$                ▷ Create a mask
7:     $\mathcal{C} \leftarrow \sqrt{\mathcal{D}_r^2 + \mathcal{D}_g^2 + \mathcal{D}_b^2} \times \mathcal{M}$
8:     $\mathcal{S} \leftarrow \mathcal{C} > T$
9:     $\mathcal{F}'_r \leftarrow \mathcal{F}' \wedge \neg\mathcal{S}$
10:    return $\mathcal{F}'_r$
11: end procedure
```

over 60 images. To extract foreground objects from the images, basic background subtraction using the mean background model is performed:

$$\mathcal{F} = |\mathcal{I} - \mathcal{B}| > T \tag{2}$$

where $\mathcal{F}$ is the foreground image, $\mathcal{B}$ is the background image, $\mathcal{I}$ is the input, and T is a manually chosen threshold. Although this is a very simple background subtraction technique, we will show that it is still effective (other methods can be used). Example foreground images are shown in Fig. 2(a) and (b). Notice that in these images shadows appear strong in the foreground. Just as in many other applications, shadows in the foreground of these images will cause problems later in the pipeline.

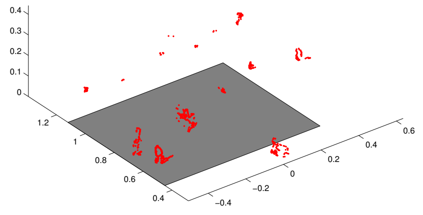
To remedy this we attempt to remove shadows using a technique similar to that in [10]. The input to the algorithm is the homography $\mathbf{H}$ between the two ground planes, the input images $\mathcal{I}$ and $\mathcal{I}'$ and the original foreground images $\mathcal{F}$ and $\mathcal{F}'$ and a threshold T. In this algorithm, the notation "$\times$" is an element-wise multiplication. A description of the algorithm in Alg. 1 follows.

As a preprocessing step, we use image warping to remove areas in each of the foreground images that can only be seen in one view (this allows us to reason on only foreground objects seen in both views). We also balance the luminance across the images. We then warp both the original image and the foreground image from view one into view two. We get the image difference of image two and the warped image in the color space. We can then calculate the "distance" between these two images by squaring each plane element-wise, and taking the square root. With this distance image, we want only to examine the elements that are in the foreground in either image, so we make a mask from the foreground of image two and the warped foreground of image one. We then mask the distance image with this and find elements that are smaller than threshold T. Anything that passes this threshold is assumed to be a shadow because it has similar color in both the warped and original image, which means it is very likely to lie on the ground plane.

Although in [10] the authors utilize an appearance model for shadows, we found comparable performance with this algorithm using a single threshold. An example of shadow removal is shown in Fig. 2(c) and (d).

3.3 Image Matching

With two foreground images (without shadows), we now find correspondences among the foreground objects. To do this we employ a version of epipolar search. Since we know


(a)

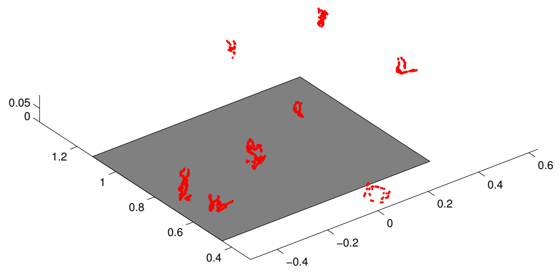

(b)

Figure 3: (a) Matching via simple epipolar search. (b) Matching via our epipolar search strategy using two geometric constraints.

the fundamental matrix $\mathbf{F}$ from our estimation of the motion between cameras, for any point of interest $\tilde{\mathbf{x}}$ in image $\mathcal{I}$ we can determine the line $\mathbf{l}'$ in image $\mathcal{I}'$ on which the corresponding point $\tilde{\mathbf{x}}'$ lies with the following formula:

$$\mathbf{l}' = \mathbf{F}\tilde{\mathbf{x}} \tag{3}$$

Epipolar search is a common technique and a good reference on epipolar geometry and search can be found in [8]. Typical implementations, however, often employ color histograms as a metric for matching points across views. Due to the low resolution of the foreground objects and noise, this constraint alone generates many false matches.

However, if the objects move with respect to some dominant ground plane (which we assume exists to remove shadows) and the cameras are relatively far from this ground plane (which is often the case in surveillance), we can impose a second geometric constraint to limit our search instead of a constraint based solely on appearance. We assume that any point of interest on an object in $\mathcal{I}$ (because it is moving with respect to a dominant ground plane) will be transformed "near" the corresponding point in $\mathcal{I}'$ via the homography $\mathbf{H}$. Knowing this will allow us to limit our search to a much smaller area, and in general it is important to note that in these types of applications geometric constraints are *far* more robust than appearance-based constraints alone (although using appearance based features to fine-tune matches after this stage may help disambiguate multiple matches).

The goal is then to find the point along the epipolar line $\mathbf{l}'$ in $\mathcal{I}'$ that is within radius ρ of point $\tilde{\mathbf{x}}' = \mathbf{H}\tilde{\mathbf{x}}$ that is most similar to $\tilde{\mathbf{x}}$ in appearance. The radius ρ is in pixels, and depends on the distance from the camera to the ground plane. In our case, we chose the number 20 for all experiments. In Fig. 3(a) we show an example of basic epipolar search without a second geometric constraint. In Fig. 3(b) we show our approach to epipolar search to illustrate the utility of the added constraint which reduces the number of noisy matches significantly. It is also important to note that all of the points generated in this plot correspond to people.

We apply this matching method first using view one as the cue and searching in view two, then reverse the process using view two as the cue and searching in image one for matches. This results in a denser reconstruction, which is beneficial for clustering.

Once matches are recovered from the images, we use basic linear triangulation to reconstruct the 3D points from the matches. The linear solution for a pair of matching points $\mathbf{m} = [u_1, v_1]^\top$ and $\mathbf{m}' = [u_2, v_2]$ from image one and image two, respectively, can be formulated [17] as follows

$$\begin{bmatrix} \mathbf{k}_1 - u_1\mathbf{k}_3 \\ \mathbf{k}_2 - v_1\mathbf{k}_3 \\ \mathbf{b}_1 - u_2\mathbf{b}_3 \\ \mathbf{b}_2 - v_2\mathbf{b}_3 \end{bmatrix} \tilde{\mathbf{p}} = \begin{bmatrix} 0 \\ 0 \\ (u_2\mathbf{k}'_3 - \mathbf{k}'_1)\mathbf{t} \\ (v_2\mathbf{k}'_3 - \mathbf{k}'_2)\mathbf{t} \end{bmatrix} \tag{4}$$

where matrix $\mathbf{B} = \mathbf{K}'\mathbf{R}$. We can rewrite Eqn. 4 as $\mathbf{Z}\tilde{\mathbf{p}} = \mathbf{z}$. The linear solution to $\tilde{\mathbf{p}}$ is

$$\tilde{\mathbf{p}} = \mathbf{Z}^\dagger \mathbf{z} \tag{5}$$

where $\mathbf{Z}^\dagger$ denotes the pseudo-inverse $(\mathbf{Z}^\top\mathbf{Z})^{-1}\mathbf{Z}^\top$.

We perform this reconstruction for each of the matches found, and as a result get a set of 3D points in space corresponding to our foreground objects.

4. MEAN SHIFT CLUSTERING

Point clouds like those generated in Sec. 3.3 cannot be input directly to a tracking system. We need a much more succinct representation for observations of foreground objects. To do this we cluster the 3D points using mean shift [4].

Mean shift clustering allows one to groups a set of points without knowing the number of clusters *a priori*. When using this clustering technique it is important to address two issues: the metric of the feature space and the shape of the kernel. For our purposes, fortunately, the feature space is Euclidean since it is a reconstruction of real 3D objects (up to a uniform scale factor). We select the standard Epanechnikov kernel, which guarantees convergence [4]. The Epanechnikov kernel has the profile

$$k_E(x) = 1 - x, \quad 0 \leq x \leq 1 \tag{6}$$

The profile is 0 when x is outside the designated range. This yields the radially symmetric kernel

$$K_E(\mathbf{x}) = \frac{1}{2}c_d^{-1}(d+2)(1 - ||\mathbf{x}||^2), \quad ||\mathbf{x}|| \leq 1 \tag{7}$$

where $\mathbf{x}$ is a data point in d-dimensional space (in our case this is 3), and c_d^{-1} is the volume of the d-dimensional unit sphere. This as well evaluates to 0 when $||\mathbf{x}|| > 1$.

We also must select the bandwidth of our clustering technique. This depends on the result of the structure and motion parameters extracted in Sec. 3.1. Structure and motion will return the relative orientation and translation up to an unknown scale, which in turn makes our metric reconstruction from Sect. 3.3 up to an unknown scale factor. This means that this parameter will have to be adjusted so that humans are not clustered together, but also so that a single human is not broken into smaller pieces in 3D.

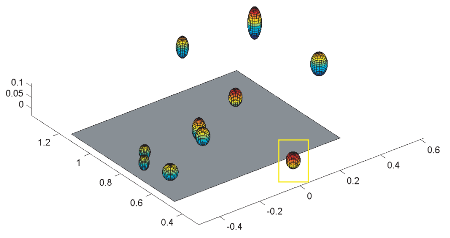


Figure 4: A mean shift clustering of the point clouds in Fig. 3(b)

We then randomly take half of the points, cluster these, and keep those clusters as the input to our tracking system. Each cluster then has a 3-vector as an observation (the (x, y, z) position of the cluster). A sample clustering is shown in Fig. 4. Note that all clusters shown in this image correspond to exactly one person save the one highlighted by the yellow box. That cluster merged two people together.

We use the cluster centroids as input observations to a Kalman filter tracker in 3D. Kalman filters are represented in state-space form, which is a set of two equations: (1) the state equation (Eqn. 8), which models the underlying latent process and (2) the observation equation (Eqn. 9) which relates the process to observable phenomena.

$$\mathbf{x}_{k+1} = \mathbf{G}\mathbf{x}_k + \mathbf{w}_k \tag{8}$$

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{9}$$

In tracking with Kalman filters, the state vector is often the position and velocity of the object being tracked, which we also use.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} \tag{10}$$

We also use the standard matrices $\mathbf{G}$ and $\mathbf{H}$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The observation and state noise covariance matrices are set experimentally. With our system identified, we can apply the Kalman recursions and filter observations.

We still, however, have the problem of associating the proper observation from an incoming frame pair (a cluster) with the corresponding filter. A natural solution, since we are dealing with filtering trajectories, is to use the distance between the estimated position at time t for each filter i, which is found with the following formula:

$$\hat{\mathbf{z}}_t^i = \mathbf{HG}\mathbf{x}_{t-1}^i \tag{11}$$

and compare this estimate with the incoming observations using the Euclidean distance. We then associate an observation with the filter corresponding to the smallest distance. As there are possible conflicts, we first choose the smallest value for all observations, associate that to the proper filter, and then remove all other distance measurements generated by that filter and repeat the process until all observations are either assigned, or are deemed unassignable (i.e. they are too far away from any available filter).

For those observations which are not assigned, if they are near the "edge" of the images (in this case "edge" does not refer to the normal edges of images, but the edge of the area that is seen in both views) then a new filter is generated and seeded with that starting position. If the observation is not near the edge, it is ignored (assumed to be noise).

5. EXPERIMENTS

To evaluate the method outlined above, we applied it to five different outdoor sequences. Each sequence was captured with a Matrox Quad digitizer so that two channels could be recorded simultaneously. The sequences were captured at the full 640×480 resolution at a full 30 Hz. In each sequence, sets of corresponding points ($\sim$ 35) were manually extracted from sequence pairs to estimate the epipolar geometry.

We selected sequences for validation and testing that traditional trackers, as well as 3D trackers based on fusing 2D information [2,19], would fail.

Resulting video frames from the tracked sequences are shown in figures Figs. 5, 6, 7. First in Fig. 5 we validate the tracking approach with a simple sequence. In this sequence, you can see that the pedestrian is tracked through the wide-baseline views accurately. We show frames from both views.

In Fig. 6 we show the tracker functioning in an environment with a high volume of pedestrians. As a reminder, the approach only tracks the pedestrians that can be seen in *both* views. In this case we can see that the 3D tracker performs well. Although in frame 50, you can see that tracker 10 started to die off, it has started to recover by frame 100 and returned near the target. This loss of target occurred due to background subtraction. With such low resolution from the second camera, it was difficult to robustly extract point clouds from the two people being tracked by trackers 10 and 11, and often only on observation was extracted for the two people.

Finally, in Fig. 7, we show the most general case. In this example, any 2D tracker (or any 3D tracker based on the fusing of 2D information) will fail because all people are low-resolution, have unreliable appearance, and are all merged into a single blob in both views. To show that 2D trackers would fail, we implemented the exact same type of tracker from our 3D case in 2D and based the observations on blobs instead of on 3D clusters. The results are shown in Fig. 8 (these are the same frames shown in Fig. 7). We have shown the final images of the two sequences with the tracks discovered from the tracker overlaid.

By looking at these two figures (Fig. 7 and 8) showing the same sequence with the two different algorithms, one can see that the 3D tracking algorithm is much more robust to occlusion. Although the tracks look erratic (jagged) the tracker is able to observe all three people in the track and keep the filters alive. However, in Fig. 8 we see that in both views, two of the three trackers initialized on the group of three people die out very quickly while one tracker seems to lock on and count them as a single person. Even if one attempts to intelligently combine these 2D trackers across views to extract 3D information, that information will be unreliable. This is just a single example to provide evidence to the fact that 2D trackers will always fail when an object is merged with another object (or objects) in every single view.

However, this comes at a price. First of all, the algorithm is obviously much slower than algorithms based on fusing 2D information to get 3D tracks. There are two very time-consuming processes in the system.

The first such process is shadow removal. It requires multiple image warps, which is very time consuming. We first tried to implement more standard techniques to remove shadows, such as those based on normalized color space, but got very poor results, and turned to this method for its robustness since our matching method will return extremely poor results without shadow removal.

The second major bottleneck in the system is the matching process itself. For every point in every foreground edge a search along an epipolar line in the corresponding image must occur to find a match. This is time consuming and in future work could have complexity reduced by adding more cameras.

Furthermore, it is evident in sequences we have not shown that shadows are not being completely removed. This is because the shadow removal system will not perform well when the shadow cannot be seen in both views. This will allow more noise points in the reconstruction stage, making clusters more noisy and affecting the location of the 3D centroid somewhat. Again, in future work this issue could be resolved by adding more cameras.

6. CONCLUSION

In this paper we presented a technique for tracking humans using two cameras. The technique employs a specialized shadow removal technique that uses the geometry of the scene and a specialized epipolar matching technique that utilizes two geometric constraints (epipolar and homographic) due to the nature of geometric constraints being more robust than those based on appearance. Point clouds of foreground objects are generated and then clustered and the centers of these clusters are used as observations to a Kalman filter tracking system.

The approach was then tested and results were shown. It is apparent from these results that other 3D trackers based on fusing 2D results and information cannot handle cases where people overlap in multiple camera views.

Future work will be focused on adding more cameras to speed up the system and to achieve more accurate results.

7. ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under grant No. IIS-0428249.

8. REFERENCES

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002.

[2] J. Black and T. J. Ellis. Multi camera image tracking. *Image and Vision Computing*, 2005.

[3] K. J. Bradshaw, I. D. Reid, and D. W. Murray. The active recovery of 3d motion trajectories and their use in prediction. *Pattern Analysis and Machine Intelligence*, 19(3):219–234, 1997.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, May 2002.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 142–151, Hilton Head, SC, USA, 2000.

[6] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, May 2003.

[7] F. Dellaert and C. Thorpe. Robust car tracking using kalman filtering and bayesian templates. In *Conference on Intelligent Transportation Systems*, 1997.

[8] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[9] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[10] K. Jeong and C. Jaynes. Moving shadow detection using a combined geometric and color classification approach. In *IEEE Wkshp. on Motion and Video Computing*, Breckenridge, CO, USA, Jan. 2004.

[11] S. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *European Conference on Computer Vision*, 2006.

[12] L. Lu, X.-T. Dai, and G. Hager. A particle filter without dynamics for robust 3d face tracking. In *Conf. on Computer Vision and Pattern Recognition Workshop*, volume 5, 2004.

[13] O. Masoud and N. Papanikolopoulos. A novel method for tracking and counting pedestrians in real-time using a single camera. *IEEE Transactions on Vehicular Technology*, 50.

[14] R. Rosales and S. Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. Santa Barbara, CA, USA, 1998.

[15] R. Tsai. A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, Aug. 1987.

[16] M. Vergauwen et al. Wide-baseline 3d reconstruction from digital stills. In *Int. Wkshp. on Visualization and Animation of Reality-based 3D Models*, Engadin, Switzerland, Feb. 2003.

[17] Z. Zhang. A new multistage approach to motion and structure estimation by gradually enforcing geometric constraints. In *ACCV (2)*, pages 567–574, 1998.

[18] Z. Zhang, R. Deriche, O. D. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1-2):87–119, 1995.

[19] Q. Zhou and J. K. Aggarwal. Object tracking in an outdoor environment using fusion of features and cameras. *Image and Vision Computing*, 2005.

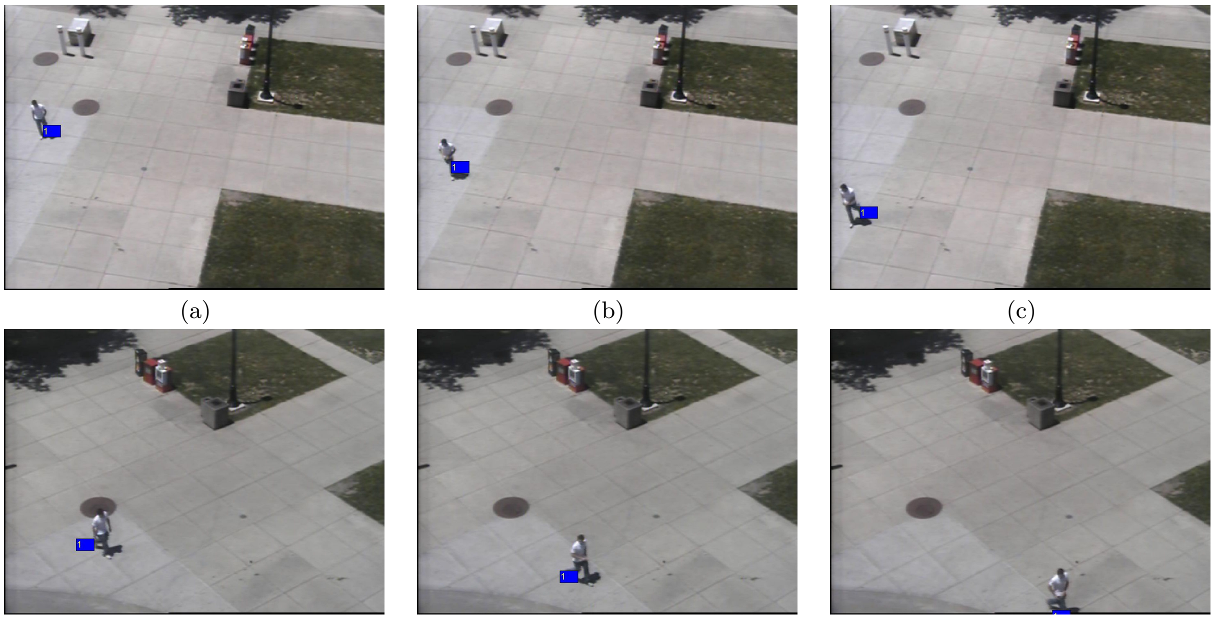


Figure 5: Results from test sequence one. (a) Frame 1/Cam 1 (b) Frame 50/Cam 1 (c) Frame 100/Cam 1 (d) Frame 1/Cam 2 (e) Frame 50/Cam 2 (f) Frame 100/Cam 2.

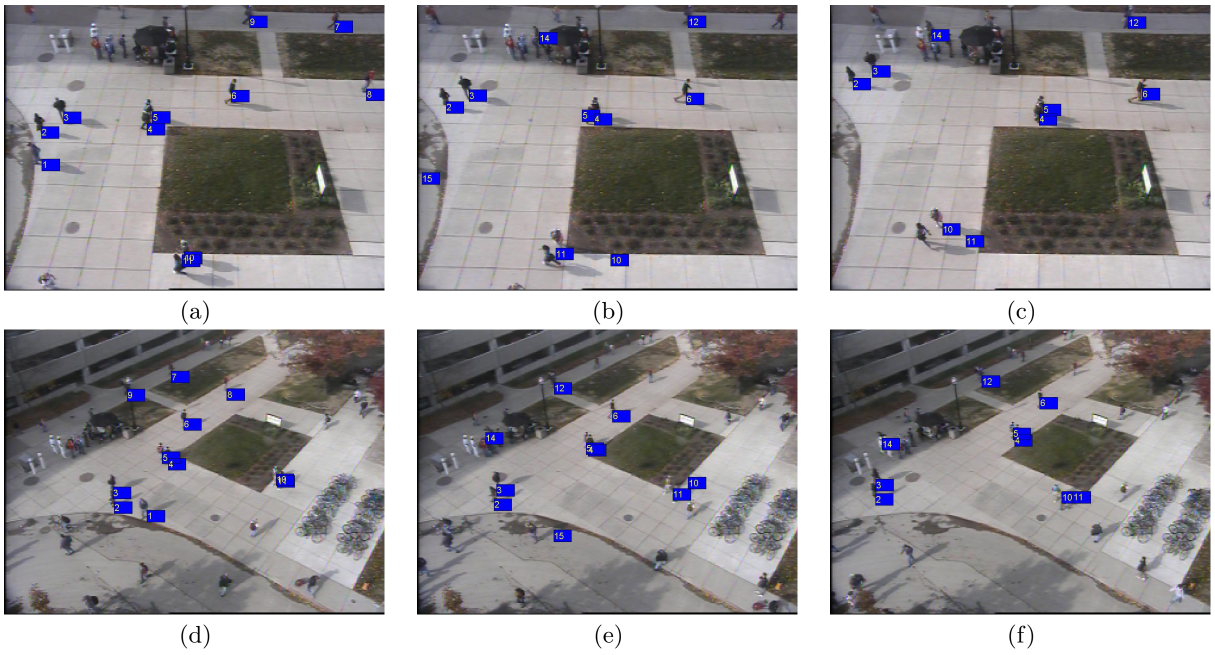


Figure 6: Results from test sequence two. (a) Frame 1/Cam 1 (b) Frame 50/Cam 1 (c) Frame 100/Cam 1 (d) Frame 1/Cam 2 (e) Frame 50/Cam 2 (f) Frame 100/Cam 2.

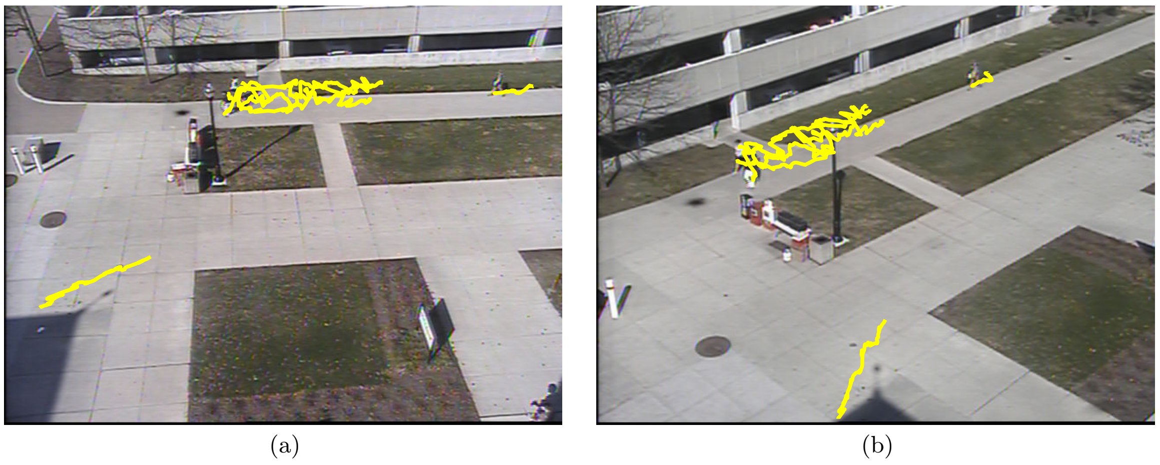


Figure 7: Results from test sequence three. The full tracks are shown from both views to compare with tracks generated from only the 2D trackers. (a) Tracks in the first view. (b) Tracks in the second view.

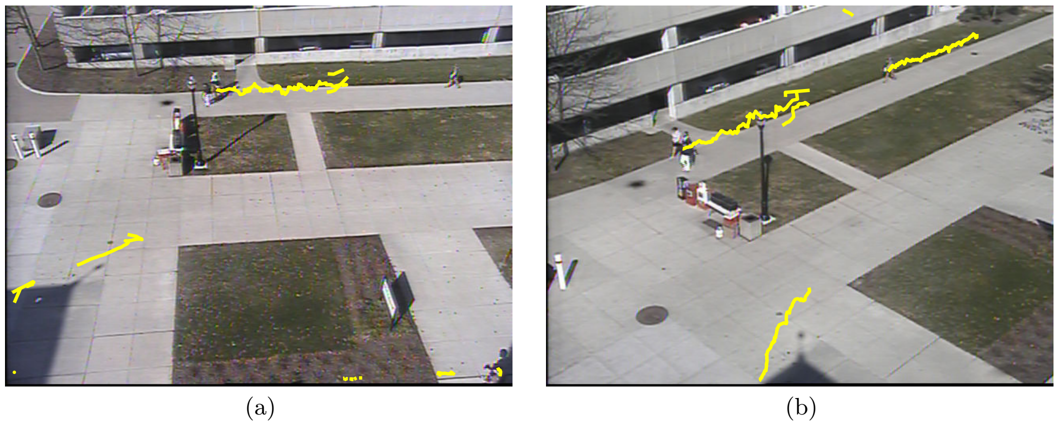


Figure 8: Results from test sequence three from the 2D tracker. Compare the tracks in these images to those in Fig. 7. (a) Tracks in the first view. (b) Tracks in the second view.