Consistent Labeling of Tracked Objects in Multiple Cameras with Overlapping Fields of View

Sohaib Khan and Mubarak Shah

Abstract—In this paper, we address the issue of tracking moving objects in an environment covered by multiple uncalibrated cameras with overlapping fields of view, typical of most surveillance setups. In such a scenario, it is essential to establish correspondence between tracks of the same object, seen in different cameras, to recover complete information about the object. We call this the problem of consistent labeling of objects when seen in multiple cameras. We employ a novel approach of finding the limits of field of view (FOV) of each camera as visible in the other cameras. We show that, if the FOV lines are known, it is possible to disambiguate between multiple possibilities for correspondence. We present a method to automatically recover these lines by observing motion in the environment. Furthermore, once these lines are initialized, the homography between the views can also be recovered. We present results on indoor and outdoor sequences containing persons and vehicles.

Index Terms—Tracking, multiple cameras, multiperspective video, surveillance, camera handoff, sensor fusion.

1 Introduction

Tracking moving objects is a key problem in computer vision. It is important in a wide variety of applications, like surveillance, human motion analysis, traffic monitoring, and man machine interfaces. Tracking is essentially a correspondence problem; correspondence needs to be established between entities (objects, tokens) seen in the current image and those in the previous image. Single-camera-multiple-object tracking [13], [15] is a problem that has received considerable attention in computer vision literature. Constraints are added to disambiguate cases in which objects are in close proximity to each other. Such constraints exploit some property of the objects before and after the ambiguous event assuming that this property will be invariant; for example, constant velocity assumptions, motion uniformity constraints, object shape models, and color features.

Single camera tracking is limited in scope of its applications. While suited for certain applications like kiosks and local environments, even simple surveillance applications demand the use of multiple cameras. Multiple-camera-multiple-object tracking is used in indoor and outdoor surveillance. Typical scenarios in wide use include banks, convenience stores, airports, toll-plazas, and parking lots. Even the simplest of these setups requires multiple cameras, for two reasons: First, it is not possible for one camera to provide adequate coverage of the environment because of limited field of view (FOV). Second, it is desirable to have multiple cameras observing critical areas, to provide robustness against occlusion.

Multiple-camera multiple-object tracking [8], [5] has not received much attention in computer vision until very recently. Most current surveillance applications still treat multiple cameras as a set of single cameras; that is, there is no additional information gained from multiple cameras. However, multiple cameras provide us with a more complete history of a person's actions in an environment. To take advantage of additional cameras, it is necessary to establish correspondence between different views. Thus, we see a parallel between the traditional tracking problem in a single camera and that in multiple cameras: Tracking in a single camera is essentially a correspondence problem from frame to frame over time. Tracking in multiple cameras, on the other hand, is a correspondence problem between tracks of objects seen from different viewpoints at the same time instant. We call this the *consistent-labeling* in multiple cameras problem, i.e., *all views of the same object should be given the same label*.

1.1 Related Work

Multiple camera tracking is a relatively new problem in computer vision, but one that has gained increasing interest recently. The papers on this topic can be organized by what types of features are used, what is the matching strategy, and whether cameras are calibrated or not. The recent papers addressing this problem can be organized into three loose categories.

1.1.1 Feature Matching Approaches

The simplest scheme to establish consistent labeling may be to match color or other features of objects being tracked in each camera, to generate correspondence constraints. This matching may be performed statistically in a Kalman Filter framework [16] or using a Bayesian Network approach, as in [4]. In both cases, the authors do not restrict themselves to a single type of features but use a number of different features within the same framework. They also use camera calibration information, to learn more about the camera geometry and derive additional constraints. For example, in [4], the features used are grouped into geometry-based modalities and recognition-based modalities; the former including epipolar geometry, homography, and landmark modalities, the latter comprising of apparent height and color modalities. 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.

Color feature correspondence in multiple cameras is highly unreliable and, therefore, researchers have attempted, in these approaches, to make it more robust by statistical sampling and through augmentation by other features, such as apparent height. However, when the disparity is large, both in location and orientation, feature matches are not reliable. After all, a person may be wearing a shirt that has different colors on front and back. The reliability of feature matching decreases with increasing disparity and it is not uncommon, in fact, it is desirable, to have surveillance cameras looking at an area from opposing directions. Moreover, different cameras can have different intrinsic parameters as well as photometric properties like contrast, color-balance, etc. Lighting variations also contribute to the same object being seen with different colors in different cameras.

1.1.2 Approaches Based on 3D Information

If camera calibration and the 3D environment model are known, consistent labeling can be established by projecting the location of each 3D object in the world coordinate system, and establishing equivalence between objects that project to the same location. This is the approach taken in [9], where each camera is calibrated, and the world is a known ground plane. Therefore, the location of any object bounding-box in any camera can be found in 3D coordinates in the world. Equivalence between views is established by linking views that have similar projected 3D location. Ground plane homography recovered from camera calibration is also used in [1], [14].

While this approach may be of benefit in controlled environments, like football stadiums for which it was developed, it is difficult to have calibrated cameras and accurate environment maps

- S. Khan is with the Department of Computer Science, Lahore University of Management Sciences, Sector U, D.H.A., Lahore, 54792 Pakistan. E-mail: sohaib@lums.edu.pk.
- M. Shah is with the Computer Vision Laboratory, University of Central Florida, 4000 Central Florida Blvd., Computer Science Building (Bldg. No. 54), Room 201, Orlando, FL 32816. E-mail: shah@cs.ucf.edu.

Manuscript received 11 Jan. 2002; revised 27 Sept. 2002; accepted 14 Mar. 2003.
Recommended for acceptance by Z. Zhang.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 115687.

0162-8828/03/$17.00 © 2003 IEEE Published by the IEEE Computer Society

in the general surveillance scenario. Moreover, it is desirable to have a system that can be set up without expert intervention. We contend that camera calibration is not necessary, and is indeed an "overkill" for the consistent labeling problem. Most of the information needed can be extracted by observing motion over a period of time.

1.1.3 Alignment Approaches

Alignment-based approaches rely on recovering the geometric transformation between the cameras. Here, the correspondence between tracks is not explicitly resolved, but, if the transformation is recovered accurately, the tracks of the same object will overlap each other when aligned by the computed transformation.

Recently, the authors in [3] have considered the problem from a frame alignment point of view, extending the spatial image alignment methods to incorporate time information also. The result is complete alignment of camera sequences both, in time and space, so that the same object in different cameras will map to the same location. Of course, as is the case with spatial alignment, this can only be done when disparity between cameras is small. In our case, where the preferred camera arrangement is the one with large disparity, such a scheme is unlikely to work.

A different approach is described in [11] that uses trajectory information for alignment. The motion trajectories in different cameras are randomly matched against one another and plane homographies computed for each match. The correct homography is the one that is statistically most frequent. Finer alignment is achieved through global frame alignment. The method of trajectory alignment, though, is expensive, for a large number of possible alignments need to be computed. Nonetheless, this approach is closest in terms of its input/output relationship to our work.

On a different note, [10], [12], [17] describe approaches that try to establish time correspondences between *nonoverlapping* FOVs. The idea there is not to completely cover the area of interest, but to use the motion of the object, and the time taken by the object to move from one camera to another to to establish correspondence. Typical applications are cameras installed at intervals along a corridor [10] or on a freeway [12]. Javed et al. [17] use the observations of people or vehicles moving through nonoverlapping cameras and jointly models object velocities, intercamera travel times and locations, in a Parzen windows frame work to automatically learn the intercamera correspondence probabilities. A MAP approach is used to establish correspondence.

1.2 Our Approach

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 consistent labeling problem. The approach described in this paper does not need calibrated cameras. Since tracks of objects are available in each camera using low-level tracking (in single cameras), it is only necessary to establish just one correspondence between the tracks of the same object. The ideal place to establish this correspondence will be the instant when a new view is seen because, then, all subsequent points in that trajectory are automatically corresponded. Our system computes what we call the Field of View Lines (FOV lines). These are essentially the edges of the footprint of a camera *as seen in other cameras*. It is the computation of these lines that helps us establish correspondence between trajectories. It also allows us to compute, for each new view, the set of cameras in which that object will be visible.

To solve the multiple-camera tracking problem, the single-camera tracking problem needs to be solved first. For the purposes of this paper, we assume that reasonably correct single-camera tracking results are available through whatever method is preferred by the user. Indeed, our results are based on low-level tracking done by at least two different methods, to emphasize the independence of our approach to single-camera tracking. If there are significant errors in single-camera tracking, for example, due to occlusion, they will be reflected in the multiple-camera tracking. However, many of these errors can be corrected through integration of information from additional cameras. This is because the errors (say occlusion) might not happen simultaneously in all cameras viewing the object.

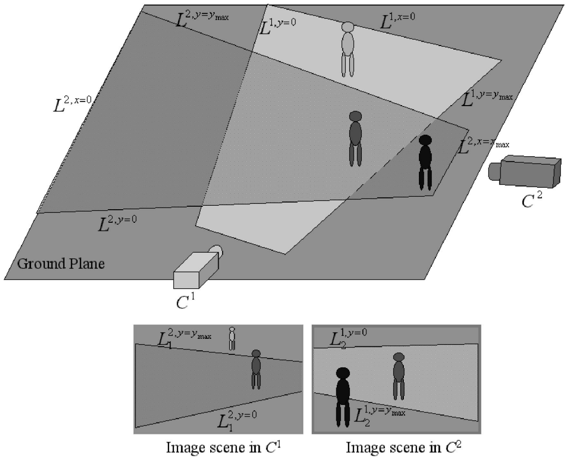


Fig. 1. FOV lines and their projections: Two cameras and their footprints are shown. The projection of boundaries of the footprint are also shown in the images that will be observed in the two cameras.

The rest of the paper is organized as follows: We first describe the notion of FOV lines in the next section and describe how they can be used to solve the consistent labeling problem. In Section 3, the main issue of automatically finding the FOV lines is tackled. We show that these lines can be initialized simply by observing motion of people in the environment. Finally, we present results of our experiments on various data sets in Section 4. Our test data sets consist of indoor and outdoor environments, containing up to three cameras and several people as well as vehicles.

2 Field of View Lines

We use the term *"view-event"* to denote an instant in time when an object enters or leaves the field of view of a camera. Ambiguity in labeling arises at the entry view-event, i.e., when an object enters the FOV of a camera. Thus, the boundaries of the FOVs of cameras are of special interest to us with regards to the consistent labeling problem. In this section, we will formalize this notion and show what information can be derived from knowledge about FOV.

We assume that the ground plane is visible in all cameras.[1] We also assume that all our sequences are already time-aligned and that cameras have overlapping FOVs. That is, it is allowed for two camera FOVs to not overlap with each other at all, as long as they are linked with cameras in between. This is not a restrictive assumption; if a person completely disappears from all the cameras and then reappears in some camera, she will be treated as a new object.

We denote the image seen in the $i\text{th}$ camera as $C^i(x, y)$. The field of view of C^i is a rectangular pyramid in space with its tip at the center of projection of the camer, and with its four sides passing through the lines $x = 0, x = x_{max}, y = 0, y = y_{max}$ on the image plane. For notational simplicity, we define S as the set of four lines defining the sides of a camera image. We will use lower case s to denote an arbitrary member of this set. The intersection of each planar side of this rectangular pyramid with the ground plane marks the boundaries of the footprint of the image. We call this a 3D FOV line.

A projection of this 3D FOV line marking the limit of the footprint may be visible in another camera because of overlapping FOVs (Fig. 1). If $L^{i,s}$ is an FOV line in 3D, i.e., it marks the viewing limit of C^i from side s, then $L_j^{i,s}$ is the *FOV line* (in 2D) of side s of C^i in C^j. In this section, we assume that these lines are known and show how they can be used to solve the consistent labeling problem.

[1] In reality, the entire ground plane may not be visible, but the portions of the limits of FOV of other cameras should be unoccluded.

2.1 Computing Visibility of Object in Other Cameras

We denote the kth object seen in C^i as O^i_k at any given time t. This is the local label of the view of the object, returned by the single-camera tracking module. O^i denotes the set of all objects visible in C^i (for example, $O^i = \{O^i_1, O^i_3\}$ means there are two objects in C^i at the current time instant, locally labeled 1 and 3). The location of a view is approximated by a single point, (x, y), given by bottom center of its bounding box, i.e., $(x^i_k, y^i_k) = \wp(O^i_k)$, where $\wp(.)$ returns the single point representing the location of the center of the bottom of the bounding box of the object. The feet of the person are chosen as a feature point because they represent the location such that rays from different cameras passing through this point will intersect at the same location on the ground plane (assuming relatively vertical people in images).

The overall consistent labeling task is to establish equivalences of the form $O^i_m \leftrightarrow O^j_n$. If the FOV lines are known, each line, $L^{i,s}_j$, divides the image into two parts, the one which is inside the projected FOV and the other which is outside. The function $L^{i,s}_j(x, y)$ returns greater than 0 if the point (x, y) lies on the side of $L^{i,s}_j$ that should visible in C^i, less than 0 if it should not be visible in C^i, and equal to zero if it is on the line.

Given this definition, we can determine whether the current object, O^i_k, in C^i will be visible in another camera or not. Note that all four FOV lines of each camera may not be visible in this camera; in such cases, we will be constrained to only the set of lines of C^j that are visible in C^i. Also, for shallow mounted cameras, it may be reasonable to consider only two FOV lines. The set of cameras $\mathbf{C}$ in which the current new view will be visible is given by

$$\mathbf{C}_i(k) = \{j | L^{j,s}_i(\wp(O^i_k)) > 0 \quad \forall \{x | L^{j,x}_i \exists C^i\}\}. \tag{1}$$

In case $\mathbf{C} = \phi$ (empty set), the view at (x', y') in C^i is that of a new object that is not currently seen in any other camera. In cases where $\mathbf{C}$ is not empty, at least one of the existing views must correspond to the current view. The camera in which the corresponding object will be found is a member of $\mathbf{C}$.

2.2 Establishing Consistent Labeling: Finding the Corresponding View

Once we know the set of cameras $\mathbf{C}$ in which the current object should be visible, we can search for the correct match among the objects seen in those cameras. This process is essentially of applying the FOV constraint to all the objects:

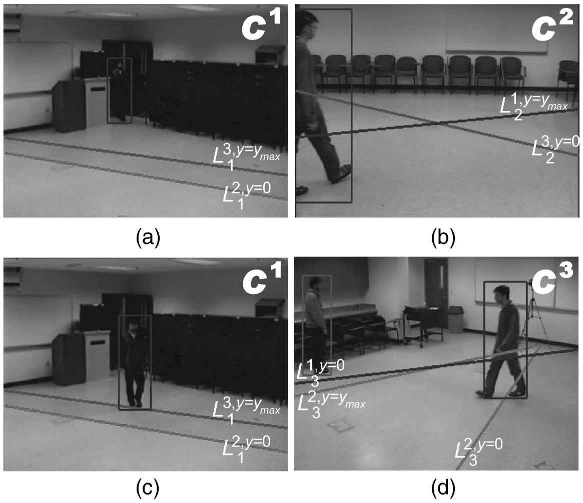
FOV Constraint. If a new view of an object is seen in C^i such that it has entered the image along side s, then the corresponding view of the same object will be visible on the line $L^{i,s}_j$ in C^j, provided $j \in \mathbf{C}$. Moreover, the direction of motion of this corresponding view will be such that the function $L^{i,s}_j(x', y')$ changes from negative to positive.

Based on this constraint, we can make a short list of the candidates for possible correspondence. In most situations, this constraint is enough to disambiguate between possible matches and to find the corresponding view of the same object. In that case, the new view in C^i is given the same label as the corresponding view in C^j. Practically, we implement this constraint as a minimum distance measure between the possible candidate views $C^j, j \in S$ and the line $L^{i,s}_j$.

$$O^i_m \leftrightarrow O^{j'}_n \;\; if \;\; \arg\min_{p,j} D(L^{i,s}_j, O^i_p) \quad \forall j \in \mathbf{C}_i(m), \tag{2}$$

where p is the label of objects in C^j and $D(L, O)$ returns the distance of an object O from a line L.

Fig. 2 clarifies the above discussion. In Fig. 2a, a person is entering the scene but he is visible in only one of the three cameras. In this case, the arrangement of FOV lines in the camera is such that we will obtain $|\mathbf{C}| = 1$, where $|\mathbf{C}|$ is the cardinality of set $\mathbf{C}$, according to (1). This indicates that the current view is that of an object not seen before and, therefore, will be given a unique label. Figs. 2b, 2c, and 2d show another view-event in which a person entering a camera (C^2) along $y = 0$ line. In this case, $\mathbf{C} = \{1, 3\}$. We therefore label this person to have the same label as the person at the minimum distance from $L^{2,y=0}_1$ and $L^{2,y=0}_3$ ((2)).



(a) (b)

(c) (d)

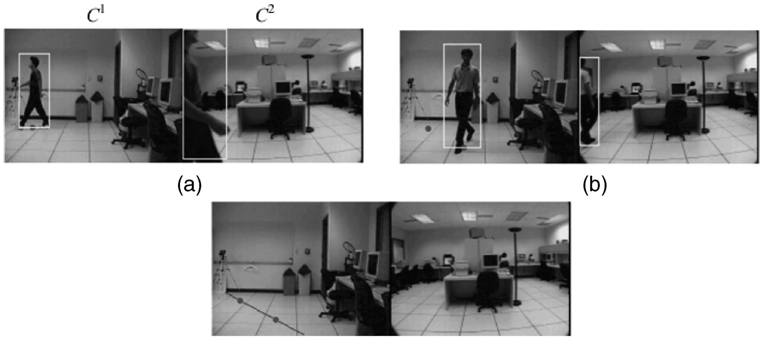
Fig. 2. Example of consistent labeling in 3-camera environment: Two different scenarios are shown here. The person in (a) has just entered the scene; it can be seen from the marked lines on the ground plane that he is outside the FOV of all other cameras. Therefore, he will be given a new label. The person in (b) is entering C^2, but from the marked FOV lines, it can be seen that he should also be visible in C^1 and C^3. The images of C^1 and C^3 at the same time instant are shown in (c) and (d), respectively. The person crossing the left FOV line of C^2 at this instant is the correct match and his label will be transferred to the new view in C^2. $y = 0$ denotes the left FOV line and $y = y_{max}$ denotes the right FOV line.

Until now, we have only considered view-events, which are not "actual" entry/exits from the environment, but represent only entry/exit in and out of the FOV of a camera. In contrast, there are "real" entry/exit events, which will occur when an object comes into the scene not from one of the sides of the FOV, but from somewhere in the middle of the FOV. This might be so due to several reasons. A person might appear from a door or from behind an object. Or an existing group of objects might split to generate separate trajectories (for example, a person emerging after parking a car). Such cases will not be used in the initialization phase, where FOV lines are being generated. However, during tracking, we need to not only identify this situation, but also establish correspondence of this new trajectory with existing trajectories in other cameras.

Once equivalences of the form $O^i_m \leftrightarrow O^j_n$ between views of the same object have been established using FOV lines, this one correspondence essentially establishes a correspondence between the entire tracks in the two views. To assimilate information from different views, we find the transformation between the ground plane in one camera to the other. This transformation can be easily computed after a sufficient number of equivalent tracks are known through the use of FOV lines. In our experiments, a simple affine transformation was used, but a higher order transformation, such as projective, can easily be substituted. For this work, only an approximate transformation is needed, and we found affine to be sufficient.

Once the homography is known, we can easily identify "real" entry/exit events from the middle of the scene using the computed homography. If a new track is observed which is initiated from the middle of the scene, we map it into the other cameras in which it should be visible and find the minimum distance from existing tracks. If the new track matches one of the existing ones, then it is given the same label, otherwise it is given a new label.

Thus, we may view this approach as an alternate way to compute the correspondences and homography between cameras. This method is simple compared to other approaches of computing homography, for example, [11]. In fact, it is interesting to note that the search problem in [11] looks for the statistically most common homography from random matches between tracks.



(a)

(b)

(c)

Fig. 3. (a) A person entering the FOV of C^2 from the left yields a point on the line $L_1^{2,y=0}$ in the image taken from C^1. (b) Another such correspondence yields the second point. The two points are joined to find the line $L_1^{2,y=0}$ shown in (c).

Here, (assuming time alignment is available as in most real-time systems), the search is done for statistically best lines and, thus, over a much reduced space (as explained in the next section). In case of high traffic during tracking, if more than one possible match is found along the FOV line, the additional test of consistency in location using homography will still disambiguate wrong correspondences.

3 Automatic Determination of FOV Lines

Determination of FOV lines is a trivial exercise if the camera intrinsic and extrinsic parameters are known, and the equation of the ground plane is available. The aim of this paper is to show that consistent labeling can be established even without knowing this information.

Determining each FOV line, $L_j^{i,s}$, requires knowing at least two correct correspondences between an object in C^i along the side s and the view of the same object in C^j (Fig. 3). Therefore, determination of FOV lines essentially requires solving this multiple-camera correspondence problem. Approximate correspondences can be found out by observing motion in the environment. Our strategy is to evaluate every possible correspondence and pick the ones that are known to be correct. Most consistent labeling scenarios are ambiguous, but simple nonambiguous situations also occur frequently. It is these unambiguous correspondences that we exploit; while determining the FOV lines. The system starts with no information about the lines in the beginning but after observing activity for a period of time, it is able to discover the locations of these lines. One way to initialize the system is to have only one person walk around the environment. In this case, the correspondences are trivially correct and, therefore, the lines can be found easily. However, in practical situations, we need a realistic scheme that can initialize the system even in the presence of several people.

When multiple people are in the scene and someone crosses the FOV line, all persons in other cameras are picked as being candidates for the projection of FOV line. False candidates are randomly spread on both sides of the line, whereas the correct candidates are clustered on a single line. Therefore, the correct correspondences will yield a line in a single orientation, but the wrong correspondences will yield lines in scattered orientations. We can then use the Hough transform to find the best line in this case (Fig. 4).

Some additional details need to be worked out. 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 incorrect because the correct ones will not even be visible. This may result in a wrong estimate of the line via the Hough Transform, even though, given enough observations, the confidence in the line found will be very low.

The problem we are faced with, therefore, is to reduce the number of false correspondences in our system to generate lines. Our approach to solve this problem is based on exploiting additional information from all cameras simultaneously.

3.1 Visibility Constraint

We define the binary *invisibility map* in C^i with respect to C^j (V_j^i) as the region of the image in C^i which is not visible in C^j. We can recover some information about the invisibility map by observing the motion of objects in the environment. Notice that the full invisibility map contains essentially the same information as the FOV lines (because the edges of the invisibility map are FOV lines). However, complete information about the invisibility map is not available during the initialization phase, although, based on each track in the environment, we can recover some information.

As an example, consider a camera pair C^i and C^j such that only one person is visible in C^i and nothing is visible in C^j. This means that the location of the person in C^i is such that in the 3D world, this location is not part of the footprint of C^j. Therefore, this location is included in the invisibility map of C^i with respect to C^j. In general, the following constraint needs to be satisfied for inclusion of a point in the invisibility map V_j^i:

Invisibility Map Generation: If $|O_t^i| = 1$ and $|O_t^j| = 0$, then $\wp(O_t^i) \in V_j^i$.

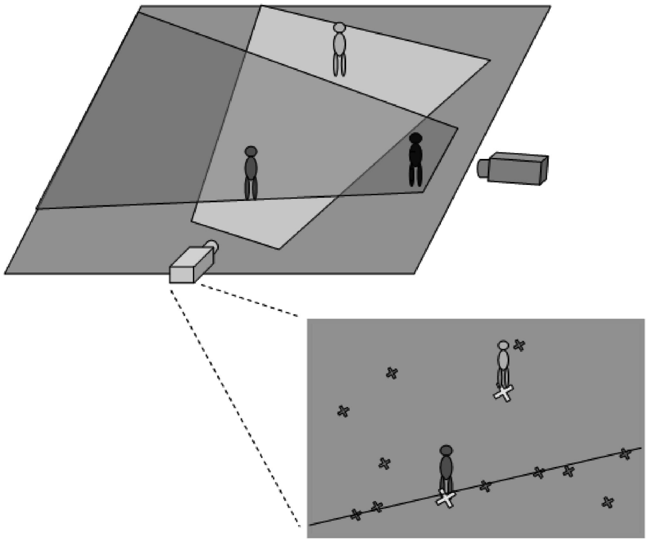


Fig. 4. Using Hough transform to find the FOV Line: Two correspondences are marked at the time instant shown (in white), only one of which is correct. Combined with previous correspondences (in gray), the best line is computed.

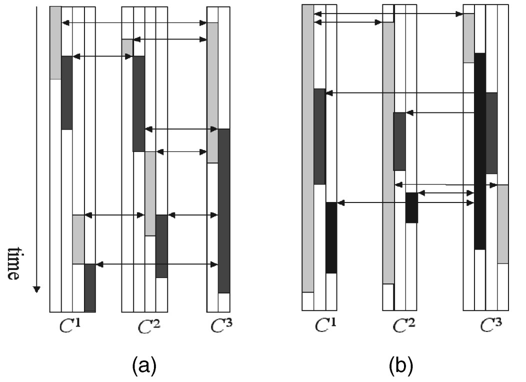


Fig. 5. Results of consistent labeling: Tracks seen in each camera are linked to each other through the consistent labeling approach. (a) and (b) show results on two different sequences. Tracks of the same color denote the same object. There are two objects with 10 tracks in (a) and three objects with 10 tracks in (b).

In practice, we do not just add a single point representing the object to the invisibility map, but we add several points around it in the map, too.

The visibility constraint progressively reduces the number of false correspondences that are encountered. The influence of this constraint increases as more and more cases of sparse traffic within the environment are observed.

Finally, we consider the issue of where this framework will break down. If traffic is high and occlusion is frequent, it should only affect the system in terms of longer time needed for recovering the FOV lines. However, if the underlying single-camera tracking system breaks down frequently, that may yield poor performance of the multiple-camera system. In addition, we have assumed the world to be planar and the feet to be visible. If there are significant deviations from this assumption, then the performance may break down. For example, if the system is being used indoors in a cluttered environment such that furniture is present at places where FOV line will pass, then the feet of the person may not be visible (as the person entering the scene may be partially occluded by the furniture). This framework will not work well in such a case because in reality, we will not have the limits of FOV as lines now, but rather as more complicated curves.

4 Results

We have performed a series of experiments on standard as well as home-grown sequences to test our approach. Our experiments consist of indoor and outdoor scenes, containing multiple persons and vehicles and covered by up to three cameras.

4.1 Indoor Environments with Three Cameras

In this set of experiments, we demonstrate that consistent labeling can be established if FOV lines are recovered. These experiments involved three cameras in a room, arranged to cover most of the floor area. To track persons, we used a simple background difference tracker. Each image was subtracted from a background image and the result thresholded, to generate a binary mask of the foreground objects. We performed noise cleaning heuristically, by dilating and eroding the mask, eliminating very small components and merging components likely to belong to the same person. To deal with occluding cases, we incorporated constant-velocity-based assumption in our tracker.

FOV lines were recovered by observing the motion of a person in the environment. The Hough transform method for determining the FOV lines works well with such sparse traffic as no false correspondences are observed. All significant edge of field of view lines were recovered from a short sequence of only about 40 seconds. To show that FOV lines can be used for multiple camera tracking, two persons entered the room, walked among the camera FOVs, and exited. The single-camera tracking module tracked each view of these persons separately and assigned a unique label to each track in every camera. Overall, 10 different trajectories of these persons were seen in the three cameras. Fig. 5a shows all the tracks. These are four tracks in C^1, four in C^2, and two in C^3. Our algorithm identified eight view-events, where a new view of an existing person was observed. In each of these situations, a person was seen entering a new camera. The distance of all other persons from the FOV line of that camera is used to find the previous view of the person. The arrows in Fig. 5 show the equivalence relations determined by our system. Once the equivalences are marked, the complete tracking history of the person is recovered by linking all the tracks of the same person together. The two different shades of gray in Fig. 5a show the globally consistent labels of the two persons. It can be seen that all view-events were handled correctly and the global tracking information was consistent at all times.

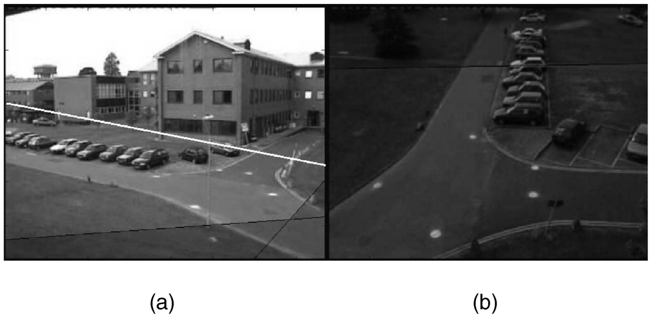


Fig. 6. FOV lines in PETS sequence: Two lines are found in C^1 (left) and one in C^2. The third line in C^1, marked in white is not recovered because of no view-events along that line. The shaded areas show the regions that are outside the FOV of the other camera.

We performed another experiment involving three persons in a different environment. Fig. 5b shows the recovered relationships between the 10 tracks seen in three cameras. In this case, our system correctly identified that these 10 tracks actually represented three different persons, with Person 1 entering in Camera 1, then moving to Cameras 2 and 3 before exiting the room while seen by Camera 1, etc.

4.2 Experiments on Outdoor Sequences with Two Cameras

To test the initialization process in a more complicated scenario, we used sequences from a standard data set in our experiments (Data set 1 from the PETS 2001 data sets). Here, we present the results for an outdoor environment, with two cameras and multiple persons and cars going through the environment. This data set consists of a Training Sequence and a Testing Sequence for each camera. The images were JPEG compressed and contained significant noise. We reduced the size of image by half along each dimension and convolved it with a low pass filter to reduce noise. Moving objects were detected using a background subtraction method [7] that was robust to local and global intensity changes. Detected objects were tracked in single cameras using the algorithm described in [18]. The algorithm was run on both sequences with the same parameters. The trajectories obtained by establishing correspondences were pruned to remove trajectories that did not move significantly throughout their existence. These trajectories were obtained due to uncovered background or due to motion of trees.

To run multiple camera tracking on the Test Sequence, we used the Training Sequence to generate the FOV lines. Because of the short length of the sequence, we removed additional false correspondences by employing object classification constraint, where a vehicle in one camera is not matched to a person in another. We did this categorization manually for the Training Sequence. Some standard method, for example [6], may be utilized here. There are 31 "key-frames" in the Training sequence indicating a view-event. We used the bounding boxes in these frames for the generation of the lines.

Due to the setup, only one FOV line of C^1 (left) should be visible in C^2, and three FOV lines of C^2 (left, right, and bottom) should be visible in C^1. However, out of the latter three lines, no interaction actually happens on the right line of C^2 in the Training Sequence

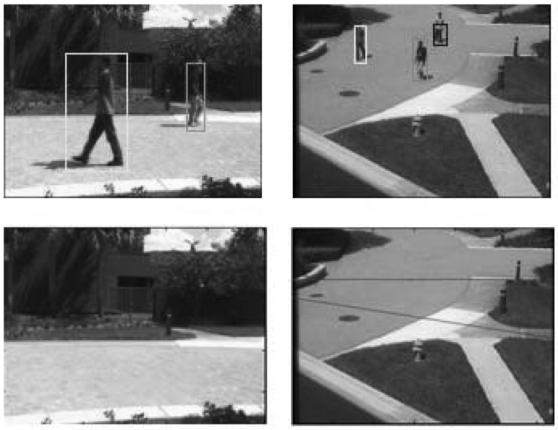


Fig. 7. Two camera data set with widely separated views which are different in obliquity and nearness to objects. Top row shows a typical image pair from the scene. Bottom row shows the lines recovered.

and there is 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; however, this does not result in any degradation of results. The lines generated are shown in Fig. 6. It should be noted that the lines are accurate where correspondences take place. For example, the line in C^2 is correct along the road where all the interactions took place during the experiments but deviates as we go away from this location. If more traffic is observed at a different location later, the estimate would automatically be improved.

Finally, to test different camera configurations, we tested the initialization phase of our algorithm with another two camera sequence (Fig. 7). In this case, there is a significant difference between the obliquity and the zoom factor of the cameras. One camera is looking at the environment from far above, while the other is near the path of movement of people. In this case, all necessary FOV lines were recovered from a sequence of 1,300 frames in which several people passed through the environment.

5 Conclusion

We have described a framework to solve the consistent labeling problem using uncalibrated cameras. We have presented a system based on FOV lines of cameras to establish equivalences between views of the same object as seen in different cameras. The process to automatically find the FOV lines was outlined. These lines are used to resolve the ambiguity between multiple tracks. This approach does not require feature matching, which is difficult in widely separated cameras. The whole approach is simple and fast. Results of experiments with both indoor and outdoor sequences were presented.

Acknowledgments

The authors would like to express their gratitude to Omar Javed and Zeeshan Rasheed for their help with carrying out the experiments reported in this paper.

References

[1] J. Black and T. Ellis, "Multiple Camera Image Tracking," *Proc. Performance Evaluation of Tracking and Surveillance Conf.(PETS 2001), with CVPR 2001,* Dec. 2001.

[2] Q. Cai and J.K. Aggarwal, "Tracking Human Motion in Structured Environments Using a Distributed Camera System," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 11, pp. 1241-1247, Nov. 1999.

[3] Y. Caspi and M. Irani, "A Step Towards Sequence-to-Sequence Alignment," *IEEE Proc. Conf. Computer Vision and Pattern Recognition,* June 2000.

[4] T.-H. Chang and S. Gong, "Tracking Multiple People with a Multi-Camera System," *Proc. IEEE Workshop Multi-Object Tracking, with ICCV '01,* July 2001.

[5] R.T. Collins, A.J. Fujiyoshi, and T. Kanade, "Algorithms for Cooperative Multisensor Surveillance," *Proc. IEEE,* vol. 89, no. 10, pp. 1456-1477, Oct. 2001.

[6] H. Fuiyoshi and A.J. Lipton, "Real-Time Human Motion Analysis by Image Skeletonization," *Proc. Image Understanding Workshop,* 1998.

[7] O. Javed, K. Shafique, and M. Shah, "A Hierarchical Approach to Robust Background Subtraction Using Color and Gradient Information," *Proc. Workshop Motion and Video Computing,* Dec. 2002.

[8] T. Kanade, R.T. Collins, A.J. Lipton, P.J. Burt, and L. Wixson, "Advances in Cooperative Multi-Sensor Video Surveillance," *Proc. DARPA Image Understanding Workshop,* pp. 3-24, 1998.

[9] P. Kelly, A. Katkere, D. Kuramura, S. Moezzi, S. Chatterjee, and R. Jain, "An Architecture for Multiple Perspective Interactive Video," *Proc. ACM Multimedia 95,* pp. 201-212, 1995.

[10] V. Kettnaker and R. Zabih, "Bayesian Multi-Camera Sureveillance," *Proc. Computer Vision and Pattern Recognition,* pp. 253-259, June 1999.

[11] L. Lee, R. Romano, and G. Stein, "Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 758-767, Aug. 2000.

[12] H. Pasula, S. Russell, M. Ostland, and Y. Ritov, "Tracking Many Objects with Many Sensors," *Proc. Int'l Joint Conf. Artificial Intelligence '99,* 1999.

[13] C. Stauffer and W.E.L. Grimson, "Learning Patterns of Acitivty Using Real-Time Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 747-757, Aug. 2000.

[14] T.N. Tan, G.D. Sullivan, and K.D. Baker, "Recognizing Objects on the Ground Plane," *Image Vision Computing,* vol. 12, no. 164, p. 172, Apr. 1994.

[15] H. Tao, H.S. Sawhney, and R. Kumar, "Object Tracking with Bayesian Estimation of Dynamic Layer Representations," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 1, pp. 75-89, Jan. 2002.

[16] A. Utsumi and J. Ohya, "Multiple-Camera-Based Human Tracking Using Non-Synchronous Observations," *Proc. Asian Conf. Computer Vision,* pp. 1034-1039, Jan. 2000.

[17] O. Javed, Z. Rasheed, K. Shafique, and M. Shah, "Tracking Across Multiple Cameras with Disjoint Views," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* 2003.

[18] S. Khan, O. Javed, and M. Shah, "Tracking in Uncalibrated Cameras with Overlapping Field of View," *Proc. Performance Evaluation of Tracking and Surveillance PETS 2001,* (with CVPR 2001), Dec. 2001.