Trajectory occlusion handling with multiple-view distance-minimization clustering

Mikhail Mozerov
Ariel Amato
Xavier Roca
Universitat Autònoma de Barcelona
Computer Vision Center and Departament d'Informàtica
08193 Cerdanyola, Spain
E-mail: mozerov@cvc.uab.es

Jordi Gonzàlez
Institut de Robòtica i Informàtica Industrial (UPC-CSIC)
Edifici U Parc Tecnològic de Barcelona
Barcelona 08028, Spain

Abstract. A robust and efficient method for overcoming the negative effects of long-time occlusion in the tracking process is presented. The proposed approach is based on the matching of multiple trajectories in time. Trajectories are sets of 2-D points in time and in a joint ground plane of the world coordinate system. In order to avoid mismatches due to possible measurement outliers, we introduce an integral distance between compared trajectories. The proposed method can also be considered as an interpolation algorithm for a disconnected trajectory during the blackout. Thus this technique solves one of the most difficult problems of occlusion handling: the matching of two unconnected parts of the same trajectory.

Subject terms: motion analysis; trajectory tracking; occlusion handling.

1 Introduction

1.1 *Motivation*

Analysis of human motion in a 3-D scene remains one of the most challenging open problems in computer vision,[1–3] and tracking algorithms are an essential part of the research in the field. There exist a wide variety of approaches that address this general problem.[4–6]

Many uncontrollable factors can severely complicate the tracking process. Temporal discontinuity of a trajectory is one of the most unpredictable events. For example, cognitive vision systems for human motion and behavior understanding have to identify each agent continuously to comprehend the developments observed within a scene. In other words, high-level descriptions assume assignment of a unique trajectory for each agent. It is known that fusing multiple views of a scene yields an analysis that is generally more accurate than any one view alone; therefore we propose using the multiple-view approach to solve the occlusion problem.

1.2 *Occlusion Classification*

There exist many approaches to classify occlusions. Some of them are intended to be general; others give consideration to specific features of the model. For example, one of the three possible classes of occlusion presented in Ref. 7 is the so-called self-occlusion: this occurs when there is an abrupt illumination change or a self perspective change of an object. Such approaches can have advantages for specific tasks. However, we adhere in this paper to the traditional notion of occlusion: blocking the visibility of objects by other objects. In such a context there exist two main classes of occlusion: static occlusions and dynamic occlusions. In this paper we do not consider the case of the static occlusions, because this problem can be solved by determining the region of interest (ROI) via the effective field of view of each camera. In turn, dynamic occlusions can be subdivided into two groups, which are short- and long-time occlusions. The first group is a well-studied subject in the tracking literature. Most of the algorithms that handle this category of occlusion are based on different sorts of prediction techniques. Such methods are very useful for mono-camera systems and can improve the robustness of occlusion handling in a multiple-view context.

1.3 *Related Work*

Occlusion handling is a quite complex problem that has recently been thoroughly investigated.[8–12] In all these works the occlusion problem is considered as a classification task: First the tracked blobs are labeled into classes according to several model cues, and then the occluded agent is matched with an appropriate class. Zhou and Aggarwal[13] consider multiple-view tracking as a problem of correspondence between objects seen from different views at the same time. This approach implies that all blobs of the same object should be matched. Our model requirement implies that the full 3-D relative positions of cameras have to be recovered; in other words, all cameras must be calibrated. Most traditional camera calibration techniques require specific knowledge regarding the geometric characteristics of the referenced object, such as direct linear transformation (DLT),[14] which solves for the perspective matrix linearly; similar methods include Tsai's[15] and Zhang's.[16] An elegant calibration method is used in the paper of Lee et al.,[17] where both intrinsic and extrinsic parameters of the cameras are recovered using only the tracks of moving objects.

The DLT-based methods require more than three points in the world coordinate system; for example, Tsai's calibration technique requires five points. In this work we utilize a calibration method with only three world coordinate points. The most interesting theoretical aspects of the three-point problem are considered by Haralick et al.[18] Ultimately, the

calibration process can be treated as two independent and intuitively clear parts with simple geometrical interpretation; the first part is to obtain the values of the marker triangle points in a camera coordinate system, and the second is to derive the rotation matrix **R** and the translation vector **T**.

1.4 *Proposed Contribution*

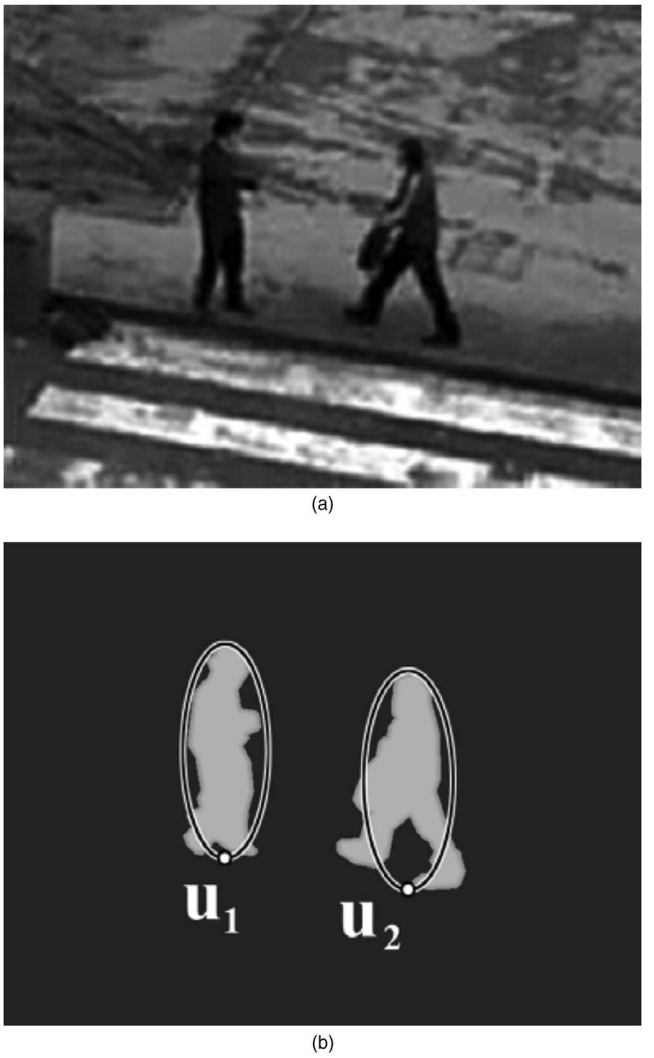
In our work we focus mostly on long-time occlusion handling. The motivation is clear, in that most related works do not even consider this problem: If long-time occlusion occurs, a new object template is constructed.[19] In the papers where the task of long-time occlusion handling is considered, this part of the algorithm assumes the application of classification and pattern recognition approaches.[20] If a multiple-camera system is available, it is natural to use a model that assumes that at each moment of the tracking process the object is visible at least at one of the cameras. This assumption allows one to avoid the object recognition part, which is computationally expensive. Another advantage of the presented method is the mathematical formalization of the tracking process and its results, which allows one to apply the proposed model to an arbitrary scene. The proposed algorithm aims to cluster together all visible and virtual (occluded) points that belong to the same agent in one *trajectory*. The method is based on the minimization of an objective function, which in turn depends on measured Euclidean distances between points in the ground plane.

The rest of this paper is organized as follows: Sec. 2 describes the method of greedy tracking that uses a background subtraction approach. Section 3 describes the calibration method. Section 4 explains the concept of trajectory clustering. Section 5 is devoted to the experimental results. Concluding remarks are made in Sec. 6.

2 Greedy Tracking Algorithm with Color Segmentation

The initial phase of our tracking process involves color segmentation. We choose for color background subtraction the method described in Ref. 21. The result of the segmentation is shown in Fig. 1(b). Once the segmentation is performed, the size and position of the ellipse that surrounds the object can be obtained. In this paper the lowest point **u** of the generated ellipse is referred to as a *floor point*. For example, in Fig. 1(b) two floor points $\mathbf{u}_1$ and $\mathbf{u}_2$ in the image coordinate system are shown. If a segment has no history (there is no matched neighbor ellipse in the previous frame), the initial parameter of a new ellipse is generated using features of the considered segment. All previously tracked ellipses are matched with a new segment map even if the same segment is matched with partly or fully overlapped ellipses. This technique allows us to avoid uncertainties that can arise if the segment represents more than one agent.

On the other hand, the generation of new ellipses in each frame is a computationally more complicated task than the matching of existing ellipses with the segment map. Suppose that T frames $t \in \{1, \ldots, T\}$ are considered, and there are $N(t)$ floor points $n \in \{1, \ldots, N(t)\}$ detected in each frame. In this case the greedy algorithm provides us with a set of unlabeled points $\mathbf{u}_n(t)$, and all these points have to be attributed. Note that the points $\mathbf{u}_n(t)$ are projections of real-world points; thus, to be able to measure the Euclidean distance between real-world points, all points $\mathbf{u}_n(t)$ have to be projected onto the joint ground plane.



(a)

(b)

Fig. 1 (a) Original image. (b) Segmentation with floor points $\mathbf{u}_1$ and $\mathbf{u}_2$.

3 Data Fusion

In order to convert the image planes obtained from multiple camera views into one ground-plane map of an inspected scene, one needs to perform a calibration.

3.1 *Camera Model*

In our geometric derivation the pinhole model is used (Fig. 2). The camera coordinate system is defined by $(\mathbf{p}_0, \mathbf{X}_c, \mathbf{Y}_c, \mathbf{Z}_c)$, where $\mathbf{p}_0$ is the origin, and $\mathbf{Z}_c$ is the axis that coincides with the optical axis. The subindices c, w, and n have the following meanings: c, a vector that belongs to the camera coordinate system, w, a vector that belongs to the world coordinate system; n, is a vector numeration subindex. Additionally, we have to note that in Sec. 3 the upper index t means the matrix transposition operation, and **T** is a translation vector. A scene point $\mathbf{p}_n$ can be represented in

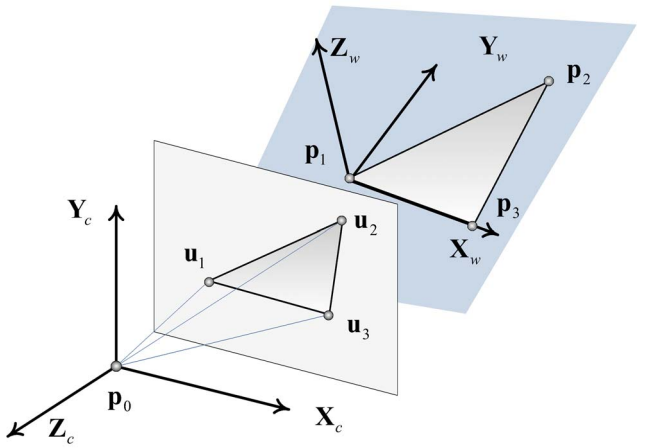


Fig. 2 Pinhole camera model.

the camera coordinate system by $\mathbf{p}_n=[x_n,y_n,z_n]^t$, and in the world coordinate system $(\mathbf{O}_w,\mathbf{X}_w,\mathbf{Y}_w,\mathbf{Z}_w)$ by $\mathbf{p}'_n=[x'_n,y'_n,z'_n]^t$.The projection of any world point onto the discrete grid of the image plane, which is parallel to the camera plane, is denoted by $\mathbf{u}_n=[\varphi i_n,\varphi j_n,-1]^t$, where φ is the intrinsic camera parameter, which is equal to the reciprocal of the focal length measured in pixels: $\varphi=1/f_{\text{pix}}$. The discrete grid indices i and j belong to the domain $I\times J: i\in\{-I/2,\ldots,0,\ldots,I/2\}$, $j\in\{-J/2,\ldots,0,\ldots,J/2\}$, where I and J are the vertical and horizontal sizes of a CCD matrix, respectively. The relationships between the two coordinate points $\mathbf{p}_n$ and $\mathbf{u}_n$ is as follows:

$$\mathbf{p}_n=Z_n\mathbf{u}_n. \tag{1}$$

3.2 *Camera Calibration*

We assume that the intrinsic constant φ is given or predetermined. Note that, in this case, only the depth parameter Z_n is needed to describe completely all the visible points in the world coordinate system of the camera; see Eq. (1). On the other hand, three noncollinear points of the world coordinate system are enough to determine the extrinsic coordinate system. So the next step of our calibration process can be subdivided, in turn, into two independent parts. First, we calculate (or measure, if possible) the depth values of marker triangle points in the camera coordinate system, and then we obtain the extrinsic-parameter rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$. To obtain the depth values $\{z_1,z_2,z_3\}$ of the marker triangle we use the method that is described in Ref. 17. After the unique solution of the marker triangle problem is obtained, it is possible to derive the extrinsic parameters $\mathbf{R}$ and $\mathbf{T}$.

The rotation matrix and the translation vector transform the new world coordinates of a point into the camera world coordinates given by

$$\mathbf{p}_n=\mathbf{R}\mathbf{p}'_n+\mathbf{T}. \tag{2}$$

To obtain the transformation parameters, first it is necessary to define the new world coordinate system. Let us the origin of our world coordinate system to be one of the marker triangle vertices $\mathbf{O}_w=\mathbf{p}_1$, and take one of the triangle legs as the $\mathbf{X}$ coordinate axis. It is reasonable to put our marker triangle into the new coordinate system plane with $z=0$. Then, the new world coordinate system can be defined by its basis

$$\mathbf{X}_w=\frac{\mathbf{p}_3-\mathbf{p}_1}{|\mathbf{p}_3-\mathbf{p}_1|},$$

$$\mathbf{Z}_w=\frac{\mathbf{X}_w\times(\mathbf{p}_2-\mathbf{p}_1)}{|\mathbf{X}_w\times(\mathbf{p}_2-\mathbf{p}_1)|},$$

$$\mathbf{Y}_w=\mathbf{Z}_w\times\mathbf{X}_w. \tag{3}$$

Now, the rotation matrix can be represented as a simple combination of the basis vectors and the translation vector is equal to the origin vector:

$$\mathbf{R}=\begin{bmatrix}\mathbf{X}_w & \mathbf{Y}_w & \mathbf{Z}_w\end{bmatrix},\quad \mathbf{T}=\mathbf{p}_1. \tag{4}$$

The use of a three-point calibration method is not compulsory: any known calibration method can be used (e.g., Ref. 15) to obtain the extrinsic parameters $\mathbf{R}$ and $\mathbf{T}$. In that case, the only requirement is to increase the number of calibration points.

3.3 *Inverse Perspective Mapping*

Once the tracking algorithm finds a pixel of interest $\{i,j\}$ in the image of one camera (in our case it is a floor point of an agent), this pixel should be projected onto the joint plane in the world coordinate system to match with another point in a different image obtained from a second camera. This projection process is referred to as the *inverse perspective mapping*. In other words, a function $\mathbf{p}'(i,j)$ that projects an arbitrary pixel $\{i,j\}$ into the real-world plane should be derived. In order to achieve this, we need to obtain the value of the optical center point in the joint world coordinate system:

$$\mathbf{p}'_0=\mathbf{R}^t(\mathbf{p}_0-\mathbf{T})=-\mathbf{R}^t\mathbf{T}. \tag{5}$$

Let us denote the pure rotation of the image plane vectors by

$$\breve{\mathbf{u}}(i,j)=\mathbf{R}^t\mathbf{u}(i,j)=\mathbf{R}^t[\varphi i,\varphi j,-1]^t. \tag{6}$$

Then the desired inverse perspective mapping function is

$$\mathbf{p}'(i,j)=\mathbf{p}'_0-\breve{\mathbf{u}}(i,j)\frac{\hat{z}_0}{\breve{z}}_{ij}. \tag{7}$$

Our model implies that the inspected scene is a plane. After the inverse perspective projection, the matched points in the image of camera 1 and in the image of camera 2 have to coincide in the joint ground plane. The result of such inverse projections is shown in Fig. 3(a) and 3(b). Curvature of the ground plane or an inaccurate calibration process can cause divergence between the matched points. To check the accuracy of our inverse perspective projection, we mapped the two images from camera 1 and camera 2 into one joint image. Then we measured the divergence errors between several known points. In the region of inter-

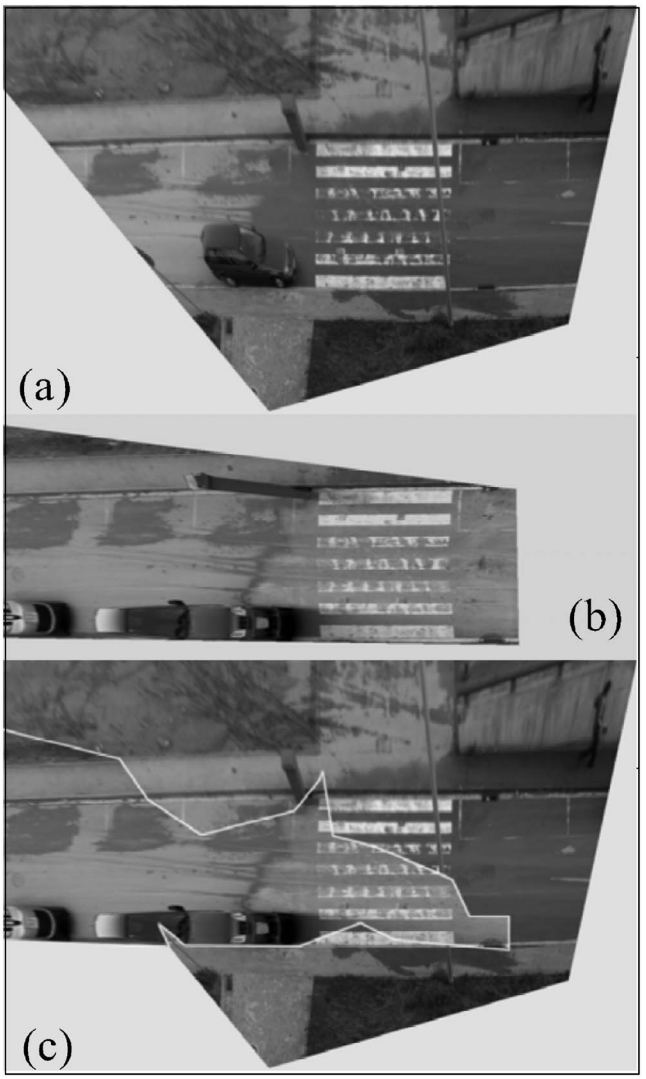


Fig. 3 (a) Inverse perspective projection of the image in camera 1. (b) Inverse perspective projection of the image in camera 2. (c) Composite image as a result of two camera views overlapping.

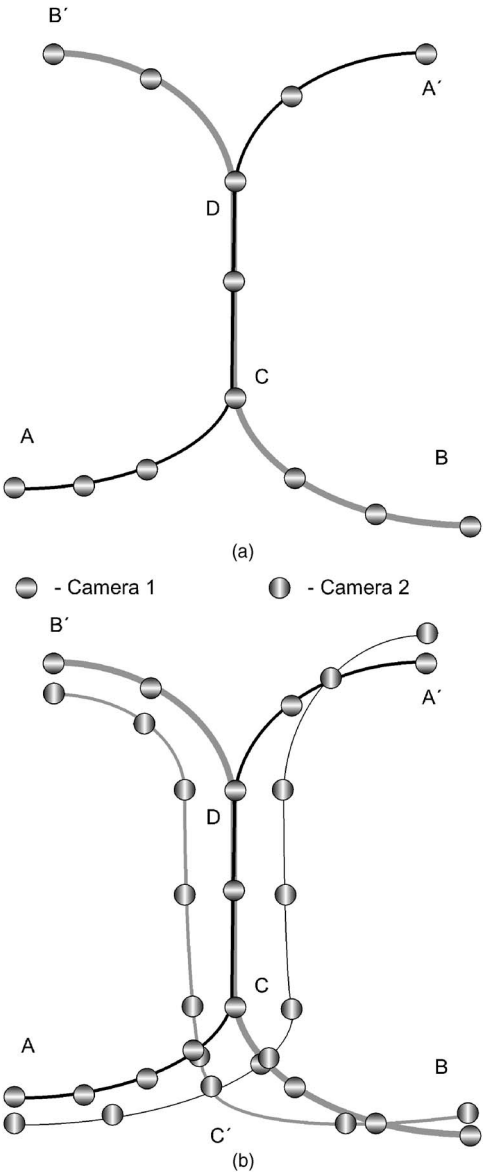


Fig. 4 (a) Trajectories' points of one camera in a joint ground plane. (b) Trajectories' points of two cameras overlapped in a joint ground plane.

est this error does not exceed three pixels (or 7 cm). A visual illustration of the accuracy of our inverse perspective projection is shown in Fig. 3(c). Here an arbitrarily drawn fringe divides the composite image in Fig. 3(c) into two parts: one belongs to the image of camera 1, and the other to the image of camera 2. We can see that the visual features (e.g., the zebra-crossing lines) coincide along the arbitrary fringe.

4 Trajectory Tracking

In this paper we present a concept of trajectory matching that allows us to overcome confusing effects of temporary occlusion. Let us formalize the negative effects of occlusion.

4.1 *Understanding the Occlusion Problem*

Let us consider an imaginary scenario with only two cameras and two agents, as in Fig. 4. Ideally, any method that handles occlusion should answer two basic questions:

- Which trajectory, $[A,A']$ or $[A,B']$, belongs to the same agent
- Which agent, A or B, is visible during the occlusion blackout $[C,D]$.

It is not enough to have the information about coordinates of trajectories' points taken from one camera [see Fig. 4(a)] to solve the problem, and additional cues have to be utilized. Indeed, when one agent occludes another (segment $[C,D]$), both trajectories' points (of agents A and B) become virtual or logically inseparable. Here we consider separability within the framework of the Euclidean distance criterion. However, the two-camera system provides us with much more stronger tracking cues [Fig. 4(b)]. Actually, a separability break can only occur in two relatively rare cases: (a) when both pairs of interacting trajectories'

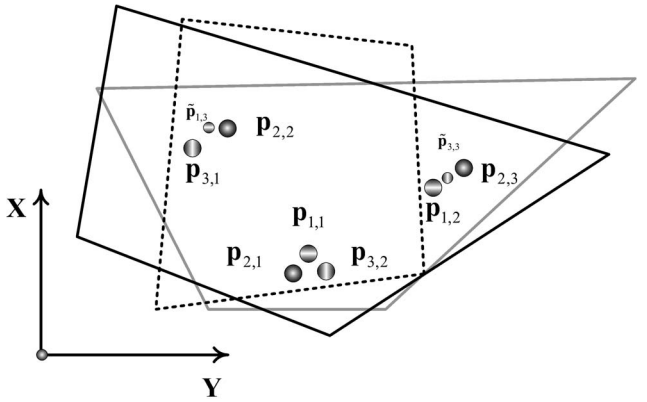


Fig. 5 Unlabeled trajectories' points in a joint ground plane.

points at the same snapshot time are occluded or virtual; (b) when interacting points of a pair taken from one camera view are virtual. Then in the next snapshot time, interacting points of another pair become virtual. In other cases we can recover the lost information (segment $[C',C]$). The multiple-view approach also answers the second question. Using the values of coordinates of the visible pair [camera 2 in Fig. 4(b)], it is possible to say which agent is closer to camera 1, and consequently, to know which agent (A or B) is visible in the view of camera 1. Below we formalize these intuitive explanations.

4.2 *Trajectory Clustering Algorithm*

The experimental setup gives us a set of $k \in \{1,\dots,K\}$ cameras C_k and T images $I_{t,k}$ obtained from each camera at a specific time $t \in \{1,\dots,T\}$. Using the greedy tracking algorithm described in Sec. 2, we obtain a set of floor points:

$$\mathbf{Q}_0 = \bigcup_{t,k,n \in \Omega_0} \mathbf{p}^t_{k,n}, \tag{8}$$

where $\mathbf{p}$ is a point of the world coordinate system in the joint ground plane; the subindex domain Ω_0 is $k \in \{1,\dots,K\}$, $t \in \{1,\dots,T\}$, $n \in \{1,\dots,N_0(k,t)\}$; and $N_0(k,t)$ is the number of detected floor points at a specific snapshot time in a certain camera view.

The number of detected floor points in a single image at a certain snapshot time t depends on the camera index k, due to occlusion and nonoverlapping fields of view of different cameras. So, for our algorithm it is convenient to complete the set of real points $\mathbf{Q}_0$ with a set of invisible or virtual points

$$\widetilde{\mathbf{Q}} = \bigcup_{t,k,n \in \widetilde{\Omega}} \tilde{\mathbf{p}}^t_{k,n}; \tag{9}$$

where the domain $\widetilde{\Omega}$ can be obtained by replacing $N_0(k,t)$ with the number of additional virtual points $\widetilde{N}(k,t)$ in the domain Ω_0. The greedy algorithm described in Sec. 2 generates almost all invisible trajectory points $\tilde{\mathbf{p}}$ (for example, point $\tilde{\mathbf{p}}_{1,3}$ in Fig. 5), except the ones that are outside the field of view of the specific camera. In this case we apply a preprocessing algorithm that generates a virtual point near some visible points. For example, in Fig. 5 two points $\mathbf{p}_{1,2}$ and $\mathbf{p}_{2,3}$ that lie, respectively in the fields of view of two cameras generate the virtual point $\tilde{\mathbf{p}}_{3,3}$, whose coordinate value is equal to the mean value of the coordinates of the generated pair. So now, at a certain snapshot t of any camera, we have a number of points that does not depend on the index k:

$$N(t) = N_0(k,t) + \widetilde{N}(k,t). \tag{10}$$

Consequently, a set $\mathbf{Q}^t$ at time t consists of floor points $\mathbf{p}^t$ with a total number of points that is equal to $K \times N(t)$:

$$\mathbf{Q}^t = \bigcup_{k,n \in K,N(t)} \mathbf{p}^t_{k,n}. \tag{11}$$

Let us define a selection operation with an arbitrary selection matrix $\mathbf{a}^t_s$, where the subindex $s \in \{1,\dots,S(t)\}$ denotes that the value of the matrix $\mathbf{a}^t_s$ is one of $S(t)$ possible values. The size of such a matrix at time t is $K \times N(t)$, and this matrix extracts a set $\mathbf{P}^t_s$ of K points from $K \times N(t)$ points of the set $\mathbf{Q}^t$:

$$\mathbf{P}^t_s = \mathbf{a}^t_s \circ \mathbf{Q}^t, \tag{12}$$

where $\circ$ denotes a selection operation:

$$\mathbf{a} \circ \mathbf{Q} = \bigcup_{k \in K, n \in N} (a_{k,n} \cap \mathbf{p}_{k,n}). \tag{13}$$

The value of an element $a_{k,n}$ of the selection matrix $\mathbf{a}$ can be 0 or 1 (TRUE or FALSE), but for each fixed string the matrix must have only one unity. This constraint has a simple meaning: An agent trajectory can be exactly represented by one floor point in a certain camera view at any fixed time. Thus the number $S(t)$ of all the possible matrices $\mathbf{a}^t_s$ is equal to $N(t)^K$. One of the consequences of the defined selection operation is that any set $\mathbf{P}^t_s$ consists exactly of K points with different k indices. Furthermore, if the Euclidean distance between points of a set $\mathbf{P}^t_s$ is minimal among other possible sets, it is reasonable to suggest that the set $\mathbf{P}^t_s$ is a package of trajectory points, which belong to the same agent seen from different cameras. Therefore, it is natural to form a trajectory $\mathbf{O}_s$ as a consecutive group of such sets in time:

$$\mathbf{O}_s = \bigcup_{t \in T} \mathbf{P}^t_s. \tag{14}$$

In order to develop a rule for trajectory formation, we introduce a composite distance that consists of the interframe distance Δ^t_s between two consecutive sets in the same trajectory,

$$\Delta^t_s = \|\mathbf{P}^t_s - \mathbf{P}^{t-1}_s\| = \sum_{k,l \in K} \sum_{n \in N(t), m \in N(t-1)} a^t_{k,n,s} a^{t-1}_{l,m,s} \|\mathbf{p}^t_{k,n} - \mathbf{p}^{t-1}_{l,m}\|, \tag{15}$$

and the intrinsic distance D^t_s of the set $\mathbf{P}^t_s$ itself,

$$D^t_s = \sum_{k,l \in K} \sum_{n,m \in N(t)} a^t_{k,n,s} a^t_{l,m,s} \|\mathbf{p}^t_{k,n} - \mathbf{p}^t_{l,m}\|. \tag{16}$$

It is reasonable to say that these distances have to be minimal for the trajectory of a real agent. So, now we for-

mulate an optimization problem as follows: to find the sequences of selection matrices that minimizes the sum of distances:

$$\bigcup_{\tau_s^0 \le t \le \tau_s^f} \mathbf{a}_s^t = \arg\min_{\mathbf{a}_s^t} \sum_{t \in [\tau]_s} (D_s^t + \Delta_s^t), \tag{17}$$

where the notation $[\tau]_s$ is short for the time domain $[\tau_s^0 + \tau_s^f]$ of a trajectory $\mathbf{O}_s$ (τ_s^0 is the first frame and τ_s^f is the last frame of the domain).

This problem can be solved by recurrent growing:

$$\bigcup_{t \le \tau} \mathbf{a}_s^t = \left(\bigcup_{t \le \tau - 1} \mathbf{a}_s^t \right) \cup \mathbf{a}_s^\tau, \tag{18}$$

where

$$\mathbf{a}_s^\tau = \arg\min_{\mathbf{a}_s^t} (D_s^\tau + \Delta_s^\tau). \tag{19}$$

The problem in Eq. (19) can be solved with a direct search for the minimum of the objective function $D_s^t + \Delta_s^t$ with $2N(t)^K$ possible search combinations. The final solution for the trajectory clustering $\mathbf{O}_s$ is as follows:

$$\mathbf{O}_s = \bigcup_{t \in [\tau]_s} \mathbf{a}_s^t \circ \mathbf{Q}^t. \tag{20}$$

In order to determine the time-domain interval $[\tau_s^0 + \tau_s^f]$ we follow two simple rules. The first one determines a constant τ_s^0, which coincides with the frame index where an agent s appeared for the first time. So, at a certain snapshot time some trajectories points can remain unselected by previously existing trajectories. The index of such a frame becomes a generating time τ_s^0 for a new trajectory $\mathbf{O}_s$. The constant τ_s^f of the trajectory $\mathbf{O}_s$ is determined by comparison of a distance term D_s^t with an experimentally determined threshold D^{Thr}. If $D^{\mathrm{Thr}} \le D_s^t$, then $\tau_s^f = t$.

It is natural to derive the threshold parameter D^{Thr} from another parameter that is related to the accuracy of the floor point position estimation. If the dynamics of the trajectories coordinates is considered, it is possible to apply the Kalman filter[21,22] and then use the innovation covariance matrix for this purpose. However our experiments showed that the simple mean-value filter and its variance can effectively replace the more complicated Kalman filter. Let us describe the estimation of a floor point $\mathbf{p}^{t+1}$ with the mean value filter:

$$\mathbf{p}^{t+1} = \frac{\hat{\mathbf{p}}^{t+1} + \mathbf{p}^{t+1|t}}{2}, \tag{21}$$

where $\hat{\mathbf{p}}^{t+1}$ is a measured value of the vector, and the dynamic prediction $\mathbf{p}^{t+1|t}$ is the position at the moment t plus $\Delta t\,(\mathbf{p}^t - \mathbf{p}^{t-1})/\Delta t$: the prediction velocity multiplied by the interframe time interval, or, after obvious simplification,

$$\mathbf{p}^{t+1|t} = 2\mathbf{p}^t - \mathbf{p}^{t-1}. \tag{22}$$

We define the variance of the measurement as

(a)

(b)

Fig. 6 (a) Snapshot of camera 1. (b) Snapshot of camera 2.

$$\delta^2 = \frac{1}{T} \sum_{t=1}^{T} (\mathbf{p}^t - \hat{\mathbf{p}}^t)^2, \tag{23}$$

where T is the total number of frames involved in an experiment. If a dynamic prediction of more than one step back is considered, the mean-value filter can also be replaced by more a robust median filter

$$\mathbf{p}^{t+1} = \mathbf{med}\{\hat{\mathbf{p}}^{t+1}, \mathbf{p}^{t+1|t}, \ldots, \mathbf{p}^{t+1|t-n}\}; \tag{24}$$

were $\mathbf{med}\{\}$ denotes the vector median operator, whose output is one of the points in the braces, and the total distance from this point to all other points is minimal. The prediction in this case, is as follows:

$$\mathbf{p}^{t+1|t-n} = \mathbf{p}^t + \frac{\mathbf{p}^t - \mathbf{p}^{t-n-1}}{n}. \tag{25}$$

Let us note that this kind of dynamic filter does not need predefined parameters like the two initial covariance matrices in the Kalman filter. Moreover, the value of the measurement covariance matrix usually depends on the position

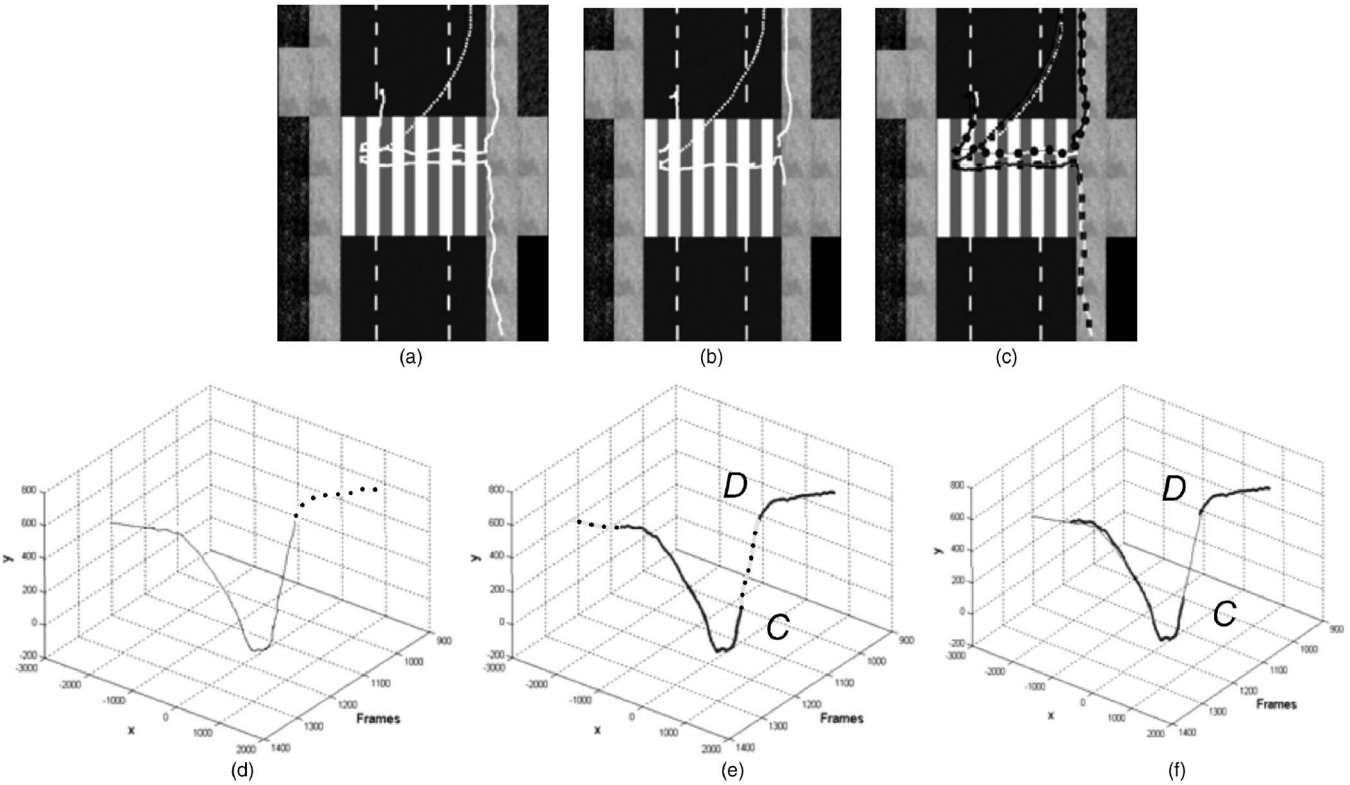


Fig. 7 Ground truth 2-D trajectories of two agents: (a) camera 1; (b) camera 2; (c) the result of tracking with the two camera views overlapping. 3-D trajectory of an agent in time-space domain: (d) camera 1; (e) camera 2; (f) the two camera views overlapping.

of the tracked agent, and this fact is contradictory with the basic condition of the Kalman filter: the constancy of the measurement noise parameter.

5 Experimental Results

Our method was tested with different sequences from two databases: the HERMES outdoor sequence[23] and the PETS.[24] Two frames taken from different viewpoints of the HERMES outdoor sequence scene are shown in Fig. 6. The full scenario of both databases includes multiple agents and different kinds of occlusions.

To explain the performance of the proposed approach we chose a typical situation of long-time occlusion, where two agents are visible from the point of view of camera 1 [Fig. 6(a)] during the tracking process. In the sequences obtained from the point of view of camera 2 one agent occludes another on the zebra crossing. In Fig. 7, parts 7(a) and 7(b) show the ground truth trajectory in the views of camera 1 and camera 2 respectively. The task is to divide a set of floor points [black spots in Fig. 7(c)] into two trajectories. The result of trajectory clustering is shown in Fig. 7(c), where two different shapes, circles and squares, represent the two trajectories. The comparison with the ground truth data superposed on the clustered floor points shows that all visible trajectories' points are selected properly.

Another property of the algorithm is its ability to complete the invisible part of a trajectory (from the viewpoint of camera 2) by virtual floor points. This property is illustrated in Fig. 7(d)–7(f). Here, the trajectory of the first agent in the time-space representation is shown in Fig. 7(d) and 7(e). The set of trajectory points for camera 2 includes a segment with virtual points: the segment $[C,D]$ in Fig. 7(e). To determine the real visibility of this part of the trajectory, the algorithm compares related visible coordinates of trajectory points of the segment with the visible coordinates of the interacting agent trajectory, using the distance order criterion described in Sec. 4.1. The result of such a comparison is shown in Fig. 7(f), where the segment $[C,D]$ is invisible.

To execute the trajectory tracking algorithm we need to choose the thresholds described in Sec. 4. The maximal error due to the camera calibration process is 12 cm, which is less than the dynamically calculated accuracy of the algorithm. The maximal error due to the segmentation algorithm in nonoccluded segments of the trajectories is 43 cm, and the variance from Eq. (23) is 19 cm. In our experiments we used the maximal error as the threshold D^{Thr}.

Another aspect of the error analysis is possible algorithm failure. Single-person tracking is robust. Errors can occur when multiple people interact. In our approach such failure can occur if the set $\mathbf{P}_s^t$ includes only virtual or invisible points—in other words, an agent is invisible from all available viewpoints or several blobs are merged. In this case our algorithm needs the support of other techniques to recover the lost trajectory. The template matching algorithm might be used, as for example, in Ref. 13. However, we should note that in our approach such events are quite rare.

The computational complexity of the algorithm is the

sum of computational costs of the blocks that are used during the tracking. Application of the mean-value or the Kalman filter has the lowest cost, proportional to the number of tracked agents per frame. The computational cost of the proposed trajectory clustering is proportional to $N(t)^K$ per frame. Thus, for a real situation the clustering procedure can be considered a low-cost operation.

Almost all known agent-tracking algorithms include a background substation and a segmentation procedure. The computational complexity of such a procedure is proportional to the number of pixels in the frame. The matching algorithm (which can include, for example, motion estimation or vector support machine methods) usually is the most time-consuming block of the general procedure.

We cannot compare our trajectory clustering algorithm with others for a case of long-time occlusion, because most related works do not even consider this problem. However, we can compare our techniques with the related works in the case of short-time occlusion. The closest approach to ours is described in Ref. 13; moreover, we started our research using algorithms proposed in that work. We found that in more than 70% of the failures that occur utilizing its algorithm one of the floor points involved in the interaction is visible. Thus, the application of our clustering algorithm can restrict the computational complexity of the tracking. Indeed, in 70% of the cases we do not use the most costly operation of template matching.

6 Conclusions

A novel trajectory-matching algorithm for multiple-view tracking is presented. The experimental results show that it is possible to track trajectories effectively, even in the presence of long-time occlusion, using only the ground plane points of the agents. This method uses a soundly developed mathematical model, which makes the tracking process robust and highly efficient. We also propose a simple calibration method, which is adequate to handle multiple-camera systems.

Acknowledgments

This work has been supported by EC grant IST-027110 for the HERMES project and by the Spanish MEC under projects TIC2003-08865 and DPI-2004-5414. M. Mozerov acknowledges the support of the Ramon y Cajal research program, MEC, Spain, and J. Gonzàlez also acknowledges the support of a Juan de la Cierva postdoctoral fellowship from the Spanish MEC.

References

1. T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Comput. Vis. Image Underst.* **81**, 231–268 (2001).
2. L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recogn.* **36**, 585–601 (2003).
3. T. B. Moeslund, A. Hilton, and W. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Int. Comput. Vision Image Understanding* **104**(2–3), 90–126 (2006).
4. T. I. Haritaoglu, D. Harwood, and L. S. Davis, "W/sup 4/: real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 809–830 (2000).
5. C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using realtime tracking," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 747–757 (2000).
6. S. Park and J. K. Aggarwal, "A hierarchical Bayesian network for event recognition of human actions and interactions," *ACM J. Multimedia Systems* **10**(2), 164–179 (2004).
7. R. Cucchiara, C. Grana, G. Tardini, and R. Vezzani, "Probabilistic people tracking for occlusion handling," in *Proc. IEEE* I-132–135 (2004).
8. W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviours," *IEEE Trans. Syst. Man Cybern.* **34**, 334–352 (2004).
9. A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.* **25**(10), 1296–1311 (2003).
10. M. Hu, W. Hu, and T. Tan, "Tracking people through occlusion," in *Proc. 17th Int. Conf. on Pattern Recognition*, Vol. **2**, pp. 724–727 (2004).
11. P. Guha, A. Mukerjee, and K. S. Venkatesh, "Efficient occlusion handling for multiple agent tracking by reasoning with surveillance event primitives," in *Proc. 2nd Joint IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 49–56 (2005).
12. P. Peursum, S. Venkatesh, and G. West, "Observation-switching linear dynamic systems for tracking humans through unexpected partial occlusions by scene objects," in *Proc. IAPR Int. Conf. on Pattern Recognition*, Vol. **4**, pp. 929–934 (2006).
13. Q. Zhou and J. K. Aggarwal, "Object tracking in an outdoor environment using fusion of features and cameras," *Image Vis. Comput.* **24**, 1244–1255 (2006).
14. Y. I. Abdel-Aziz and H. M. Karara, "Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry," in *Proc. ASP Symp. on Close-Range Photogrammetry*, pp. 1–18 (1971).
15. R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE J. Rob. Autom.* **3**, 323–344 (1987).
16. Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 1330–1334 (2000).
17. L. Lee, R. Romano, and G. Stein, "Monitoring activities from multiple video streams: establishing a common coordinate frame," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 758–767 (2000).
18. R. Haralick, C. Lee, K. Ottenberg, and M. Nolle, "Review and analysis of solutions of the three point perspective pose estimation problem," *Int. J. Comput. Vis.* **13**, 331–356 (1994).
19. D. Nistér, "A minimal solution to the generalized 3-point pose problem. On plane-based camera calibration: a general algorithm, singularities, applications," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. I-560–567 (2004).
20. T. Yang, S. Z. Li, Q. Pan, and J. Li, "Real-time multiple objects tracking with occlusion handling in dynamic scenes," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. I-970–975 (2005).
21. I. Huerta, D. Rowe, M. Mozerov, and J. Gonzàlez, "Improving background subtraction based on a casuistry of colour-motion segmentation problems," in *Proc. 3rd Iberian Conf. on Pattern Recognition and Image Analysis*, pp. 475–482 (2007).
22. "Kalman filter," Wikipedia, http://en.wikipedia.org/wiki/Kalman_filter.
23. HERMES Outdoor Database, http://iselab.cvc.uab.es/?q=tools.
24. PETS test sequences, http://www.cvg.cs.rdg.ac.uk/cgi-bin/PETSMETRICS/page.cgi?dataset.