A Novel Method for Tracking and Counting Pedestrians in Real-Time Using a Single Camera

Osama Masoud and Nikolaos P. Papanikolopoulos, *Senior Member, IEEE*

***Abstract*—This paper presents a real-time system for pedestrian tracking in sequences of grayscale images acquired by a stationary camera. The objective is to integrate this system with a traffic control application such as a pedestrian control scheme at intersections. The proposed approach can also be used to detect and track humans in front of vehicles. Furthermore, the proposed schemes can be employed for the detection of several diverse traffic objects of interest (vehicles, bicycles, etc.) The system outputs the spatio-temporal coordinates of each pedestrian during the period the pedestrian is in the scene. Processing is done at three levels: raw images, blobs, and pedestrians. Blob tracking is modeled as a graph optimization problem. Pedestrians are modeled as rectangular patches with a certain dynamic behavior. Kalman filtering is used to estimate pedestrian parameters. The system was implemented on a Datacube MaxVideo 20 equipped with a Datacube Max860 and was able to achieve a peak performance of over 30 frames per second. Experimental results based on indoor and outdoor scenes demonstrated the system's robustness under many difficult situations such as partial or full occlusions of pedestrians.**

***Index Terms*—Applications, image sequence analysis, intelligent trasportation systems, pedestrian control at intersections, pedestrian tracking, real-time tracking.**

I. INTRODUCTION

THERE is a wealth of potential applications of pedestrian tracking. Different applications, however, have different requirements. Tracking systems suitable for virtual reality applications and for those measuring athletic performance require that specific body parts be robustly tracked. In contrast, security monitoring, event recognition, pedestrian counting, traffic control, and traffic-flow pattern identification applications emphasize tracking on a coarser level. Here all individuals in the scene can be considered as single indivisible units. Of course, a system that can perform simultaneous tracking on all different scales is highly desirable but until now, no such system exists. A few systems that track body parts of one person [24], [13], [16] and two persons [6] have been developed. It remains to be seen whether these systems can be generalized to track an arbitrary number of pedestrians.

The work described in this paper targets the second category of applications [12], [10]. The proposed approach has a large number of potential applications which extend beyond pedestrians. For example, it cannot only detect and track humans in front of or around vehicles but it can also be employed to track several diverse traffic objects of interest (vehicles in weaving sections, bicycles, rollerbladers, etc.). One should note that the reliable detection and tracking of traffic objects is important in several vehicular applications (e.g., parking sensory aids, lane departure avoidance systems, etc.). In this paper, we are mainly interested in applications related to traffic control with the goal of increasing both safety and efficiency of existing roadways. Information about pedestrians crossing the streets would allow for automatic control of traffic lights at an intersection, for example. Pedestrian tracking also allows the use of a warning system, which can warn drivers and workers at a work zone from possible collision risks.

Several attempts have been made to track pedestrians as single units. Baumberg and Hogg [3] used deformable templates to track the silhouette of a walking pedestrian. The advantage of their system is that it is able to identify the pose of the pedestrian. Tracking results were shown for one pedestrian in the scene and the system assumed that overlap and occlusions are minimal [2]. Another use of the silhouette was made by Segen and Pingali [19]. In their case, features on the pedestrian silhouette were tracked and their paths were clustered. The system ran in real-time but was not able to deal well with temporary occlusions. Occlusions and overlaps seem to be a primary source of instability for many systems. Rossi and Bozzoli [17] avoided the problem by mounting the camera vertically in their system which aimed to mainly count passing pedestrians in a corridor. Such a camera configuration, however, may not be feasible in some cases. Occlusions and overlaps occur very commonly in pedestrian scenes; and cannot be ignored by a pedestrian tracking system. The use of multiple cameras can alleviate the occlusion problem. Cai and Aggarwal [4] tracked pedestrians with multiple cameras. The system, however, did not address the occlusion problem in particular but rather how to match the pedestrian across different camera views. The switching among cameras was done manually. Smith *et al.* [22] performed pedestrian detection in real-time. The system used several simplistic criteria to judge whether the detected object is a pedestrian or not but did not actually track pedestrians.

Shio and Sklansky [20] presented a method for segmenting people in motion with the use of correlation techniques over successive frames to recover the motion field. Iterative merging was then used to recover regions with similar motion. The method

Manuscript received April 22, 1997; revised January 28, 2001. This work was supported by the Minnesota Department of Transportation through Contracts #71 789-72 983-169 and #71 789-72 447-159, the Center for Transportation Studies through Contract #USDOT/DTRS 93-G-0017-01, the National Science Foundation through Contracts #IRI-9 410 003 and #IRI-9502245, the Department of Energy (Sandia National Laboratories) through Contracts #AC-3752D and #AL-3021, and the McKnight Land-Grant Professorship Program at the University of Minnesota.

The authors are with the Artificial Intelligence, Robotics, and Vision Laboratory, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: masoud@cs.umn.edu; npapas@cs.umn.edu).

Publisher Item Identifier S 0018-9545(01)08257-3.

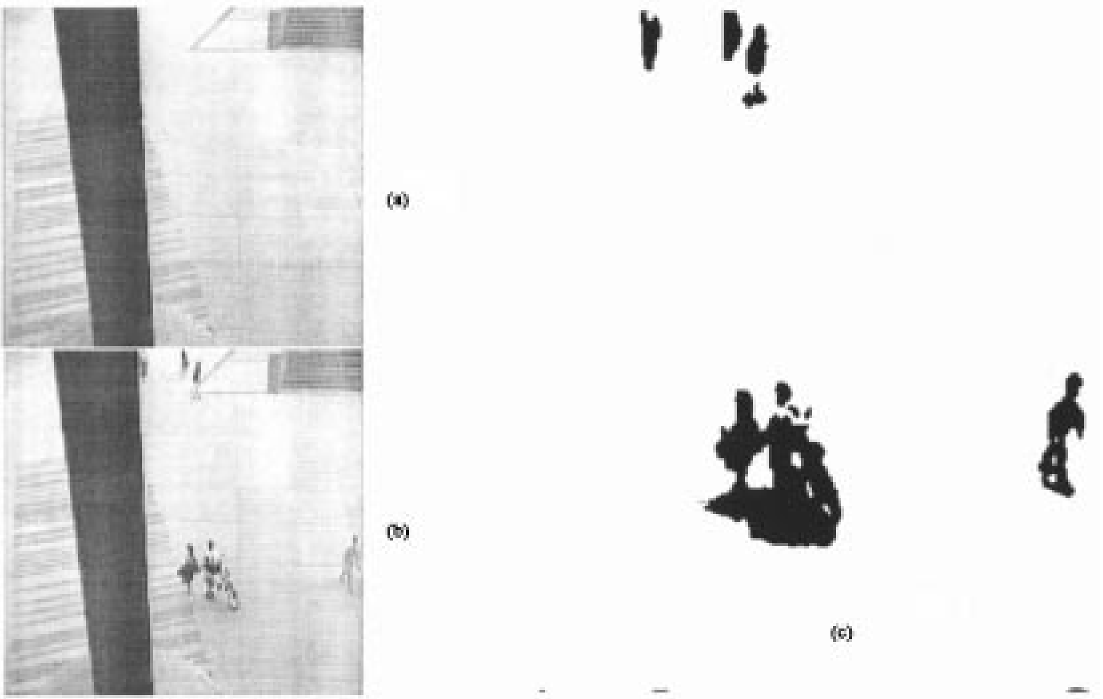


Fig. 1. (a) Background image. (b) Foreground. (c) Difference image showing that a blob does not always correspond to one pedestrian.

was able to deal with partial occlusions. The assumption was that pedestrians do not change direction as they move. A disadvantage of this method is the high computational cost of the correlation and the iterative merging steps. An interesting approach which was presented by Heisele *et al.* [9] is based on their earlier work on color cluster flow [8]. An image is clustered into regions of similar color. In the subsequent images, the clusters are updated using a k-means clustering algorithm. Assuming that the pedestrian legs form one cluster, a step to recognize legs enables the system to recognize and track pedestrians. This was done by checking the periodicity of the cluster shape and by feeding the gray scale images of the legs into a time delay neural network. The advantage of this approach is that it works in the case of a moving camera. Unfortunately, due to several costly steps, real-time implementation was not possible. Lipton *et al.* [14] performed classification and tracking of vehicles and pedestrians. They used a simple criterion for classification and template matching, guided by motion detection for tracking. The system was able to track multiple isolated pedestrians and vehicles robustly. The classification step is critical since the template that is used for tracking is decided based on it. More recently, Haritaoglu *et al.* [7] successfully tracked multiple pedestrians as well as their body parts. The system made use of silhouette analysis in finding body parts which can be sensitive to occlusions.

In developing our method, robustness in arbitrary input scenes with arbitrary environmental conditions without compromising real-time performance was the primary motivation. Our approach does not have a restriction on the camera position. More importantly, we do not make any assumptions about occlusions and overlaps. Our system uses a single fixed camera mounted at an arbitrary position. We use simple rectangular patches with a certain dynamic behavior to model pedestrians. Overlaps and occlusions are dealt with by allowing pedestrian models to overlap in the image space and by maintaining their existence in spite of the disappearance of some cues. The cues that we use are blobs obtained by thresholding the result of subtracting the image from the background.

Our choice of using blobs obtained after background subtraction is motivated by the efficiency of this preprocessing step even though some information is permanently lost. In a typical scene, a blob obtained this way does not always correspond to a single pedestrian. An example is shown in Fig. 1. This is the main source of weakness in many of the systems mentioned above which assume a clean one-to-one correspondence between blobs and pedestrians. In our system, we allow maximum flexibility by allowing this relation to be many-to-many. This relation is updated iteratively depending on the observed blobs behavior and predictions of pedestrians behavior. Fig. 2 gives an overview of the system. Three levels of abstraction are used. The lowest level deals with raw images. In the second level, blobs are computed and subsequently tracked. Tracked blobs are passed on to the pedestrians level where relations between pedestrians and blobs as well as information about pedestrians are inferred using previous information about pedestrians in that level.

At the images level, we perform background subtraction and thresholding to produce *difference images*. Background subtraction has been used by many to extract moving objects in the scene [2], [4], [19], [22]. Change detection algorithms that compare successive frames [11], [21] can also be used to extract motion. However, these algorithms also output regions in the background that get uncovered by the moving object as well which is undesirable in our case. The choice of the threshold is critical. Many thresholding techniques [15], [18] work very well when there is an object in the scene. This is because these techniques assume that the image to be thresholded contains two categories

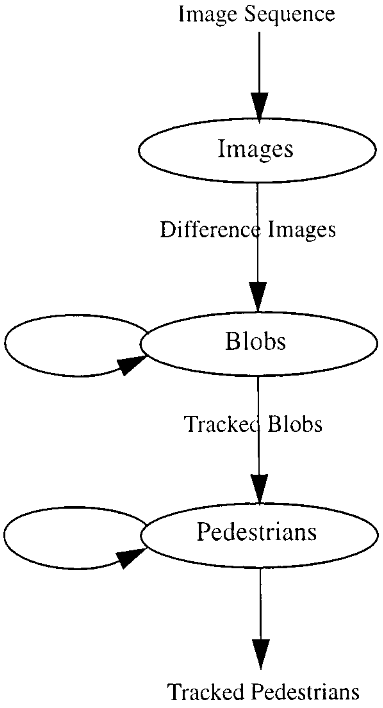


Fig. 2. The three levels of abstraction and data flows among them.

of pixel values and they try to separate the two. However, when there is only one category, the results become unpredictable. In our case, this happens often since the scene may not have any objects at one point in time. Instead, we used a fixed threshold value. The value was obtained by examining an empty background for a short while and measuring the maximum fluctuation of pixel values during this training period. The threshold is set to be slightly above that value. This technique worked sufficiently well for our purpose. Several measures were taken to further reduce the effect of noise. A single step of erosion followed by a step of dilation is performed on the resulting image and small clusters are totally removed. Also, the background image is updated using a very slow recursive function to capture slow changes in the background (e.g., changes in lighting conditions due to a passing cloud).

It should be noted that even with these precautions, in a real world sequence, the feature image may still capture some undesirable features (e.g., shadows, excessive noise, sudden change in lighting conditions, and trees moved by the wind (see Fig. 7 frame 104), etc.). It also may miss parts of moving pedestrians due to occlusions (see Fig. 8 frame 32) or color similarity between the pedestrian clothes and the background (see Fig. 8 frame 44). Our system handles the majority of these situation as will be explained in the subsequent sections.

The next section describes the processing done at the blobs level. The pedestrians level is presented in Section III. Experimental results follow in Section IV. Finally, conclusions are presented in Section V.

II. Blobs Level

In this level, we present a novel approach to track blobs regardless of what they represent. The tracking scheme attempts to describe changes in the difference image in terms of motion of blobs and by allowing blobs to merge, split, appear, and vanish. Robust blob tracking was necessary since the pedestrians level relies solely on information passed from this level. The first step is to extract blobs. Connected regions are extracted efficiently using boundary following [5]. Another way to extract connected components is to use the raster scan algorithm [5]. The advantage of the latter method is that it extracts holes inside the blobs while boundary following does not. However, for the purpose of our system, holes do not constitute a major issue. Moving pedestrians usually form solid blobs in the difference image and if these blobs have holes, they may be still considered part of the pedestrian. Boundary following has the extra advantage of being more efficient since the blob interior is not considered. The following parameters are computed for each blob b:

1) area—$A(b)$: the number of pixels inside the blob;
2) bounding box—the smallest rectangle surrounding the blob;
3) density—$D(b)$: $A(b)$/bounding box area;
4) velocity—$\mathbf{V}(b)$, calculated in pixels per second in horizontal and vertical directions.

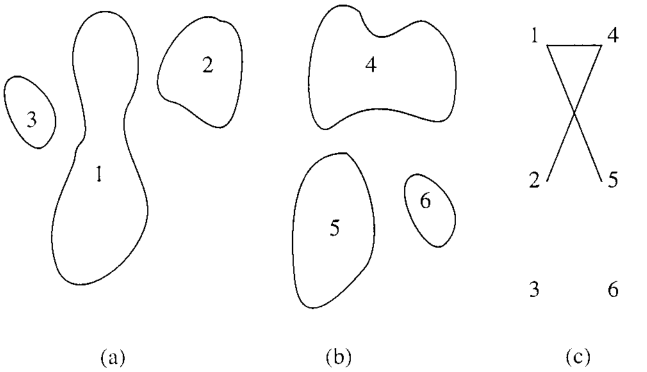


Fig. 3. (a) Blobs in frame $(i-1)$. (b) Blobs in frame i. (c) Relationship among blobs.

A. Blob Tracking

When a new set of blobs is computed for frame i, an association with frame $(i-1)$'s set of blobs is sought. Ideally, this association can be an unrestricted relation. With each new frame, blobs can split, merge, appear, or disappear. Examples of blob behavior can be seen in the blob images in Figs. 7 and 8. The relation among blobs can be represented by an undirected bipartite graph, $G_i(V_i, E_i)$, where $V_i = B_i \cup B_{i-1}$. B_i and B_{i-1} are the sets of vertices associated with the blobs in frames i and $i-1$, respectively. We will refer to this graph as a *blob graph*. Since there is a one-to-one correspondence between the blobs in frame i and the elements of B_i, we will use the terms blob and vertex interchangeably. Fig. 3 shows how the blobs in two consecutive frames are associated. The blob graph in the figure expresses the fact that blob 1 split into blobs 4 and 5, blob 2 and part of blob 1 merged to form blob 4, blob 3 disappeared, and blob 6 appeared.

The process of blob tracking is equivalent to computing G_i for $i = 1, 2, \ldots, n$, where n is the total number of frames. Let $N_i(u)$ denote the set of neighbors of vertex $u \in V_i$,

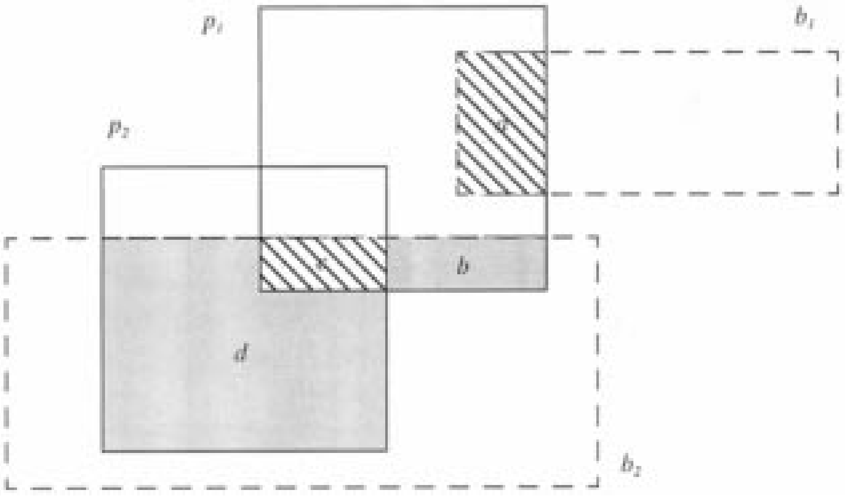


Fig. 4. Overlap area. Pedestrians p_1 and p_2 share blob b_2 while b_1 is only part of p_1 (see Section III-B3 for overlap area computation).

$N_i(u) = \{v \mid (u,v) \in E_i\}$. To simplify the graph computation, we will restrict the generality of the graph to those graphs which do not have more than one vertex of degree more than one in every connected component of the graph. This is equivalent to saying that from one frame to the next, a blob may not participate in a splitting and a merging at the same time. We refer to this as the *parent structure constraint*. According to this constraint, the graph in Fig. 3(c) is invalid. If, however, we eliminate the arc between 1 and 5 or the arc between 2 and 4, it will be a valid graph. This restriction is reasonable assuming a high frame rate where such simultaneous split and merge occurrences are rare. To further reduce the number of possible graphs, we use another constraint which we call the *locality constraint*. With this constraint, vertices can be connected only if their corresponding blobs have a bounding box overlap area which is at least half the size of the bounding box of the smaller blob. This constraint, which significantly reduces possible graphs, relies on the assumption that a blob is not expected to be too far from where it is predicted to be, taking into consideration its speed and location in the previous frame. This is also reasonable to assume if we have a relatively high frame rate. We refer to a graph which satisfies both the parent structure and locality constraints as a *valid* graph.

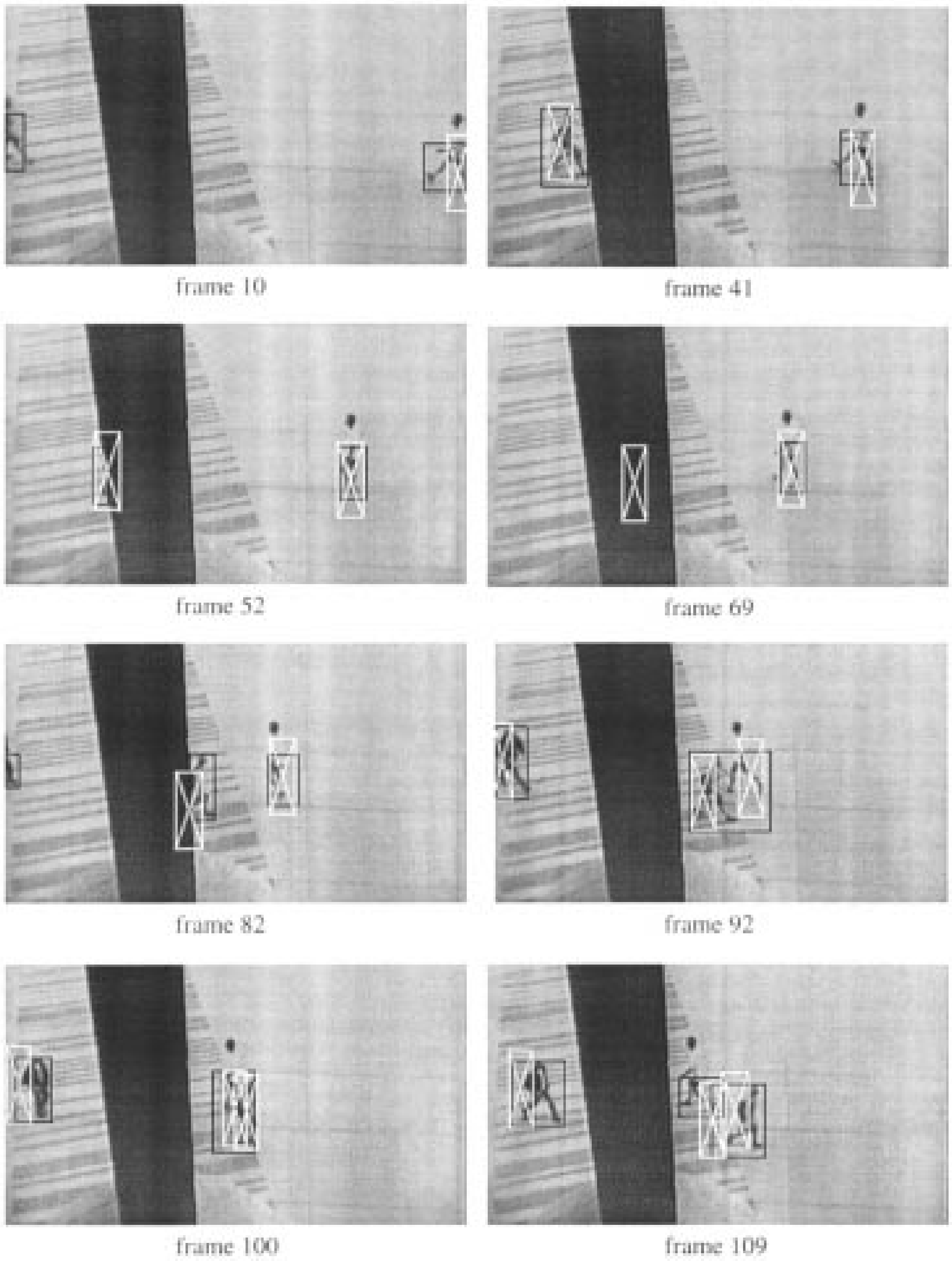
To find the optimum G_i, we define a cost function $C(G_i)$ so that different graphs can be compared. A graph with no edges, i.e., $E_i = \emptyset$, is one extreme solution in which all blobs in V_{i-1} disappear and all blobs in V_i appear. This solution has no association among blobs and should, therefore, have a high cost. In order to proceed with our formulation of the cost function, we define two disjoint sets, which we call *parents*, P_i, and *descendents*, D_i, whose union is V_i such that $D_i = \cup_{u \in P_i} N_i(u)$. P can be easily constructed by selecting from V_i all vertices of degree more than one, all vertices of degree zero, and all vertices of degree one that are only in B_i. Furthermore, let $S_i(u) = \sum_{v \in N_i(u)} A(v)$ be the total area occupied by the neighbors of u. The cost function that we use penalizes graphs in which blobs change significantly in size. A perfect match would be one in which blob sizes remain constant (e.g., the size of a blob that splits equals to the sum of the sizes of blobs it split into). We now write the formula for the cost function as

$$C(G_i) = \sum_{u \in P_i} \frac{|A(u) - S_i(u)|}{\max(A(u), S_i(u))}. \tag{1}$$

This function is a summation of ratios of size change over all parent blobs.

Using this cost function, we can proceed to compute the optimum graph. First, we notice that given a valid graph $G(V,E)$ and two vertices $u, v \in V$ such that $(u,v) \notin E$, the graph $G'(V, E \cup \{(u,v),(v,u)\})$ has a lower cost than G provided that G' is a valid graph. If it is not possible to find such a G', we call *G dense*. Using this property, we can avoid some useless enumeration of graphs that are not dense. In fact, this observation is the basis of our algorithm to compute the optimum G.

Our algorithm to compute the optimum graph works as follows: A graph G is constructed such that the addition of any edge to G makes it violate the locality constraint. There can be only one such graph. Note that G may violate the parent structure constraints at this moment. The next step in our algorithm systematically eliminates just enough edges from G to make it satisfy the parent structure constraint. The resulting graph is valid and also dense. The process is repeated so that all possible dense graphs are generated. The optimum graph is the one with the minimum cost. By systematically eliminating edges, we are effectively enumerating valid graphs. The computational complexity of this step is highly dependent on the graph being considered. If the graph already satisfies the parent structure constraint, it is $O(1)$. On the other hand, if we have a fully connected graph, the complexity is exponential in the number of vertices. Fortunately, because of the locality constraint and the high frame rate, the majority of graphs considered already satisfy the parent structure constrained. Occasionally, a small cluster of the graph may not satisfy the parent structure constraint and the algorithm will need to enumerate a few graphs. In practice, the algorithm never took more than a few milliseconds to execute even in the most cluttered scenes. Other techniques to find the optimum (or near optimum) graph (e.g., stochastic relaxation using simulated annealing) can also be used. The main concern, however, would be their efficiency which may not be appropriate for this real-time application due to their iterative nature.



(a)

Fig. 5. (a) A number of snapshots from the input sequence overlaid with pedestrian boxes shown in white and blob boxes shown in black.

At the end of this stage, we use a simple method to calculate the velocity of each blob v based on the velocities of the blobs at the previous stage and the computed blob graph. The blob velocity will be used to initialize pedestrian models as described later. If v is the outcome of a splitting operation, it will be assigned the same velocity as the parent blob. If v is the outcome of a merging operation, it will be assigned the velocity of the largest child blob. If v is a new blob, it will be assigned zero velocity. Finally, if there is only one blob, u, related to v, the velocity is computed as

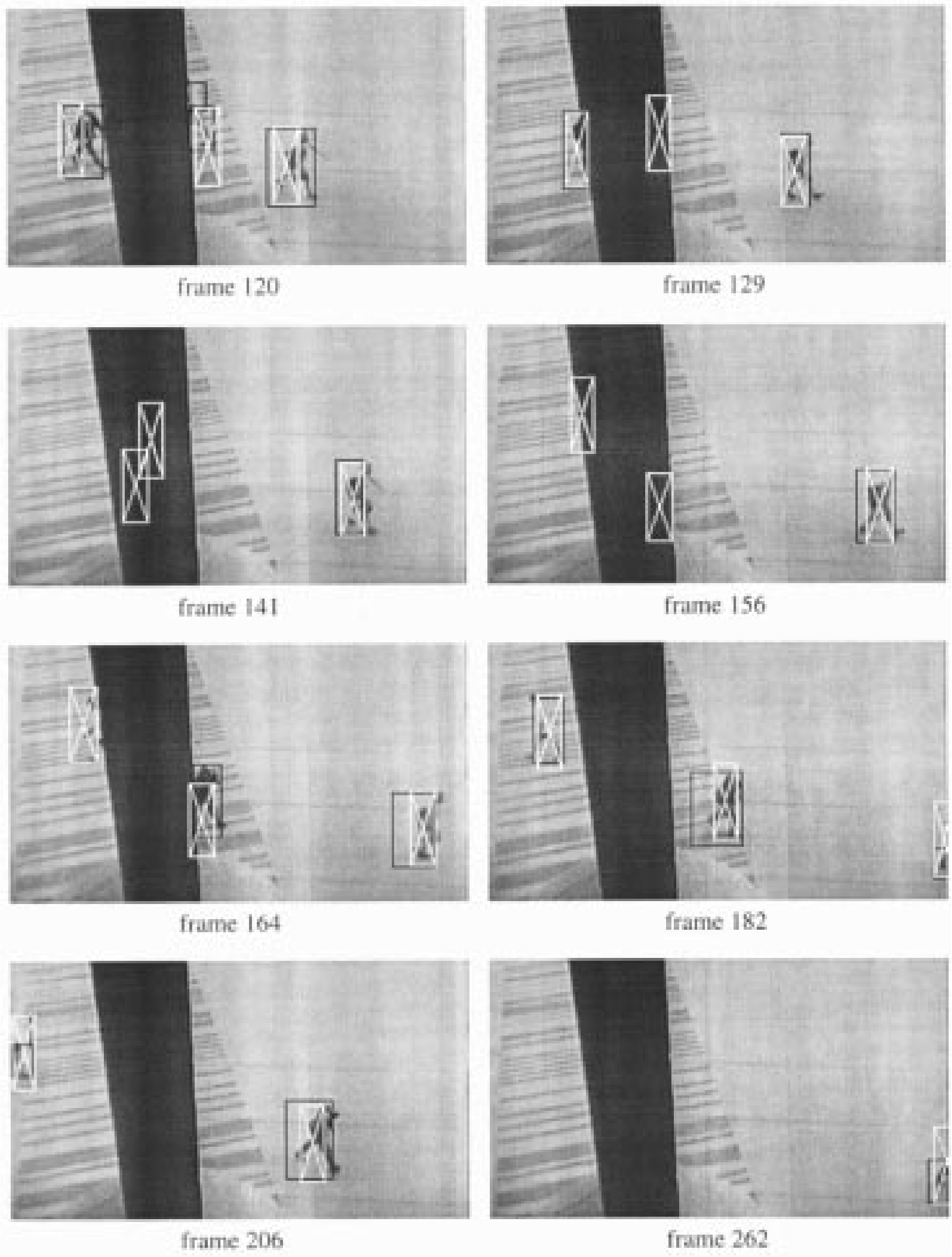
$$\mathbf{V}(v) = \beta\frac{(\mathbf{b}_v - \mathbf{b}_u)}{\delta t} + (1-\beta)\mathbf{V}(u) \tag{2}$$

where

$\mathbf{b}_v$ and $\mathbf{b}_u$	centers of the bounding boxes of v and u, respectively;
β	weight factor set to 0.5 (found empirically);
δt	sampling interval since the last stage.

III. Pedestrians Level

The input to this level is tracked blobs and the output is the spatio-temporal coordinates of each pedestrian. The relationship between pedestrians and blobs in the image is not necessarily one to one. A pedestrian wearing clothes that are close in color to the background may show up as more than one blob. Partially occluded pedestrians may also result in more than one blob or



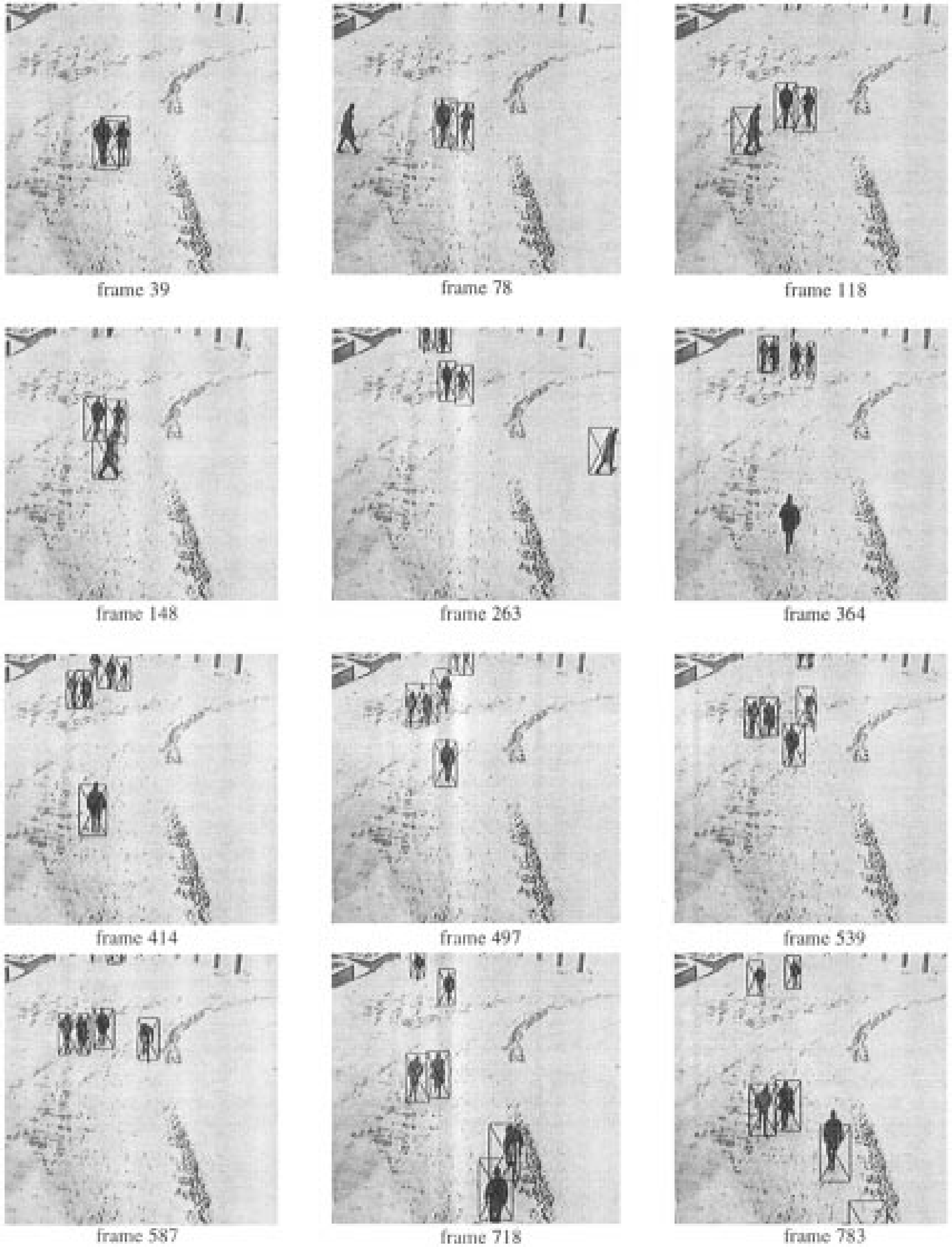
(b)

Fig. 5. (*Continued.*) (b) A number of snapshots from the input sequence overlaid with pedestrian boxes shown in white and blob boxes shown in black.

even in no blobs at all if the pedestrian is fully occluded. Two or more pedestrians walking close to each other may give rise to a single blob. For this reason, it was necessary to make the pedestrians level capable of handling all the above cases. We do this by modeling the pedestrian as a rectangular patch with a certain dynamic behavior. We found that for the purpose of tracking, this simple model adequately resembles the pedestrian shape and motion dynamics. We now present this model in more detail and then describe how tracking is performed.

A. Pedestrian Model

Pedestrians usually walk with a constant speed. Moreover, the speed of a pedestrian usually changes gradually when the pedestrian desires to stop or start walking. Our approach assumes that motion in the scene is constrained to a plane (also called the ground-plane constraint). With this assumption, back projection from the scene to the image plane can be performed (with the knowledge of the camera geometry) to determine the expected dimensions and dynamic behavior of the pedestrian in the image coordinate system. Small variations in ground elevation will still be tolerated especially in distant areas. This restriction can be removed if the scene topology can be determined *a priori*. Camera geometry can be obtained using calibration techniques. A suitable technique for traffic scenes which makes use of the ground-plane constraint is given in [23].



frame 39 frame 78 frame 118

frame 148 frame 263 frame 364

frame 414 frame 497 frame 539

frame 587 frame 718 frame 783

Fig. 6. A number of snapshots from the input sequence in a snowy afternoon overlaid with pedestrian boxes shown in black.

The pedestrian is modeled as a rectangular patch whose dimensions depend on its location in the image. The dimensions are equal to the projection of the dimensions of an average size pedestrian at the corresponding location in the scene. The patch is assumed to move with constant velocity in the scene coordinate system. The patch acceleration is modeled as zero mean, Gaussian noise to accommodate for changes in velocity. Given a sampling interval δt, the discrete-time dynamic system for the pedestrian model can be described by the following equation:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{v}_t \tag{3}$$

where

$\mathbf{x} = [x\ \dot{x}\ y\ \dot{y}]^T$ state vector consisting of the pedestrian location, (x, y) and velocity, $(\dot{x}, \dot{y})$;

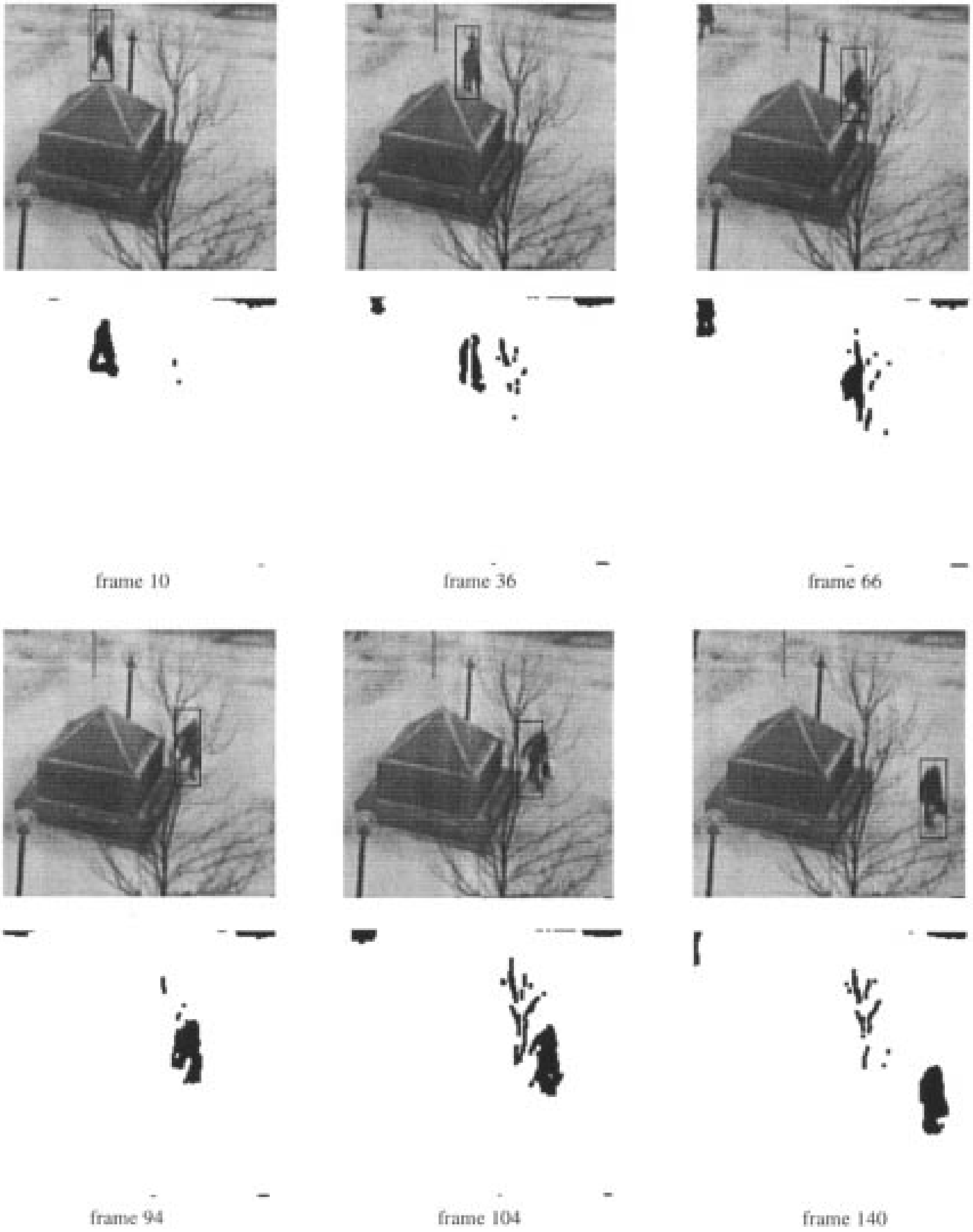


Fig. 7. Tracking sequence of a pedestrian in different occlusion situations.

$\mathbf{F}$ transition matrix of the system given by $\begin{bmatrix} 1 & \delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$;

$\mathbf{v}_t$ sequence of zero-mean, white, Gaussian process noise with covariance matrix $\mathbf{Q}$.

(x, y) denotes the location of the pedestrian in the scene. $\mathbf{Q}$ is computed as in [1] (p. 84) to become $\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} q$ where $\mathbf{A} = \begin{bmatrix} (\delta t)^3/3 & (\delta t)^2/2 \\ (\delta t)^2/2 & \delta t \end{bmatrix}$ and q represents the variance of the acceleration.

B. Pedestrian Tracking

Tracking pedestrians depends on the current state of pedestrians as well as the input to pedestrians level which is the tracked blobs. In our system, we use extended Kalman filtering (EKF) to estimate pedestrian parameters. We maintain a many-to-many relationship between pedestrians and blobs and

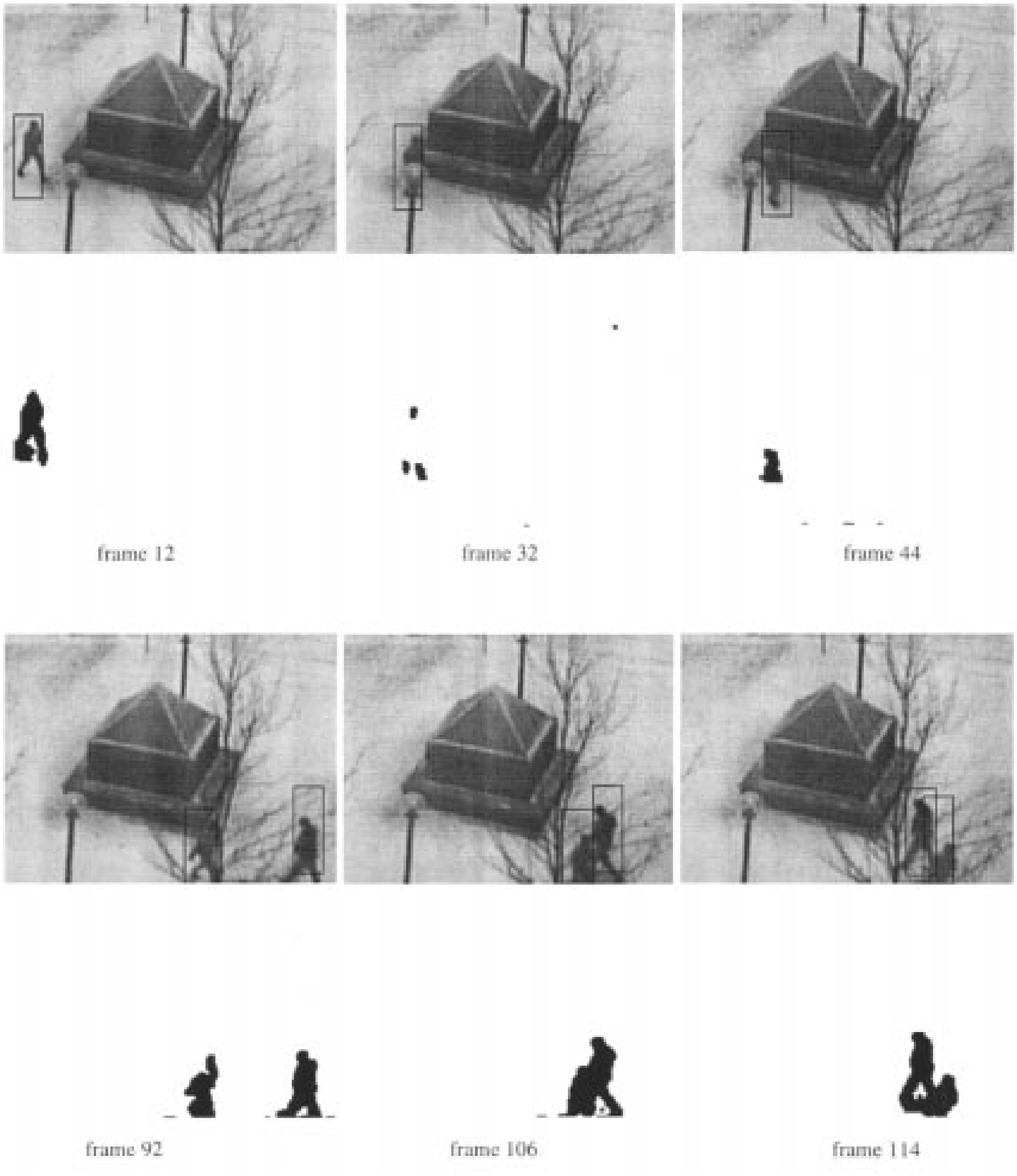


Fig. 8. Tracking sequence demonstrating occlusions and pedestrian overlap.

then use it to provide measurements to the filter. The next five sections describe one tracking cycle.

1) Relating Pedestrians to Blobs: We represent the relationship between pedestrians and blobs as a directed bipartite graph. We relate pedestrians to blobs using a simple rule: if a pedestrian was related to a blob in frame $(i-1)$ and that blob is related to another blob in the ith frame (through a split, merge, etc.), then the pedestrian is also related to the latter blob.

2) Prediction: Given the system equation as in Section III-A, the prediction phase of the Kalman filter is given by the following equations:

$$\begin{aligned}\hat{\mathbf{x}}_{t+1} &= \mathbf{F}\mathbf{x}_t \\ \hat{\mathbf{P}}_{t+1} &= \mathbf{F}\mathbf{P}_t\mathbf{F}^T + \mathbf{Q}.\end{aligned} \tag{4}$$

Here, $\hat{\mathbf{x}}$ and $\hat{\mathbf{P}}$ are the predicted state vector and state error covariance matrix, respectively. $\mathbf{x}$ and $\mathbf{P}$ are the previously estimated state vector and state error covariance matrix, respectively.

3) Calculating Pedestrian Positions: In this step, we use the predicted pedestrian locations as starting positions and we employ a two-dimensional (2-D) search to locate the pedestrian. The search attempts to find the best overlap between the pedestrian patch and the blobs assigned to the pedestrian. The overlap computation takes into consideration the density of the blobs as well as the possibility that a blob is shared by more than one pedestrian. Fig. 4 illustrates this computation. The overlap area for p_1 is computed as $a \times D(b_1) + b \times D(b_2) + (c \times D(b_2)/2)$. For p_2, the overlap area is $d \times D(b_2) + (c \times D(b_2)/2)$. To perform the search, we employ a heuristic solution using relaxation. First, a large step size is chosen. Then, each pedestrian is moved in all possible directions by the step size and the location (including the original location) which maximizes the overlap area is recorded. All pedestrian locations are then updated according to the recorded locations. This completes one iteration. In each following iteration, the step size is decreased. In our implementation, we start with a step of 64 pixels and halve the step size in each iteration until 1 pixel step size is reached.

The resulting locations form the measurements that will be fed back into the EKF to produce the new state estimates. Moreover, we use the overlap area to provide feedback about the measurement confidence by setting the measurement error standard deviation, which is described below, to be inversely proportional to the ratio of the overlap area to the pedestrian area. That is, the smaller the overlap area, the less confident the measurement is considered.

4) Estimation: A measurement is a location in the image coordinate system as computed in the previous section, $\mathbf{z} = [u\ v]^T$. Measurements are related to the state vector by

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{w}_t \tag{5}$$

where $\mathbf{h}$ is the measurement function and $\mathbf{w}_t$ is a sequence of zero-mean, white, Gaussian measurement noise with covariance $\mathbf{R}_t$ given by $\left[\begin{smallmatrix}\sigma_t^2 & 0\\ 0 & \sigma_t^2\end{smallmatrix}\right]$. The measurement error standard deviation σ_t depends on the overlap area computed in the previous section. Pedestrian locations are expressed in world coordinates resulting in $\mathbf{h}$ being a nonlinear function which performs projection into image coordinates. We let $\mathbf{H}$ be the Jacobian of $\mathbf{h}$. The EKF state estimation equations become

$$\begin{aligned}\mathbf{K}_{t+1} &= \hat{\mathbf{P}}_{t+1}\mathbf{H}^T(\mathbf{H}\hat{\mathbf{P}}_{t+1}\mathbf{H}^T + \mathbf{R}_t)^{-1}\\ \mathbf{x}_{t+1} &= \hat{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{h}(\hat{\mathbf{x}}_{t+1}))\\ \mathbf{P}_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})\hat{\mathbf{P}}_{t+1}.\end{aligned} \tag{6}$$

where $\mathbf{K}_{t+1}$ is the Kalman gain at $t + 1$.

The estimated state vector $\mathbf{x}_{t+1}$ is the outcome of the pedestrians level. The dimensions of the pedestrian patch are computed based on the estimated pedestrian location.

5) Refinement: At the end of this stage, we perform some checks to refine the pedestrian-blob relationships since pedestrians have been relocated. These can be summarized as follows.

1) If the overlap area between a pedestrian and one of its blobs becomes less than a certain threshold (a percentage of the size of both), it will no longer be considered belonging to this pedestrian. This serves as the splitting procedure when two pedestrians walk past each other. This threshold determines the degree of stickiness between blobs and pedestrians. In our experiments, a threshold of 10% gave the best tracking performance.
2) If the overlap area between a pedestrian and a blob that does not belong to any pedestrian becomes more than the threshold mentioned above, the blob will be added to the pedestrian blobs. This makes the pedestrian reacquire some blobs that may have disappeared due to occlusion.
3) Look for a cluster of blobs, which are not related to any pedestrians and whose age is larger than a threshold (i.e., have been successfully tracked for a certain number of frames). A new pedestrian may be initialized only if it will become sufficiently covered by the blobs cluster. The pedestrian is given an initial velocity equal to the average of the blobs velocities. This serves as the initialization step. The requirement on the age helps in reducing chances of unstable blobs being used to initialize pedestrians (for example, blobs resulting from tree motion caused by wind and blobs due to noise). A threshold of one second was sufficient in most cases.
4) Select one of the blobs which is already assigned one or more pedestrians but can accommodate more pedestrian patches. Create a new pedestrian for this blob as in 3). This handles cases in which a group of people form one big blob, which does not split. If we do not do this step, only one pedestrian would be assigned to this blob.

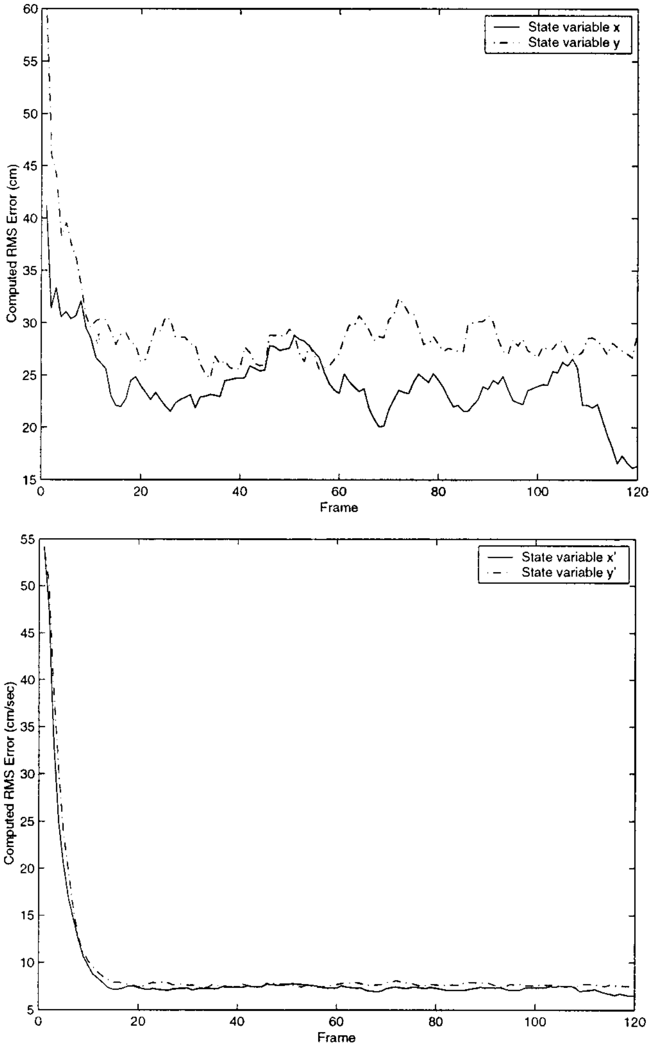


Fig. 9. Computed rms errors for a pedestrian in the sequence of Fig. 5.

IV. Experimental Results

The system was implemented on a Datacube MaxVideo 20 video processor, and a Datacube Max860 vector processor. It

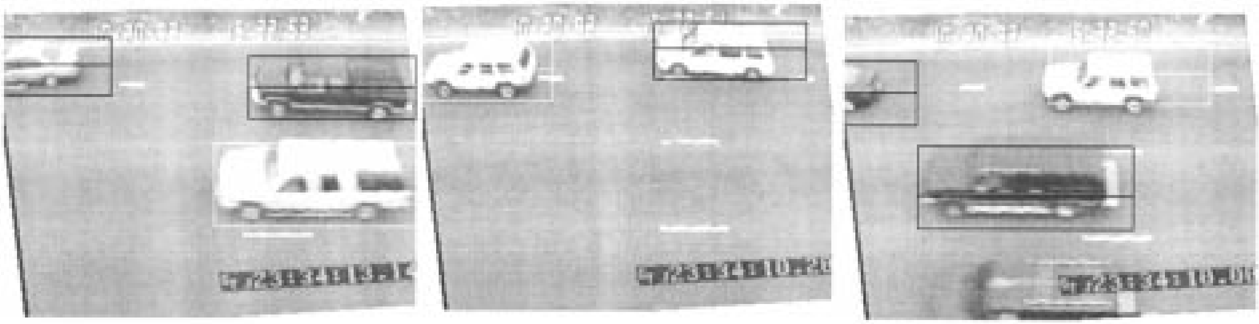


Fig. 10. A number of snapshots from a weaving section tracking sequence (the vehicles in the two lanes closer to the edge of the highway are tracked).

was later ported to a 400-MHz Pentium PC equipped with a C80 Matrox Genesis vision board.

The system was tested on several indoor and outdoor image sequences. Several outdoor sequences in different weather conditions (sunny, cloudy, snow, etc.) have been used. In most cases, pedestrians were tracked correctly throughout the period they appeared in the scene. Scenarios included pedestrians moving at a slow or very high speeds, partial and full occlusions, bicycles, and several pedestrian interactions. Interactions between pedestrians included occlusion of one another, repeated merging and splitting of blobs corresponding to two or more pedestrians walking together, pedestrians walking past each other, and pedestrians meeting and then walking back in the direction they came from. The system has a peak performance of 30 frames/s. In a relatively cluttered image with about six pedestrians, the frame processing rate dropped down to about 18 frames/s. Fig. 5 shows 16 snapshots spanning a sequence of 8.4 s. The sequence demonstrates the system behavior against occlusions—both partial and full. The figure also demonstrates how the system works at the blobs level. Blobs are shown by their black bounding boxes. Notice how tracking is preserved despite the lack of one-to-one correspondence between blobs and pedestrians. Fig. 6 shows 12 snapshots from a scene under different weather conditions. The snapshots span a sequence of 35 seconds. A more cluttered sequence is shown in Fig. 7. This sequence was taped during a snow storm. One can see the effect of the wind on the tree which resulted in false blobs in the difference images (frames 36, 66, 104, 140). The system deals with this situation in several ways.

1) If the blobs generated are too small, they are automatically eliminated.
2) A blob is considered at the pedestrian level only if it is tracked successfully for a certain period of time. This eliminates blobs that appear then disappear momentarily (such as most blobs that appear due to tree motion back and forth).
3) The system can be given information about the scene. In particular, the locations where pedestrians can be expected to appear. Thus, the system will not instantiate pedestrian boxes except at these locations.

Fig. 8 shows another tracking sequence which involves missing pedestrian blob information due to occlusion (frame 32) and similarity of color to the background (frame 44). It also shows what happens when two pedestrian walk past each other. Kalman filtering is essential here because it provides good prediction when there is little or no data. The blob-pedestrian relationship refinement procedures guarantee that the pedestrian will be related to the correct blobs when data is available again. Fig. 9 shows the computed root mean square (rms) errors for a pedestrians in the sequence of Fig. 5.

We also performed a pedestrian counting experiment for a sequence of 12 min in which 124 pedestrians were counted manually. The system gave a count of 130. Most of the failures were due to bicyclists who were double counted because the blob they generated was closer to the size of two pedestrians.

There are other cases where the system failed. Those include highly crowded images. Also, when a pedestrian becomes totally occluded but then reappears at an unexpected location, the pedestrian box will lose track. Shadows cast by pedestrian show as blobs in the difference image. When the shadows are not too large, the system can usually handle the situation by considering the shadow blob part of the pedestrian. However, large shadows are problematic and they can be confused as pedestrians. Part of our ongoing research addresses robust handling of the effects of shadows, which remains an open problem.

Overall, our system works well in a variety of real-world conditions. We feel that this is the major contribution and what makes our approach distinguishable from other approaches.

A. Other Applications: Monitoring Weaving Sections

To demonstrate the versatility of our approach, we recently used our system with slight modification to track vehicles. The idea was to track vehicles in short weaving sections that have more than one point of entry, followed by more than one point of exit. Sensors which are based on lane detection or trip-line detection often fail to monitor weaving sections since they cannot track vehicles which cross lanes. Our system was able to successfully track and record the speed of each vehicle. It also gave periodic averages of speeds of vehicles that belong to one of several categories. The categories were in the form of “vehicles that remained in lane X” and “vehicles that changed lanes from X to Y.” Example snapshots are given in Fig. 10.

V. Conclusion and Future Research

We presented a real-time model-based pedestrian tracking system capable of working robustly under many difficult circumstances such as occlusions and ambiguities. For each pedestrian in the view of the camera, the system produces location and velocity information as long as the pedestrian is visible. This data can be used by a scheduling algorithm to control walk signals at an intersection in order to increase the safety and efficiency of existing traffic systems.

There are several issues that still need to be addressed. Spatial interpretation of blobs is one such issue. In the current system, the only spatial attribute of blobs taken into consideration is the blob area. The shape of the blob can give a good clue on its contents. Although blobs obtained from difference images can be sufficient to decide the location of pedestrians in many cases, the intensity information may be useful to resolve certain ambiguities. The use of such information in the form of statistical distribution of intensities may add to the robustness of the current system and is worth pursuing.

In its current state, our system assumes that all the objects in the scene are pedestrians. This means that if another object (such as a vehicle) comes into the scene, it will be tracked as a pedestrian (or a group of pedestrians). The problem is currently handled by performing simple classification based on the location and the direction of motion of the object being tracked. Further work is in progress to classify pedestrians and vehicles.

Acknowledgment

The authors would like to thank the anonymous reviewers for their thoughtful comments.

References

[1] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. New York: Academic, 1988.

[2] A. Baumberg and D. Hogg, "Learning flexible models from image sequences," in *Proc. Eur. Conf. Computer Vision*, vol. 1, Berlin, Germany, May 1994, pp. 229–308.

[3] ——, "An efficient method for contour tracking using active shape models," *Proc. IEEE Workshop Motion of Nonrigid and Articulated Objects*, pp. 194–199, Nov. 1994.

[4] Q. Cai and J. K. Aggarwal, "Tracking human motion using multiple cameras," in *Proc. 13th Int. Conf. Pattern Recognition*, Los Alamitos, CA, Aug. 1996, pp. 68–72.

[5] E. R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, 2nd ed. New York: Academic, 1997.

[6] D. M. Gavrila and L. S. Davis, "3-D model-based tracking of humans in action: A multi-view approach," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 73–80, June 1996.

[7] I. Haritaoglu, D. Harwood, and L. S. Davis, "W^4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 809–830, Aug. 2000.

[8] B. Heisele, U. Kressel, and W. Ritter, "Tracking nonrigid, moving objects based on color cluster flow," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 257–260, June 1997.

[9] B. Heisele and C. Wohler, "Motion-based recognition of pedestrians," in *Proc. 14th Int. Conf. Pattern Recognition*, vol. 2, Brisbane, Qld, Australia, Aug. 1998, pp. 1325–1330.

[10] R. Hosie, S. Venkatesh, and G. West, "Detecting deviations from known paths and speeds in a surveillance situation," in *Proc. 4th Int. Conf. Control, Automation, Robotics, and Vision*, Singapore, Dec. 1996, pp. 3–6.

[11] R. Jain, D. Militzer, and H. H. Nagel, "Separating nonstationary from stationary scene components in a sequence of real-world TV images," in *Proc. Int. Joint Conf. Artificial Intelligence*, Cambridge, MA, Aug. 1977, pp. 612–618.

[12] N. Johnson and D. Hogg, "Learning the distribution of object trajectories for event recognition," *Image Vision Computing*, vol. 14, no. 8, pp. 609–615, Aug. 1996.

[13] I. A. Kakadiaris and D. Metaxas, "3-D human body model acquisition from multiple views," in *Proc. 5th Int. Conf. Computer Vision*, Boston, MA, June 1995, pp. 618–623.

[14] A. Lipton, H. Fujiyoshi, and R. Patil, "Moving target classification and tracking from real-time video," *Proc. IEEE Workshop Application of Computer Vision*, Oct. 1998.

[15] J. R. Parker, "Gray level thresholding in badly illuminated images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, pp. 813–819, Aug. 1991.

[16] K. Rohr, "Toward model-based recognition of human movements in image sequences," *CVGIP: Image Understanding*, vol. 59, pp. 94–115, Jan. 1994.

[17] M. Rossi and A. Bozzoli, "Tracking and counting moving people," *Proc. 2nd IEEE Int. Conf. Image Processing*, pp. 212–216, Nov. 1994.

[18] P. K. Sahoo, S. Soltani, and A. K. C. Wong, "A survey of thresholding techniques," *Computer Vision Graphics Image Processing*, vol. 41, pp. 233–260, 1988.

[19] J. Segen and S. Pingali, "A camera-based system for tracking people in real time," in *Proc. 13th Int. Conf. Pattern Recognition*, Los Alamitos, CA, Aug. 1996, pp. 63–67.

[20] A. Shio and J. Sklansky, "Segmentation of people in motion," *Proc. IEEE Workshop Visual Motion*, pp. 325–332, Oct. 1991.

[21] K. Skiestad and R. Jain, "Illumination independent change detection for real world image sequences," *Computer Vision Graphics Image Processing*, vol. 46, pp. 387–399, 1989.

[22] C. Smith, C. Richards, S. A. Brandt, and N. P. Papanikolopoulos, "Visual tracking for intelligent vehicle-highway systems," *IEEE Trans. Veh. Technol.*, vol. 45, pp. 744–759, Nov. 1996.

[23] A. D. Worral, G. D. Sullivan, and K. D. Baker, "A simple, intuitive camera calibration tool for natural images," in *Proc. 5th British Machine Vision Conf.*, vol. 2, York, U.K., Sept. 1994, pp. 781–790.

[24] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," in *Proc. 2nd Int. Conf. Automatic Face and Gesture Recognition*, Los Alamitos, CA, Oct. 1996, pp. 51–56.

Osama Masoud was born in Riyadh, Saudi Arabia, in 1971. He received the B.S. and M.S. degrees in computer science from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 1992 and 1994, respectively, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, in 2000.

He is currently the Director of Research and Development at Point Cloud Inc., Plymouth, MN, and a Postdoctoral Associate with the Department of Computer Science and Engineering, University of Minnesota. His research interests include computer vision, robotics, transportation applications, and computer graphics.

Dr. Masoud is a recipient of the Research Contribution Award from the University of Minnesota, the Rosemount Instrumentation Award from Rosemount Inc., and the Matt Huber Award for Excellence in Transportation Research.

Nikolaos P. Papanikolopoulos (S'88–M'93–SM'01) was born in Piraeus, Greece, in 1964. He received the Diploma degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1987, the M.S.E.E. degree in electrical engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, in 1988, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1992.

Currently, he is a Professor in the Department of Computer Science and Engineering, University of Minnesota, Minneapolis. He was a McKnight Land-Grant Professor at the University of Minnesota from 1995 to 1997. He has authored or coauthored more than 100 journal and conference papers in the areas of robotics, sensors for transportation applications, control, and computer vision (30 refereed journal papers). His research interests include robotics, sensors for transportation applications, control, and computer vision.

Dr. Papanikolopoulos was a finalist for the Anton Philips Award for Best Student Paper in the 1991 IEEE Robotics and Automation Conference. He was recipient of the Kritski Fellowship in 1986 and 1987 and received the NSF Research Initiation and Early Career Development Awards. He has also received grants from DARPA, Sandia National Laboratories, NSF, USDOT, MN/DOT, Honeywell, and 3M.