Monitoring Crowded Traffic Scenes

Benjamin MAURIN, Osama MASOUD, Nikolaos PAPANIKOLOPOULOS, *Senior Member, IEEE*

***Abstract*--This paper deals with real-time image processing of crowded outdoor scenes with the objective of creating an effective traffic management system that monitors urban settings (urban intersections, streets after athletic events, etc.). The proposed system can detect, track, and monitor both pedestrians (crowds) and vehicles. We describe the characteristics of the tracker that is based on a new detection method. Initially, we produce a motion estimation map. This map is then segmented and analyzed in order to remove inherent noise and focus on particular regions. Moreover, tracking of these regions is obtained in two steps: fusion and measurement of the current position and velocity, and then estimation of the next position based on a simple model. The instability of tracking is addressed by a multiple-level approach to the problem. The computed data is then analyzed to produce motion statistics. Experimental results from various sites in the Twin Cities area are presented. The final step is to provide this information to an urban traffic management center that monitors crowds and vehicles in the streets.**

***Index Terms*—Tracking, Crowd detection.**

I. Introduction

Monitoring crowded urban scenes is a difficult problem. Despite significant advances in traffic sensors, modern monitoring systems cannot effectively handle busy intersections. This is because there are too many moving objects (vehicles and pedestrians). Tracking humans in outdoor scenes is also very complex. The number of pedestrians and vehicles in a scene varies and is usually unpredictable. Recently, there was progress on techniques to track an arbitrary number of pedestrians simultaneously. However, as the number of pedestrians increases, system performance degrades due to the limited processing power. When the number of pedestrians exceeds a threshold value, the processing rate drops dramatically. This makes it hard to keep accurate track of every pedestrian and vehicle because of the large displacement among pedestrian and vehicle locations in the processed frames.

We propose a vision-based system that can monitor busy urban scenes. Our system is not only limited to vehicles but can deal with pedestrians and crowds. Our approach is robust with respect to a variety of weather conditions and is based on an efficient scheme to compute an approximation of optical flow. The optical flow helps us classify the different objects in the scene while certain statistics for each object are continuously updated as more images become available.

Addressing busy intersections can have a significant impact on several traffic operations in urban areas and on some more general applications. One traffic operation which can be affected is the automated walk signal request. Currently, a pedestrian is required to press a button to request a walk signal at a crosswalk. It may be desirable to automate this process especially when there is a large number of pedestrians waiting to cross the street. Crowd detection can be used in this case. Another application is the study of flow patterns at certain intersections. It may be desirable as a city planning consideration to study the use of crosswalks at a certain intersection. Flow data of pedestrians crossing the street throughout the day can be collected and used to make decisions such as building a pedestrian bridge, etc. Finally, our proposed system may be used to monitor crowds outside schools, nursing homes, train tracks, and at athletic events. One interesting application is to compute crowd density (with obvious public safety applications). In general, monitoring areas where accidents occur often can result into early warning signals and can reduce deadly consequences.

The paper starts with a review of previous work, continues with the detection and tracking schemes, and concludes with experimental results.

A. Previous work

Many methods have been proposed for the detection and tracking of moving objects; many of these have focused on the detection and tracking of traffic objects. The applications of this line of research are multiple: tracking of targets, vision-based robot manipulators, presence detectors, analysis of visual data, etc.

The first generation trackers made several assumptions regarding the nature of the objects in the scene. These assumptions resulted in acceptable tracking performance yet the trackers were not robust to changes in the scene (e.g., movement of the camera or small changes in the shape of the object). Baumberg and Hogg in [1] introduced one of first trackers. It was based on active contours to track the silhouette of a person. Others, such as Rossi and Bozzoli in [2], addressed the problem by locating the camera above the scene, which avoids the creation of occlusions. But this condition is not easy to implement, especially in traffic applications. The systems described require extensive models of the objects to be tracked. They also require some knowledge of their dynamic characteristics. We are going to use only a few

Manuscript received March 15, 2002. This work was supported in part by the Minnesota Department of Transportation and in part by the National Science Foundation through award #CMS-0127893.

B. Maurin is in France.

O. Masoud and N. P. Papanikolopoulos are with Department of Computer Science and Engineering at the University of Minnesota, MN 55455 USA (e-mail: masoud@cs.umn.edu, npapas@cs.umn.edu).

(a) Night video

(b) Day video

Fig. 1. Traffic scenes from the Xcel Energy Center in St Paul, MN.

assumptions on the models in order not to restrict our tracker to a single class of detection. For example, no limits will be assumed for the velocities of the traffic objects, their sizes, their accelerations, or their shapes.

Systems based only on background subtraction give good results but fail to detect and track traffic entities when they get close to each other. One example of such a method can be found in [3]. Our system extends that approach by utilizing motion cues. Cutler and Tuck [4] developed a recognition and classification module for motion which aimed to interpret gestures. Their work is based on the analysis of moving areas in a video sequence. This makes it more of a motion classifier than a tracker. Our detection method is close to [4] and to Dockstader and Tekalp's work [5], which addressed robust motion estimation.

II. Processing of the Image Sequences

A. Detection

We use grayscale image sequences as input. We do not make any assumptions on the nature of the scene (night, day, cloudy, rainy, winter, summer, etc.). Elements of the scene are not known a priori and the sizes of the traffic objects (vehicles and pedestrians) are also unknown. We apply a mask to the image to remove undesired regions such flashing billboards or reflective building walls. In the case of groups of people, it is difficult to decompose a group into individuals. Instead, one would attempt to track the group as one entity. A possible way to do this is to calculate the movement of each point and then group together points with similar motion characteristics. An assumption that we make is that a crowd of pedestrians is a slow deformable object with a principal direction of motion.

We have developed a method to detect and track blobs. Because outdoor environments cannot be modeled accurately, it is difficult to separate the traffic objects from the background. Outdoor images can have very bad quality, especially at night. Moreover, the great difference between day and night image sequences (see Fig. 1) explains why a robust detector needs to be developed. Our method combines both background removal and optical flow segmentation in an effort to increase robustness. Background removal classifies pixels as belonging to the foreground or the background. However, this information is not sufficient in certain cases. In particular, during the motion of two groups passing each other, one cannot estimate the speed or direction. Background removal is useful in the case of easily distinguishable objects with minimal occlusion. For this reason, we use an approximation of optical flow to provide the extra information needed.

1) Background removal

We use a detection scheme that subtracts the image from the background. The idea is based on an adaptive background update. The update equations are as follows:

$$\begin{aligned} A &= \mathrm{AND}\left(M_{\mathrm{bkgnd}}, I_{\mathrm{obj.mask}}\right) \\ B &= \mathrm{AND}\left(\frac{I_{\mathrm{current\ frame}} + 15 \cdot M_{\mathrm{bkgnd}}}{16}, \mathrm{NOT}(I_{\mathrm{obj.mask}})\right) \\ M_{\mathrm{bkgnd}} &= \mathrm{OR}(A, B) \end{aligned} \tag{1}$$

where M_{bkgnd} is the estimated background, $I_{\mathrm{current\ frame}}$ is the current image, and $I_{\mathrm{obj.mask}}$ is a binary mask containing the detected foreground so far. $I_{\mathrm{obj.mask}}$ is in turn computed as

$$\begin{aligned} I_{\mathrm{obj.mask}} &= \mathrm{OR}(I_{subtraction\_mask}, I_{optical\_mask}) \\ I_{subtraction\_mask} &= \begin{cases} 1 & \text{if } \left|I_{\mathrm{current\ frame}} - M_{\mathrm{bkgnd}}\right| < K\sigma \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

where $I_{subtraction\_mask}$ is the binary mask of the foreground computed by background subtraction and $I_{optical\_mask}$ is the foreground computed using optical flow estimation as discussed below. The threshold used is a multiple (K) of the standard deviation σ of camera noise (modeled as white noise). K was decided empirically to be 6.

B. Motion extraction

Motion extraction can be performed accurately using optical flow. However, optical flow estimation is a difficult problem to solve in real-time. Like Sun [6], Dockstader and Tekalp [5], and other researchers, we will concentrate on the speed requirements of the algorithm more than on the accuracy of the estimation of the optical flow. In [7], Galvin *et al.* give a comparison of estimation algorithms for optical flow. Of all the algorithms presented, only two will be considered because of their simplicity and speed. The others, such as fast separation in pyramids, the robust estimation of Black and

Anandan [8], or the wavelet decomposition for fast and accurate estimation by Bernard [9], are too complex to implement in real-time.

A differential method for computing optical flow was explained in Barron *et al.* [10]. It was also used by Rwekamp and Peter [11] with the goal of using ASIC processors for the real-time detection of movement. The idea is to use a minimization procedure where a spatial estimate of the derivative by a Laplacian is employed. One may use a spatial iterative filter to obtain a more precise estimation. We have experimented with this method using a floating point as well as an integer implementation (for efficiency concerns). Based on our experiments, we concluded that although this method works well with synthetic images, it is sensitive to noise which is present in our video sequences. In addition, the computational cost is prohibitive (less than 1 fps in the floating point implementation).

Our method is a correlation method inspired from [4]. The idea is to find the velocity vector of each pixel by performing a local search. Given a pixel $p(x, y)$ in image I_{i-1}, we conduct a minimum cost search in a predefined pattern around (x, y) in I_i. We use the sum of absolute differences (SAD) for the cost function in a predefined neighborhood size. The corresponding point in I_i for point p is therefore estimated as:

$$(\hat{u}, \hat{v}) = \underset{(u,v)\in \mathrm{Pat}(x,y)}{\arg\min} \left(\sum_{(a,b)\in \mathrm{Neighborhood}(u,v)} \left| I_i(a,b) - I_{i-1}(x+a-u, y+b-v) \right| \right) \quad (3)$$

where Pat() defines a search pattern around a pixel and Neighborhood() defines a neighborhood size. The optical flow of p becomes $(\hat{u}, \hat{v}) - (x, y)$. Computing (3) for each pixel can be computationally expensive. In [4], inter-image differencing is used to limit the number of pixels for which optical flow is computed to those pixels located near motion. This makes the algorithm efficiency dependent on the amount of motion in the image. Instead, we utilize the inherent redundancy in (3) and our vision processor board (Matrox Genesis) capabilities to compute (3) for every pixel in a constant time. Our method works as follows: We first compute a set of difference images between I_{i-1} and I_i, where I_i is shifted each time according to Pat(). Furthermore, we augment each pixel in the difference image with the value of this shift. Then we convolve each difference image with a kernel of all 1's, whose size is fixed and is determined by the Neighborhood() function. This results in the sum and hence the value of the cost function. Finally, we find the minimum of all the convolved difference images. The estimated optical flow for every pixel can now be retrieved from the augmented offsets in the resultant image.

The kernel can be of any size. The larger it is, the less the noise there will be. But that also means less sensitivity. A good compromise is a size of 5x5 or 7x7. The pattern can be of any shape, although a circular pattern would make the most sense. However, we use a sparse pattern of vertical and diagonal lines for efficiency reasons as in [4]. In our monitoring application, the cameras used are usually rather far away from the scene and motion does not exceed 2 to 3 pixels per frame. Therefore, a typical pattern is defined as follows:

1		1		1
	1	1	1	
1	1	1	1	1
	1	1	1	
1		1		1

To minimize the effect of noise, we use a threshold $K\sigma$ as before on the cost function (normalized by the kernel size). Costs smaller than the threshold are therefore interpreted as noise rather than motion.

This method gave very good results. The kernel that was used for the summation is 7x7. On 320x240 images with a pattern of ± 3 pixels (one pixel larger than the one shown above), the processing rate is approximately 4 fps. On 160x120 images, it is 10 fps. A pattern of ± 2 pixels resulted in a rate of 15 fps on 160x120 images.

C. *Segmentation*

The segmentation phase aims at extracting contiguous groups of pixels, or *blobs*, that have similar properties. In the case of the foreground obtained from background subtraction, this can be achieved using any connected component extraction algorithm (e.g., border following). In the case of optical flow, we use the floodfill algorithm and take the direction of the flow into consideration while performing segmentation. The basic idea is to cluster together contiguous pixels whose optical flow direction differs by an amount below a threshold. At the end of this stage, we will have computed blobs and their attributes. The computed attributes are the number of pixels (size), the center of mass, the mean velocity, the mean direction, and the bounding rectangle. The latter is computed by computing the eigenvectors of the covariance matrix that describes the distribution of the pixels in the blob. The length and width of the rectangle are set to a multiple of the square roots of the eigenvalues. We found that a factor of 1.5 works best in describing these rectangles.

III. TRACKING

Tracking is an essential component of the system. A diagram of the system is shown in Fig. 2. As explained above, optical flow is estimated by the correlation method and the background is used to find motionless foreground objects. The fusion of both is then segmented into blob entities.

Blobs are used for the layers model. They serve as measurements for higher level entities (called *regions*). A confidence value is associated to each of the possible links (called *relationship*) between two entities. This value depends on the overlap surface, the distance, and speed similarity between the two entities.

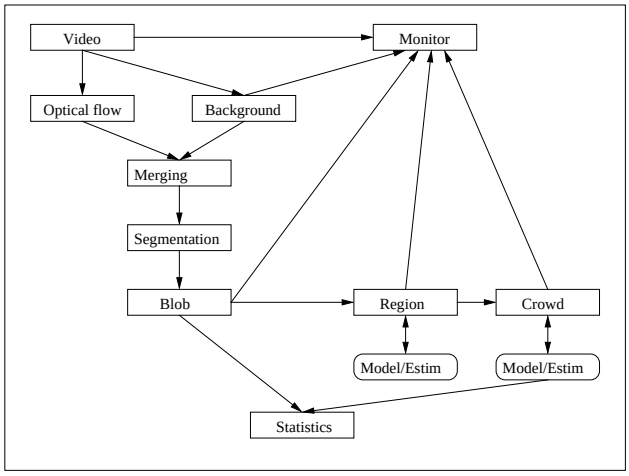


Fig. 2. Tracking system.

A. Layer model

1) First level: blobs

We model a blob using its position, size, bounding rectangle area, and velocity. These parameters are computed during the floodfill algorithm.

2) Second level: regions

The intrinsic parameters are the same as those for blobs. The differences are in the way they are computed. The regions inherit values from their parent blobs at creation time but they are adapted during tracking. Other properties, like the time of existence or inactivity, are also used for regions.

B. Details of relationships

Blob-blob relationships are one-to-one and relate blobs in one frame to blobs in the next frame. Blob-region relationships are many-to-one. All relationships are decided based on the confidence values (α, β) which reflect the similarity between two entities A and B. We introduce thresholds to define the presence or absence of a relationship between two entities. In our tracker, α is associated with the position and the area of the entities while β is related to the similarity in motion direction.

To compute α, we first define Overlap() between two rectangles as the ratio of the area of their intersection to the maximum area of the two rectangles. The intersection of the two rectangles is computed by polygon clipping. We set

$$\alpha(A,B) = \frac{\text{Overlap}(A,B)\cdot 7 + \text{Distance}(A,B)\cdot 3}{10} \quad (4)$$

where Distance(,) is computed with a fuzzy model using the distance between the two centroids and the perimeter of each rectangle:

$$\text{Distance}(A,B) = \max\left(0, 1 - \frac{\|AB\|}{2\cdot \text{Perimeter}(A)}\right). \quad (5)$$

In the case of blob-blob relationships, the above coefficients are set to (5,5) instead of (7,3). This is explained by the fact that a relationship between a region and a blob is supposed to be based more on the overlap than the position.

TABLE I
STATISTICS ON 821 FRAMES DURING 120 SECONDS

Direction (degrees)	Optical Flow	Blobs	Regions
Size (pixels)	2,019,710	1,947,316	2,255,602
Rect. Area (pixels)	N/A	1,136,066	1,433,754
Density(Size/Area)	N/A	1.7	1.57
No Motion	39	N/A	N/A
-135	6 (10)	6	13
-90	6 (10)	1	5
-45	20 (33)	1	8
0	6 (10)	79	44
45	6 (10)	3	8
90	2 (3)	3	5
135	12 (20)	1	13
180	0 (0)	2	0

We also use a fuzzy model to compute β based on the difference in direction, D. This time, we use a threshold, T, below which, differences are considered negligible:

$$\beta(A,B) = \begin{cases} 1 & \text{if } D < T, \\ \max(0, 1 - \dfrac{(D-T)}{90}) & \text{otherwise.} \end{cases} \quad (6)$$

Once α and β are calculated, we consider each blob-blob and blob-region relationship that has a value of α larger than a threshold (10%) to guarantee a minimal connection between entities. Those relationships that pass this test (usually a small fraction of all possible relationships) are assigned a confidence value of $\alpha + \beta$. The blobs' one-to-one relationships are chosen by selecting the relationships with highest confidence values.

1) Region creation

If a blob has been tracked for at least four frames in a row with blob-to-blob $\alpha > 50\%$ in each relationship, a new region is created. The new region inherits the parameters of the last blob.

2) Merging blobs

The relationships between blobs and regions are many-to-one. In other words, several blobs can be used as measurements for a region. Among all the blobs with relationships to a particular region, we create a list of blobs having relationship confidence values ranging between 70% of the maximum value and the maximum value itself. This group of blobs will be used to estimate the location measurement of the region at the next iteration. In other words, this group of blobs is merged into a region. To do this, we compute the centroid, the minimal bounding rectangle, and the mean velocity of the group. Then, we use this information in the regions measurement/estimation part (filtering) described below.

C. Kalman filtering

We use the Kalman filter to track regions. The state vector consists of position and velocity parameters. It also contains other quantities which are assumed constant. They are the covariance matrix entries (initially those corresponding to the covariance matrix that describes the distribution of the pixels

in the parent blob). These will be used to specify the size and orientation of the region's rectangle as was done for blobs (see II.C). The state vector also contains a constant value describing the size of the region (in pixels). Therefore,

$$X = \begin{bmatrix} x & \dot{x} & y & \dot{y} & < x^2 > & < y^2 > & < x, y > & s \end{bmatrix} \quad (7)$$

The measurement vector will have everything except the velocities. It is clear that the system can be split into smaller ones: one for coordinates and velocities, and one for each of the constant parameters.

Measurement noise is related to the blob-region relationship confidence values. In particular, if we let

$$\Sigma = \frac{1}{\alpha + \frac{\beta}{5}} - 1, \quad (8)$$

the measurement noise covariance for the system of coordinates and velocities is

$$\mathbf{R}_n = \begin{bmatrix} \Sigma^2 & 0 \\ 0 & \Sigma^2 \end{bmatrix}, \quad (9)$$

and it is Σ^2 for the systems of constant parameters.

IV. Experimental Results

We tried our algorithms on several outdoor scenes from the Twin Cities area. One area where one may find crowds is around the Xcel Energy Center in St Paul. We filmed scenes during day and night. The processing rate of our program depends on the desired accuracy of the optical flow approach. In the general case, a pattern with a radius of one pixel is enough to detect objects on images with a size of 160x120 pixels. All the results are obtained with a 5x5 Kernel, which allows for very good detection without significant corruption by noise. With these parameters, our tracker runs at 15 fps (this includes output display). If we use a pattern of radius 2, the frame rate drops to 10 fps. One thing worth mentioning is that in case some frames have to be dropped, it is important to drop the same number of frames consistently, e.g., one frame processed followed by two dropped and so on. Otherwise we get temporal instability. The reason is that optical flow estimation assumes that the time interval between two frames is always the same. For this reason, our tracker runs at speeds of either 15 fps or 10 fps, or $30/(n+1)$ fps (n being the number of skipped frames).

A. Statistics

The first result we present shows statistics gathered over a time period of two minutes. These statistics were computed from optical flow, blobs, and regions. The values are shown in Table 1. The statistics were computed by accumulating the flow, region, and blob directions. In the case of optical flow, the table shows the percentage of pixels moving in different directions (the values in parenthesis show the percentage out of moving pixels only). In the case of blobs (and regions), the percentages are also of pixels but the directions are those of blobs (and regions) containing the pixels. The frame rate was 10 fps. Notice that blobs were not able to sufficiently capture the motion in the direction of 135 degrees. This is due to the fact that the blobs direction may be inaccurately initialized when the underlying motion of pixels is not consistent. This also stresses the need for the regions level, which was able to capture that direction based on blob information even though the information is not accurate. Notice also that the most frequent region direction is 0 whereas the most frequent pixel direction is –45. This is merely a quantization issue since pixel directions are quantized while region directions are not.

The density represents the ratio of the number of pixels in a blob (or region) to the area of the bounding rectangle of the blob (or region). This value exceeds one because of inaccuracies of modeling the blob (or region) as a rectangle. However, it is a useful indicator to identify a crowd; a region representing a crowd would have a high density (above 0.75). Other parameters that can be used to identify a crowd are the area (sufficiently large) and the speed (sufficiently small).

B. Most used area

Fig. 3(b) is a representation of most used areas in the scene which is a direct result computed by accumulating optical flow values over a period of 2 minutes.

C. Tracking results

Fig. (3) shows some of the tracking results. The results are presented as snapshots every two seconds. Rectangles represent regions. Regions with a cross represent detected crowds. The numbers shown are the region identifiers. Fig. 3(a) shows the adaptively estimated background. Fig. 3(c-e) are from a day-time sequence and Fig. 3(f-h) are from a night sequence.

V. Conclusions

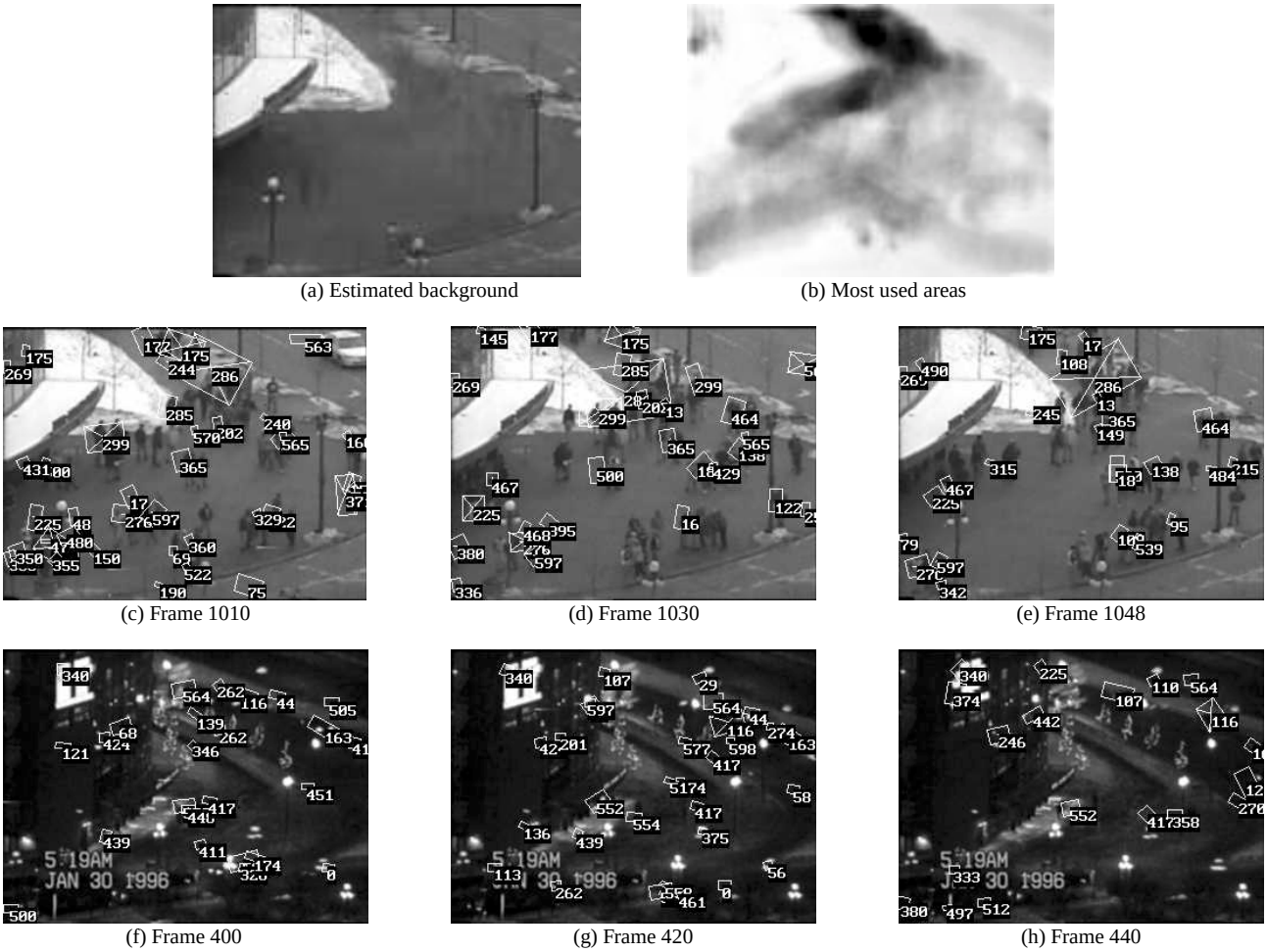
This paper presents a vision-based system for monitoring crowded urban scenes. Our approach combines an effective detection scheme based on optical flow and background removal that can locate vehicles, individual pedestrians, and crowds. The detection phase is followed by the tracking phase that tracks all the detected entities. Traffic objects are not simply tracked but also a wealth of information is gathered about them (position, velocity, acceleration/deceleration, bounding rectangle, and shape features). Potential applications of our methods include intersection control, traffic data collection, and even crowd control after athletic events. Experimental results in different lighting conditions are presented. Future work will be focused on methods to deal with shadows and occlusions.

Acknowledgement

This work was supported in part by the Minnesota Department of Transportation and in part by the National Science Foundation through award #CMS-0127893.

References

[1] A. Baumberg and D. Hogg, 'An Efficient Method for Contour Tracking Using Active Shape Models', in Proc. of IEEE Workshop on Motion of



(a) Estimated background (b) Most used areas

(c) Frame 1010 (d) Frame 1030 (e) Frame 1048

(f) Frame 400 (g) Frame 420 (h) Frame 440

Fig. 3. Tracking results.

Non-rigid and Articulated Objects, pp. 195-199, IEEE Computer Society Press, November 1994.

[2] M. Rossi and A. Bozzoli, 'Tracking and Counting Moving People', Proc. Second IEEE International Conference on Image Processing, pp. 212-216, 1994.

[3] O. Masoud, Ph.D. Thesis on 'Tracking and Analysis of Articulated Motion with an Application to Human Motion', University of Minnesota, Minneapolis, March 2000.

[4] R. Cutler and M. Turk, 'View-based Interpretation of Real-time Optical Flow for Gesture Recognition', Third IEEE International Conference on Automatic Face and Gesture Recognition, Nara, Japan, April 14-16, 1998.

[5] S. L. Dockstader and A. M. Tekalp, 'Tracking multiple objects in the presence of articulated and occluded motion', Proc. Workshop on Human Motion, pp. 88-95, Austin, TX, December 2000.

[6] C. Sun, 'A Fast Stereo Matching Method', Digital Image Computing: Techniques and Applications, pp. 95-100, Massey University, Auckland, New Zealand, December 10-12, 1997.

[7] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills, 'Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms', British Conference on Computer Vision, Computer Science Department University of Otago, New Zealand.

[8] M.J. Black and P. Anandan, 'A Framework for the Robust Estimation of Optical Flow', Proc. Fourth Int. Conf. on Computer Vision (ICCV'93), Berlin, Germany, May 1993.

[9] C. Bernard, Ph.D. Thesis on 'Ill-conditioned Problems: Optical Flow and Irregular Interpolation', document on the web, www.cmap.polytechnique.fr/~bernard/these, 1999.

[10] S.S. Beauchemin and J.L. Barron, 'The Computation of Optical Flow', Dept. of Computer Science, University of Western Ontario, ACM Computer Surveys, vol. 27, no. 3, pp. 433-467, 1995.

[11] T. Rwekamp and L. Peter, 'A Compact Sensor for Visual Motion Detection', VIDERE, vol. 1, no. 2, Article 2, MIT Press, Winter 1998.