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.

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.

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

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

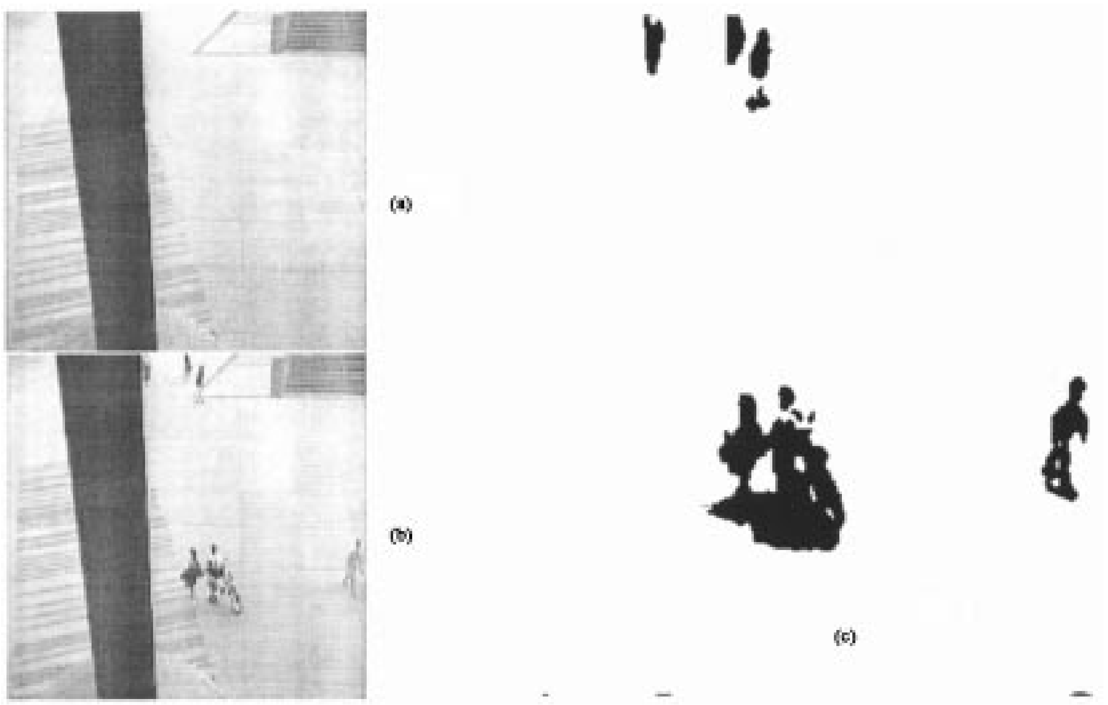


Fig. 1. (a) Background image. (b) Foreground. (c) Difference image showing that a blob does not always correspond to one pedestrian.

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 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

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

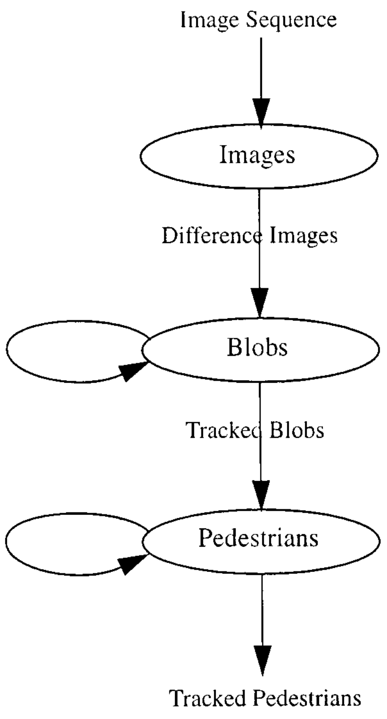


Fig. 2. The three levels of abstraction and data flows among them.

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

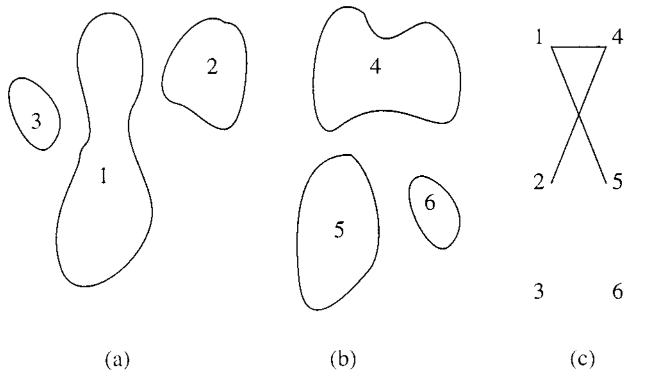


Fig. 3. (a) Blobs in frame $(i-1)$. (b) Blobs in frame i. (c) Relationship among blobs.

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.

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

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com

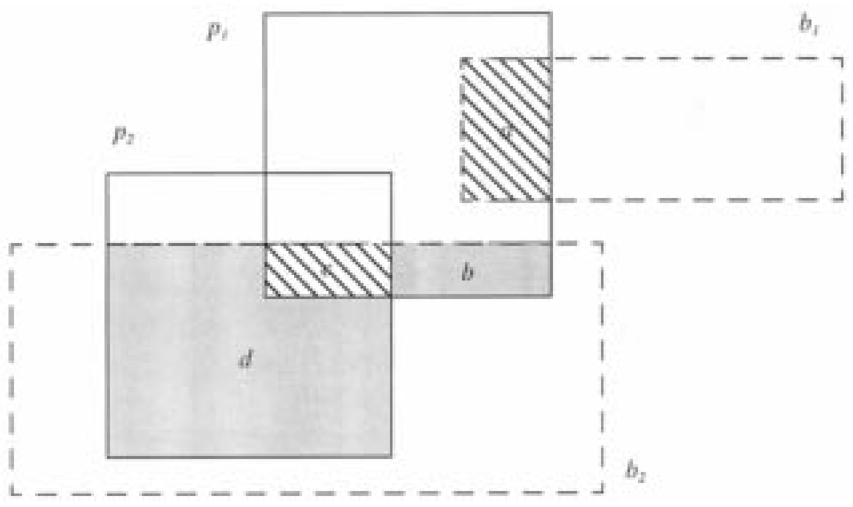


Fig. 4. Overlap area. Pedestrians p_1 and p_2 share blob b_2 while b_1 is only part of p_1 (see Section III-B3 for overlap area computation).

$N_i(u) = \{v \,|\, (u,v) \in E_i\}$. 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 now write the formula for the cost function as

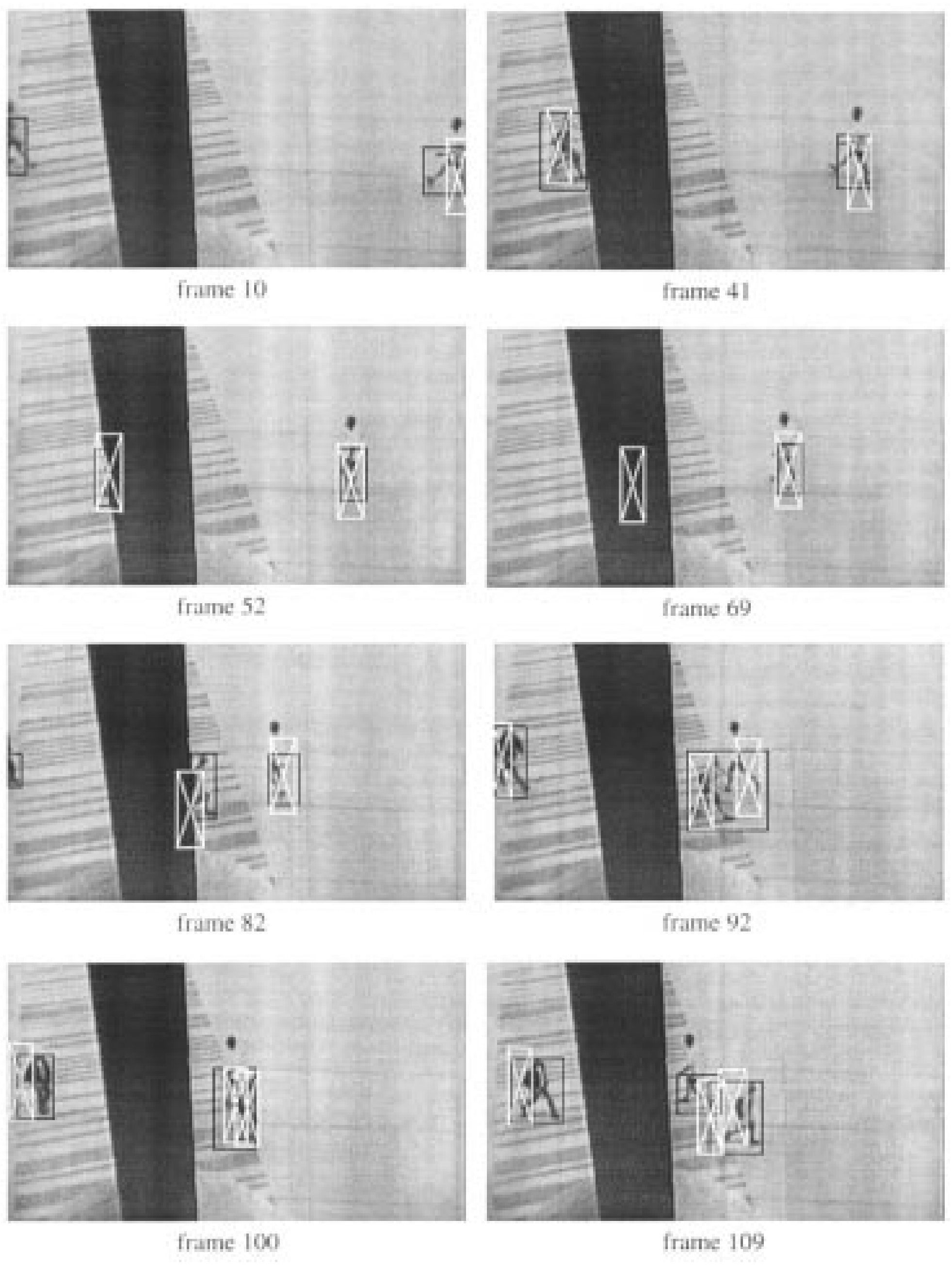
$$C(G_i) = \sum_{u \in P_i} \frac{|A(u) - S_i(u)|}{\max(A(u), S_i(u))}. \tag{1}$$

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,

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com



(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

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

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE