Tracking in Uncalibrated Cameras with Overlapping Field of View

Sohaib Khan, Omar Javed, Mubarak Shah
Computer Vision Lab
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816
{ khan, ojaved, shah}@cs.ucf.edu

ABSTRACT

To track people successfully in multiple cameras, one needs to establish correspondence between objects captured in each camera. We present a system for tracking people in multiple uncalibrated cameras. The system is able to discover spatial relationships between the camera fields of view and use this information to correspond between different perspective views of the same person. We employ the novel approach of finding the limits of field of view (FOV) of a camera as visible in the other cameras. Using this information, when a person is seen in one camera, we are able to predict all the other cameras in which this person will be visible. Moreover, we apply the FOV constraint to disambiguate between possible candidates for correspondence. Tracking in each individual camera needs to be resolved before such an analysis can be applied. We perform tracking in a single camera using background subtraction, followed by region correspondence. This takes into account the velocities, sizes and distance of bounding boxes obtained through connected component labeling. We present results on sequences taken from the PETS 2001 dataset, which contain several persons and vehicles simultaneously. The proposed approach is very fast compared to camera calibration based approaches.

Keywords: Tracking, tracking in multiple cameras, multi-perspective video, correspondence, surveillance, camera handoff, sensor fusion

1. INTRODUCTION

Tracking humans and vehicles is of interest for a variety of applications such as surveillance, activity monitoring and gait analysis. With the limited field of view (FOV) of video cameras, it is necessary to use multiple, distributed cameras to completely monitor a site. Typically, surveillance applications have multiple video feeds presented to a human observer for analysis. However, the ability of humans to concentrate on multiple videos simultaneously is limited. Therefore, there has been an interest in developing computer vision systems that can analyze information from multiple cameras simultaneously and possibly present it in a compact symbolic fashion to the user.

To cover an area of interest, it is reasonable to use cameras with overlapping FOVs. Overlapping FOVs are typically used in computer vision for the purpose of extracting 3D information. The use of overlapping FOVs, however, creates an ambiguity in monitoring people. A single person present in the region of overlap will be seen in multiple camera views. There is need to identify the multiple projections of this person as the same 3D object, and to label them consistently across cameras for security or monitoring applications.

In related work, [1] presents an approach of dealing with the handoff problem based on 3D-environment model and calibrated cameras. The 3D coordinates of the person are established using the calibration information to find the location of the person in the environment model. At the time of handoff, only the 3D *voxel-occupancy* information is compared to achieve handoff, because multiple views of the same person will map to the same voxel in 3D. In [2], only relative calibration between cameras is used, and the correspondence is established using a set of feature points in a Bayesian probability framework. The intensity features used are taken from the centerline of the upper body in each projection to reduce the difference between perspectives. Geometric features such as the height of the person are also used. The system is able to predict when a person is about the exit the current view and picks the best next view for tracking. A different approach is described in [3] that does not require calibrated cameras. The camera calibration information is recovered by observing motion trajectories in the scene. The motion trajectories in different views are randomly matched against one another and plane homographies computed for each match. The correct homography is the one that is statistically most frequent, because even though there are more incorrect homographies than the correct one, they lie in scattered orientations. Once the correct homography is established, finer alignment is achieved through global frame alignment. Finally [4, 5] describe approaches which try to establish time correspondences between non-overlapping FOVs. The idea there is not to completely cover the area of interest, but to have motion constrained along a few paths, and to correspond objects based on time from one camera to another. Typical applications are cameras installed at intervals along a corridor [4] or on a freeway [5]. Recently,

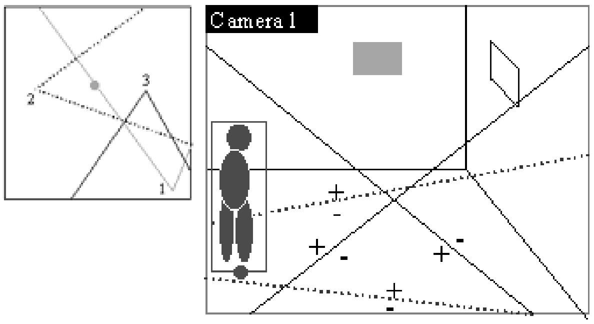

Figure 1: (Left) Three cameras setup in a room, with their FOVs shown by different lines. A person is entering the FOV of Camera 1. (Right) By looking at the FOV lines of Cameras 2 and 3 in Camera 1, we can determine that this person is visible in Camera 2 but not in Camera 3.

the work by [6] uses multiple modalities in a Bayesian network to solve the multiple camera tracking problem. The modalities used are grouped into geometry based modalities and recognition based modalities; the former including epipolar geometry, homography and landmark modalities, and the latter comprising of apparent height and color modalities.

The luxury of calibrated cameras or environment models is not available in most situations. We therefore tend to prefer approaches that can discover a sufficient amount of information about the environment to solve the handoff problem. We contend that camera calibration is unnecessary and an overkill for this problem, since the only place where handoff is required is when a person enters or leaves the FOV of any camera. By building a model of the relationship between FOV lines of various cameras can provide us sufficient information to solve the handoff problem. We extend our previous work [12] in two respects here. Firstly, we allow for the possibility of persons entering or exiting in the middle of the image, like a new person emerging from a car, and establish correspondence of such cases too, along with persons that enter from the limits of FOV of the camera. Secondly, we improve our initialization process, so that the lines can be determined using ordinary video, without the constraint of a single person visible in the environment.

To solve the multiple-camera tracking problem, we first need to perform tracking in each camera individually. Background subtraction is a popular approach in such applications, to separate the foreground from the background, in video sequences acquired by a fixed camera. Several successful background subtraction methods have been proposed in recent years, for example [7]. If only a single person is visible in the camera field of view (FOV), then background subtraction suffices as a tracker. However, if more than one object needs to be tracked, then the

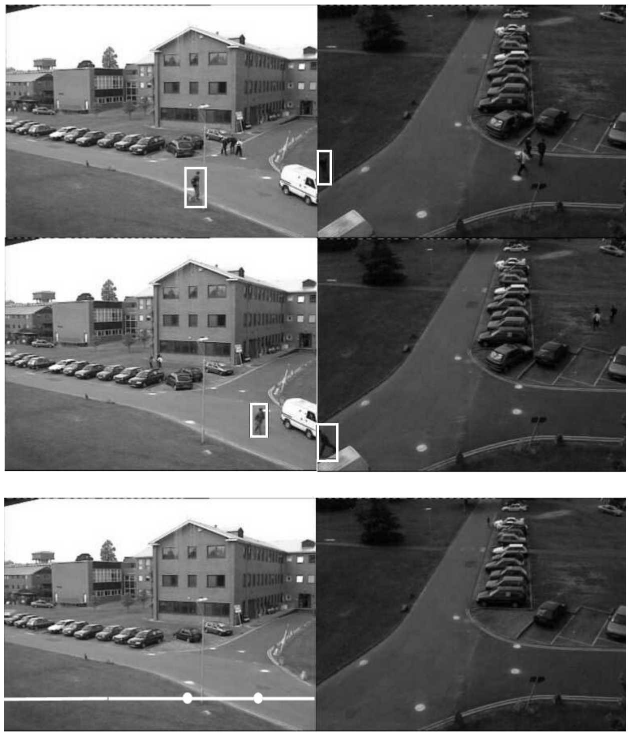

Figure 2: Generation of FOV lines. Two correct correspondences can be used to find a line. In the top pair of images, a person is entering or leaving the right camera. The position of this person in the left camera can be used to find the Left FOV line of left camera as seen in the right camera.

additional problem of corresponding between objects in successive frames needs to be addressed. There has been considerable literature on point correspondence problem, motivated by the moving light displays [8], for example [9,10]. However, due to noisy background subtraction, change in the size of regions, occlusion and entry/exit of objects, traditional point correspondence methods cannot be directly applied to the human tracking problem. We formulate this as a region correspondence problem, given background subtraction results from [7]. We describe the problems encountered in establishing correct correspondence, and present a solution based on linear velocity prediction and size and distance constraints.

In the next section we formalize the handoff problem and describe how the relationship between the FOV of different cameras can be used to solve the handoff problem. In Section 3, we describe how this relationship can be automatically discovered by observing motion of people in the environment. In Section 4, we discuss our approach of tracking in a single camera, which forms the input to the multiple camera system. Finally we present results of our experiments on the PETS 2001 dataset in Section 5.

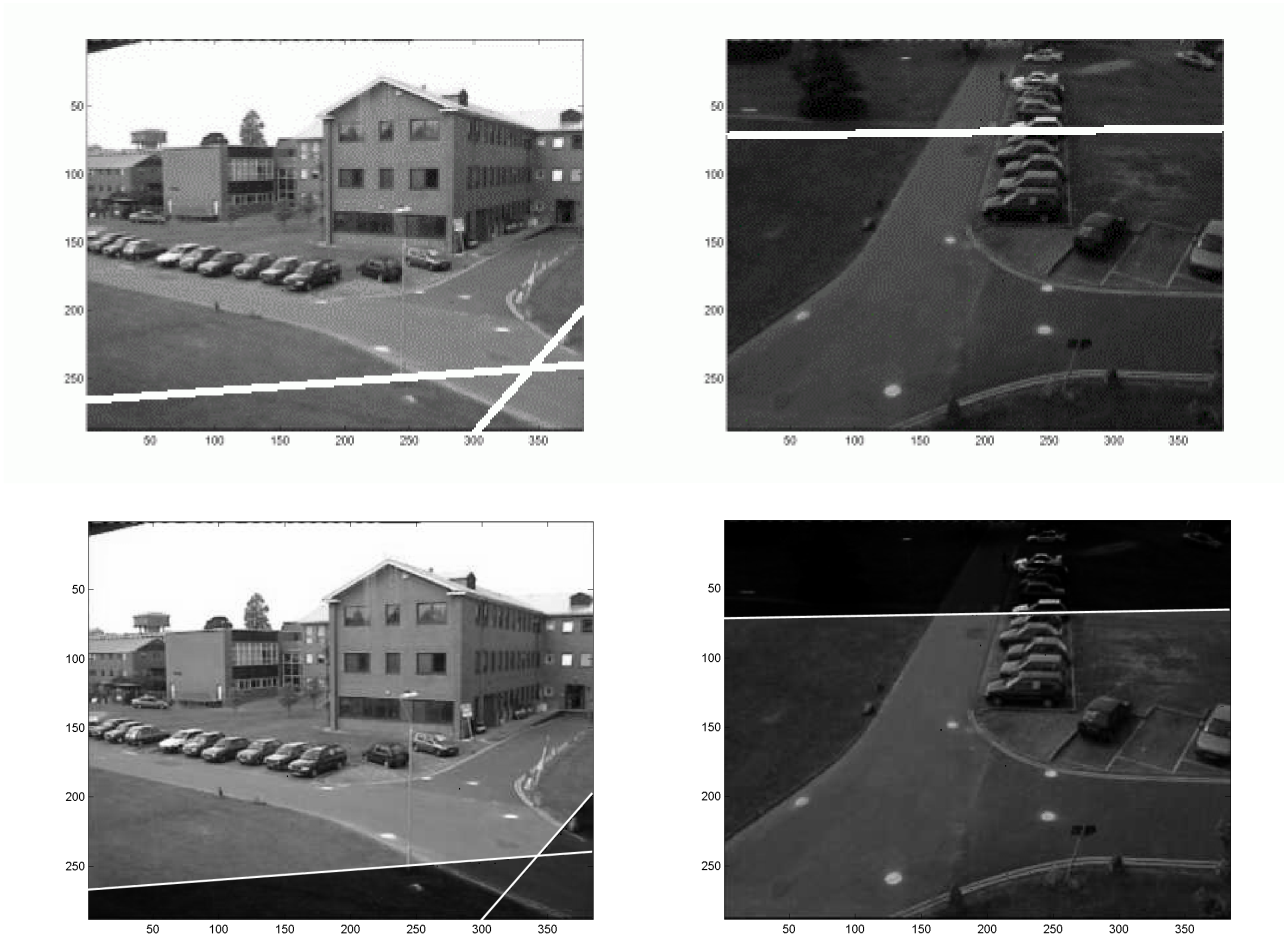

Figure 3: FOV lines determined from tracking data: The top row shows the lines determined by the system, and the bottom row shows the areas that are marked at not visible in the other camera.

2. EDGE OF FIELD OF VIEW LINES

The handoff problem occurs when a person enters the FOV of a camera. At that instant we want to determine if this person is visible in the FOV of any other camera, and if so, assign the same label to the new view. If the person is not visible in any other camera, then we want to assign a new label to this person..

Each camera's field of view can be described by four lines on the floor-plane, which are the left, right, top and bottom limits of FOV. Let L^i_l, L^i_r, L^i_t and L^i_b be the four limits of FOV of the i^{th} camera (C^i) on the ground plane (Figure 1). Let the projection of L^i_x ($x \in \{l, r, t, b\}$) in Camera j be denoted by L^{ij}_x. Note that L^{ii}_x denotes the sides of the image in C^i. As far as the camera pair i, j is concerned, the only locations of interest in the two images for handoff are L^{ij}_x and L^{ji}_x. These are up to eight lines, possibly four in each camera. Let us currently assume that a person already visible in one of the cameras is entering the FOV of another camera. In this case, all that needs to be done is to look at the associated line in the *other* camera and see which person is crossing that line. Consider the following scenario. A person is entering the FOV of C^2. There are two persons visible in C^1 at this instant. Both these persons are being tracked and we have a bounding box around them. By looking at the bottom part of the bounding boxes in C^1, we can determine quite easily which person has entered the FOV of C^2. The line that will help us determine this is L^{21}_l i.e. the left FOV of C^2 as seen in C^1. The new person in C^2 is therefore assigned the same label as that of the person who is closest to this line in C^1.

Detection of New Persons

In the example given above, it is assumed that when a person enters the FOV of a camera, he must be visible in the FOV of another camera. This is not always the case. A person might be entering from the door (in which case he might just "appear" in the middle of the image) or he might be entering the FOV from a point that is not visible in any other camera.

To establish correspondence between views, we look at the FOV lines of the current camera as seen in other cameras.

To find whether a person is visible in other cameras or not, we look at the FOV lines of other cameras as seen in the current camera. Consider the scenario when a person is entering the FOV of C^i. Whether this person is visible in any other camera (C^j, $j \neq i$) or not can be determined by looking at all the FOV lines that are of the form L^{ji}_x , i.e. edge of FOV lines of other cameras as visible in this camera (C^i). These lines partition the image C^i into (possibly over lapping) regions, marking the areas of image C^i that correspond to FOV of other cameras. Figure 2 illustrates this situation symbolically. Thus all the cameras in which current person is visible can be determined by acquiring the region of the person's feet.

Thus with each line L^{ji}_x, an additional variable δ^{ji}_x is stored. The value of δ^{ji}_x can either be +1 or –1, depending upon which side of the line falls inside the FOV of C^j. Then, given an arbitrary point (x', y') in C^i, the point's visibility in C^j can be determined by just determining if this point is on the correct side of both L^{ji}_l and L^{ji}_r. If L^{ji}_l is represented by A x' + B y' + C. The point (x', y') is visible in C^j if and only if

$$\text{sgn}(L^{ji}_x(x', y')) = \delta^{ji}_x \quad \forall\, x \in \{l, r, t, b\} \qquad (1)$$

In the case when all four lines of C^j are not visible in C^i, the condition in Eq. 1 is simplified to not include those lines.

Establishing Correspondence Between Views

When a person enters the FOV of a new camera, it can be determined whether this person is visible in the FOV of some other camera or not. Whenever a person is in the image all the other cameras in which this person will also be visible can be found out by using Eq 1. If there is no such camera, then a new label is assigned to this person. Otherwise the previous track of this person is found so that a link can be established between the two views. This is done by finding the person closest to the appropriate edge of FOV line. Say that the person entered from the left side of C^1. Then, the persons visible in all cameras other than C^1 will be searched and the person that is closest to the left edge of FOV line of C^1 in that camera will be found. These two views will then be linked together by entering them in an equivalence table. In general, if a person enters C^i from side x, then the label assigned to the new view will be:

$$\begin{aligned} &\text{label} = \arg\min_k (D(P^k, L^{ij}_x)) \quad \forall j \neq i \\ &\text{where } k = \text{set of persons visible in } C^j \end{aligned} \qquad (2)$$

where P^k is the label assigned to a person and $D(P, L)$ returns the absolute distance between the center of the bottom line of the rectangular bounding box of person P and the line L.

We have extended this formulation to also include the scenario when a person appears in the scene from a location that is not one of the edges of an image. In this case, this person will not be visible on one of the FOV lines. An example of such a scenario is when a person emerges from a car parked in the middle of the scene. In such a case, we look at all the tracks in the other camera, and see if any one of them is currently unassigned to one of the tracks in the current camera. If this track is also in the visible region, then it is an indication that it should have been visible in the current camera, and therefore a correspondence can be established. In case of multiple such tracks existing simultaneously, the decision is taken based on the consistency of motion relative to the edge of FOV lines. If a person is moving towards the left edge of one camera, and in the other camera, he is moving away from the left edge of FOV line, then this correspondence is obviously incorrect and will be ignored.

3. AUTOMATIC DETERMINATION OF FOV LINES

When tracking is initiated, there is no information provided about the FOV lines of the cameras. The system can, however, find this information by observing motion in the environment. Whenever there is a person entering or exiting one camera, he actually lies on the projection of the FOV line of this camera in all other ones in which he is visible. Suppose that there is only one person in the room. Then, when this person enters the FOV of a new camera, we find one constraint on the associated line. Two such constraints will define the line, and all constraints after that can be used in a least squares formulation. This concept is visually described in Figure 3.

In our previous paper [12], we demonstrated initialization of FOV lines by one person walking in the environment for about 40 seconds, and this was enough to initialize the lines. These lines were then used to resolve the correspondence problem between cameras. However it is not always possible to have only one person walking in the scene. Therefore, for cluttered situations where it is hard to find the correspondences to be used for initial setup, we propose another method. When multiple people are in the scene and if someone crosses the edge of FOV, all persons in other cameras are picked as being candidates for the projection of FOV line. Since the false candidates are randomly spread on both sides of the line whereas the correct candidates are clustered on a single line, correct correspondences will yield a line in a single orientation, but the wrong correspondences will yield lines in scattered orientations. We can then use Hough transform to find the best line in this case.

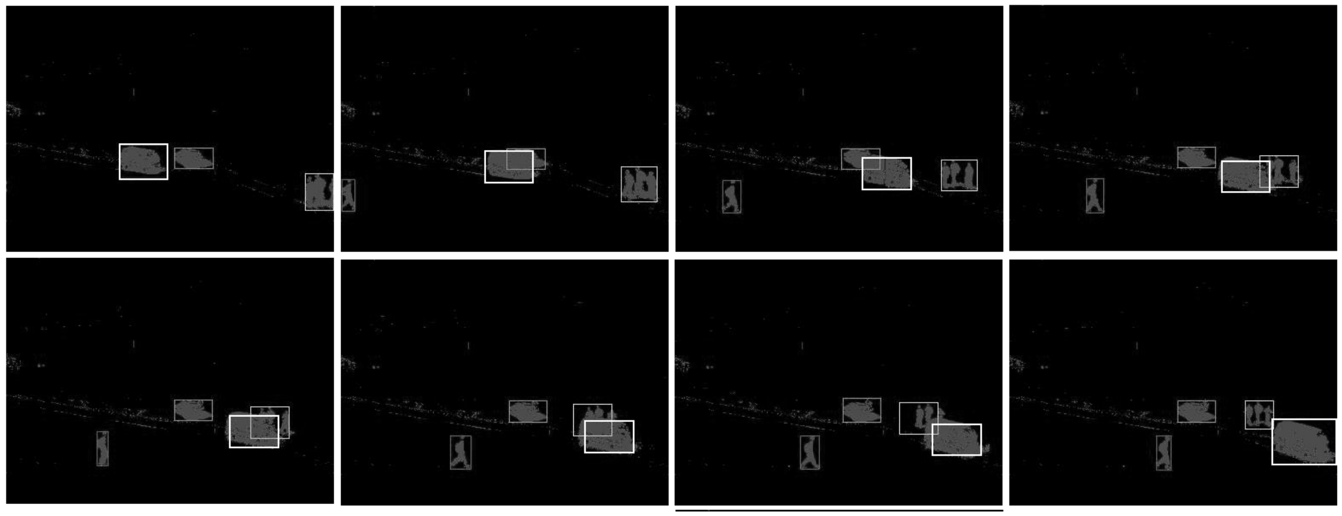

Figure 4: Complicated Occlusion Example: Results of single camera tracking algorithm, during occlusion.

This idea works when the FOV line is visible in the other cameras. However, it is easy to visualize a situation where one of the edges of the current camera is not visible in some other camera. If this is the case, then all the correspondences marked will be wrong ones, because the correct ones will not even be visible. This will result is a wrong estimate of the line via the Hough Transform.

We solve this problem by looking at long segments of the video and searching for unambiguous cases. If only one person is visible in a camera pair, then we can safely mark the bounding box of this person as an area that is *invisible in the other camera*. If we have information about classification of objects available (for example, person, car), then the idea can be further extended by looking for only that particular type of object in the other camera, and marking the area as invisible if no object of such a type exists. After a while, we obtain a *visibility map* for each camera pair. Any object entering or exiting from the invisible areas should not be added to the possible correspondences, thus reducing the number of false matches significantly. Any other type of categorization information may also be used, like for example, matching moving objects to only objects that are moving in the other camera. After doing this type of analysis, we consider only lines that have a significant amount of support from the data, i.e. they are determined by analyzing more than a certain number of correspondences.

Thus, theoretically, there are two options for initial setup of FOV lines. Quick self-calibration can be achieved by having only one person walk around the environment a few times. This should be sufficient for determining the relationship between the cameras. All lines of interest should be crossed at least twice during such a walk, which is often easily established during a 30-40 second random walk in a small room. The prior knowledge of having only one person in the room tells us that every correspondence between the cameras is a correct correspondence. However, if the environment is busy and cannot be cleared of people, we can use the second method, which finds the statistical best line, treating every valid correspondence as a potentially correct one. This method needs more points for a reliable estimate of the lines and will therefore take longer to be setup correctly. Additional constraints derived from categorization of objects and their motion may be used to reduce the number of false correspondences, thus reducing the time it requires to establish the lines. However, this method is completely automatic and does not need even the simple setup step required in the first method. Since we did not have control over the environment for the PETS 2001 dataset, we used this second method to find the FOV lines using the 'training' dataset, and then used these lines to perform multiple camera tracking on the 'testing' dataset.

4. TRACKING IN A SINGLE CAMERA

Tracking of objects in a single camera is not a trivial task. The test sequences for the workshop have a number of scenarios, which are very challenging for tracking algorithms. They include motion of groups of people, occlusion between people, occlusion between cars and people, occlusion due to scene structure and exits and entries during occlusion.

We present a tracking method which is able to accurately deal with occlusions between different objects and exit/entries during occlusion. The tracking method consists of extracting foreground regions in each frame and establishing correspondence of these regions between frames.

Background Subtraction: Moving objects are extracted from the sequence using the adaptive background subtraction method proposed by Stauffer and Grimson [7]. In the method, each pixel intensity is adaptively modeled by a mixture of *K* Gaussian distributions. The distributions are evaluated to determine which are more likely to result from a background process. The method reliably deals with long term changes in lighting conditions and scene changes. However, fast lighting changes and intensity variation due to compression do produce noise in the background subtracted image. Morphological filtering was performed to get rid of small noise.

The background subtraction method gives foreground pixels in each new frame. The foreground pixels are then segmented into regions using the connected components algorithm. We have used the term region to denote a foreground connected component in the rest of the paper.

Motion Correspondence:

The goal of tracking is to establish motion correspondence between regions that are moving in a 2D space that is essentially the projection of a 3D world. The regions can enter and exit the space. The regions can also get occluded by other regions.

For motion correspondence we have used a method which is an extension of the point correspondence paradigm [8, 9, 10], which tries to achieve correspondences that minimize the deviation in speed and direction of motion. Regions, as compared to points, have extra information like shape and size. This information can be used to further constrain the correspondences.

Each region is defined by the 2D coordinates of the centroid *X*, the bounding box *B* and the size *S*. The regions, for which correspondence has been established, have an associated velocity *V* and predicted change in size ∇S. In frame *'t'* of a sequence, there are *N* regions with centroids X_i^t (where $1 \le i \le N$) whose correspondences to previous frame are unknown. There are *M* regions with centroids X_L^{t-1} (where L is the label) in frame *t-1* whose correspondences have been established with the previous frames. The number of regions at '*t*' might be lesser than the number of regions in frame *t-1* due to exits or occlusion. Also it can be larger due to entries. The task is to establish correspondence between regions in frame *t* and frame *t-1* and to determine exits and entries in these frames.

The minimum cost criteria is used to establish correspondence. The cost function between two regions is defined as

$$C_{Li} = \rho \left\| (X_L^{t-1} + V_L^{t-1}) - X_i^t \right\| + (1-\rho) \left| (\nabla S_L^{t-1} + S_L^{t-1}) - S_i^t \right|$$

where $L \in$ {Labels of regions in frame t-1}

i is index of non-corresponded region in frame t and $1 \le i \le N$.

ρ is the weight parameter determining the percentage of cost due to change in size, and change in velocity.

The cost is calculated for all (*L*,*i*) pairs. Correspondence is established between the pair (L', i') that gives the lowest cost. The velocity and predicted size of region L' are updated as

$$V_{L'}^t = (1-\alpha) V_{L'}^{t-1} + \alpha (X_{L'}^t - X_{L'}^{t-1})$$

$$\nabla S_{L'}^t = (1-\alpha) \nabla S_{L'}^{t-1} + \alpha (S_{L'}^t - S_{L'}^{t-1})$$

where α is the learning rate.

Next all region pairs containing L' or i' are removed from consideration and the correspondence is established between the pair that gives the lowest cost among the rest of the pairs. The velocities and predicted sizes of corresponded regions are updated according to the above equations.

Once possible correspondences have been established using the minimum cost criteria, the following two cases might happen

2. Correspondences have been found between all regions in frames '*t-1*' and '*t*'.
3. Correspondences have not been found between all regions in frames '*t-1*' and '*t*'. There might be regions in frame '*t-1*' which have not been corresponded to in frame '*t*' due to occlusion or due to exits from FOV of a camera. Region can be occluded due to another region or due to scene structure. There might be regions in frame '*t*' which have not been corresponded to in frame '*t-1*' because they just entered the frame and no corresponding region in the previous frame exists. First we deal with the of frame '*t-1*'. Suppose a region X_L^{t-1} could not be corresponded to any region in frame '*t*'. A check for exit of X_L^{t-1} from the FOV of camera is done. If the position plus predicted velocity of a that regions is outside the frame boundary then it is determined to exit the frame. If this is not the case, then a check for occlusion is made. If X_L^{t-1} translated with its predicted velocity overlapped with another region in frame '*t-1*' then these two regions are declared as occluding each other. Note that this would have resulted in a single region in frame '*t*'. To compensate for the missing region we will add another region in frame '*t*' with centroid $X_L^t = X_L^{t-1} + V_L^{t-1}$ and $V_L^t = V_L^{t-1}$. If the occlusion check is not satisfied then the object is thought be occluded by a scene structure and to have exit. The non- corresponded regions in frame t are set to be entries. There initial velocity and change in size are zero.

5. EXPERIMENTS AND RESULTS

We used Dataset 1 from the PETS 2001 datasets for evaluating this system. Previously, we have demonstrated the working of these ideas for an indoor environment, with 3 cameras and up to 3 persons at a time in the room [11]. Here we present the results for an outdoor environment, with two cameras, and multiple persons and cars going through the environment (Dataset 1 Test Sequence, PETS 2001).

5.1 Single Camera Tracking

The tracking algorithm was run on dataset 1, Test sequences. The images were JPEG compressed and contained significant noise. We reduced the size of image by half and convolved it with a low pass filter to reduce noise. The algorithm was run on both sequences with same parameters.

The trajectories obtained by establishing correspondences were pruned to remove trajectories, which didn't move significantly throughout their existence. These trajectories were obtained due to uncovered background or due to motion of tree etc.

The algorithm performed well on the Camera 2 sequence. The occlusion between car and people were handled correctly. The exits under occlusion were also correctly detected. Entry of group of people was detected as single entry. However as soon as one person separated from the group he was tracked separately and its entry was detected at the point of separation.

Camera 1 sequence was more difficult for tracking. This was because the angle of elevation of the camera was lower resulting in long occlusions of multiple objects. A tree in the image was moving constantly. There was a pole in the middle of the view, which sometimes occluded objects, partially causing the foreground component to divide into multiple pieces.

Frame	Camera	Object	Camera	Object	Comment
98	1	1	2	1	Correct
470	2	2	1	2	Correct
668	1	3	2	3	Correct
774	2	4	1	4	Correct
963	2	5	1	5	Correct
1074					Incorrect
1185	1	6	2	7	Correct
1423	2	8	1	8	Correct
1578	2	9	1	7	Incorrect
2106	2	10	1	9	Correct
2177	2	12	1	10	Correct

Table 1: Results of multiple camera correspondence. For example, the first row states that object 1 in camera 1 is the same as object 1 in camera 2

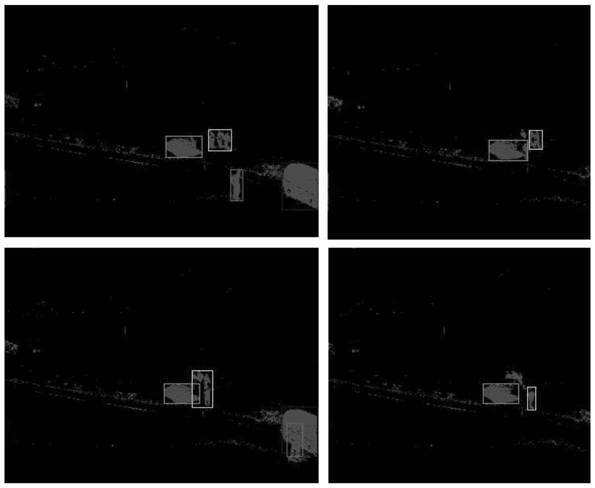
Figure 5: Example of Low level tracking failure. A person emerges from the car, while the group of people is being occluded by the car. The identity of the group is taken by the person, as if they might have turned back. This also generates an error in high level interpretation.

Decent results were obtained in Camera 2. One major problem arise when two cars occluded each other and pole divided the single component into two in frame number 847 and 849. Our tracker cannot deal with division of single connected component in two large components during occlusion. It assumes that occlusion is over and updates the predicted position with wrong predicted velocities and sizes. This region division rarely happens in cases when object are directly viewable. However in Camera 1, the pole caused the division in some frames (though in most frames the morphological operations joined the regions since the pole was thin). We manually connected the regions in 6 frames 847, 849, 2526, 2529. Our region correspondence was then able to correctly correspond the regions.

5.2 Multiple Camera Tracking

Multiple camera tracking works in two stages, the first one being establishing of FOV lines, and the second one being establishing correspondence and globally correct labels for all objects, using the FOV lines. To run multiple camera tracking on the Test Sequence, we used the Training Sequence to generate the FOV lines. We currently did not implement a classification scheme, to categorize objects into humans and vehicles, so we did this categorization manually for the Training Sequence. Some standard method, for example [12], may be utilized here. There are 31 'key-frames' in the Training sequence that consist of an entry or an exit event. We used the bounding boxes in these frames for the generation of the lines.

The way cameras are setup, only one FOV line of Camera 1(left) should be visible in Camera 2, and three FOV lines of Camera 2 (left, right and bottom) should be visible in Camera 1. However, out of the latter three lines, no interaction actually happens on the right line of Camera 2 in the Training Sequence, and only one exit event of a group of people in Testing Sequence. Since at least two correct correspondences are required to establish a line, our system does not find this line, but this does not result in any degradation of results. The lines generated are shown in Figure ().

Next we use the Test Sequence to establish correspondence between tracks of the same objects in the two cameras. The results for this are shown in Table 1. Our single camera tracking suffered from some errors, a few of which are reflected in the multiple camera results. We verified that if those errors were corrected manually, then the results of multiple camera correspondence are 100%. However, in the realistic tracking scenario, when the information from the lower level tracking is not correct, then a couple of mistakes are seen in Table 1. Each row of Table 1 corresponds two objects in different cameras. Nine correct correspondences were established. The first wrong correspondence occurs at frame 1074. Here, the single camera tracker failed during simultaneous occlusion of three objects. The occlusion starts between a group of persons and a car, but during occlusion, a new person also emerges from the car. This person assumes the label of the previous group, and we fail to register an entry event. The second failure occurs in frame 1578, where errors occur simultaneously in both the cameras. Both cameras are tracking groups of persons, and the group breaks in each camera. Thus a correspondence is established between these two new components, which is not correct. However, given the underlying tracking data, the error is to be expected.

CONCLUSION

We have described a framework to solve the camera handoff problem. We contend that camera calibration and 3D reconstruction is unnecessary for solving this problem. Instead, we present a system based on edge of FOV lines of cameras that can handle handoffs. We outline a process to automatically find the lines representing these limits, and then using them to resolve the ambiguity between multiple tracks. This approach does not require feature matching, which is difficult in widely separated cameras. We have also presented a correspondence based solution of tracking in a single camera. We show results on a two camera sequence from PETS dataset.

References

[1] P. H. Kelly, A. Katkere, D. Y. Kuramura, S. Moezzi, S. Chatterjee, R. Jain, "An architecture for multiple perspective interactive video", *Proc. ACM Conf. Multimedia*, pp. 201-212, 1995

[2] Q. Cai, J. K. Aggarwal, "Tracking Human Motion in Structured Environments Using a Distributed-Camera System", *IEEE PAMI*, Vol. 2, No. 11, pp. 1241-1247, Nov 1999

[3] L. Lee, R. Romano, G. Stein, "Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame", *IEEE Trans on PAMI*, Aug 2000, pp. 758-768

[4] Vera Kettnaker, Ramin Zabih, "Bayesian Multi-Camera Surveillance", *Proceedings of Computer Vision and Pattern Recognition*, Fort Collins, CO, June 23-25, 1999, pp. 253-259

[5] Hanna Pasula, Stuart Russell, Michael Ostland, Ya'acov Ritov, "Tracking Many Objects with Many Sensors" In *Proc. IJCAI-99*, Stockholm 1999

[6] Chang, T.-H.; Gong, S., "Tracking multiple people with a multi-camera system", Proceedings. 2001 IEEE Workshop on Multi-Object Tracking, with ICCV'01, Vancouver, BC, Canada, pp. 19-26, July 2001

[7] C. Stauffer, WEL Grimson, "Learning patterns of activity using real-time tracking", PAMI-August 2000.

[8] C. J. Veenman, M.J.T. Reinders, E. Baker, "Resolving motion correspondence for densely moving points", PAMI Jan 2001

[9] I. K Sethi, R. Jain, "Finding trajectories of feature points in monocular image sequences", PAMI, Jan 1987

[10] K. Rangarajan, M. Shah, "Establishing motion correspondence", CVGIP, July 1991

[11] S. Khan, O. Javed, M. Shah, "Human Tracking in Multiple Cameras", 8th International Conference on Computer Vision, Vancouver, Canada, July 2001.

[12] H. Fuiyoshi, A. J. Lipton, "Real-time human motion analysis by image skeletonization", Image Undertanding Workshop, 1998, Monterey, CA