

# Real-Time Tracking for Managing Suburban Intersections

Harini Veeraraghavan, Osama Masoud, Nikolaos Papanikolopoulos\*  
{harini, masoud, npapas}@cs.umn.edu  
Artificial Intelligence, Vision and Robotics Lab  
Department of Computer Science and Engineering  
University of Minnesota

*Abstract*— The goal of this project is to develop a passive vision-based sensing system capable of monitoring an intersection by observing the vehicle and pedestrian flow, and predicting situations that might give rise to accidents. A single camera mounted at an arbitrary position looking at an intersection is used. However, for extended applications multiple cameras will be needed. Some of the key elements are camera calibration, motion tracking, vehicle classification, and predicting collisions. In this paper, we focus on motion tracking. Motion segmentation is performed using an adaptive background model that models each pixel as a mixture of Gaussians. The method used is similar to the Stauffer method for motion segmentation. Tracking of objects is performed by computing the overlap between oriented bounding boxes. The oriented boxes are computed by vector quantization of blobs in the scene. The principal angles computed during vector quantization along with other cues of the object are used for classification of detected entities into vehicles and pedestrians.

*Keywords*— Motion segmentation, principal component analysis, tracking, vehicle monitoring.

## 1 Introduction

The goal of this project is to develop a system that can track objects in real-time. The trajectories of the objects over time will be used for predicting the individual trajectories at a later time in order to identify situations giving rise to incidents at intersections.

Motion segmentation is the first step in detecting the objects in a scene. A robust adaptive background segmentation method similar to Stauffer *et al.* [3, 4] is used for segmenting the different regions in the image as foreground and background. This segmentation is very robust to variations in

the scene due to changes in illumination, repetitive motions such as camera noise, moving tree leaves, slow moving objects, etc.

The individual regions are extracted using connected components extraction. The overlap area between bounding boxes in the current and previous frame helps determine whether the two regions are related or not. Oriented bounding boxes provide a tight fit to objects irrespective of their orientation with respect to the image axis. Thus, the oriented boxes are used to model the blobs. They are computed by vector quantization of each blob as explained in the following sections.

The bounding boxes are used for tracking the objects over a sequence of images. The tracking methodology used is similar to Masoud’s pedestrian tracker [2]. The eigenvectors along with the aspect ratio, velocity and size are used for classifying the blobs.

The paper is arranged as follows: Motion segmentation method is discussed in Section 2. Section 3 describes vector quantization method for computing the oriented bounding boxes. The tracking method is discussed in Section 4. Results, future work and conclusions are discussed in Sections 5, 6 and 7.

## 2 Motion Segmentation

Standard adaptive background estimation methods such as time averaging of the frames provide an approximate representation of the scene except for the parts where there has been motion. These methods, however, are not suitable for scenes with a lot of clutter, slow moving objects, etc. due to their poor scene recovery which results in “ghosts” or trails behind a moving object. In Stauffer’s method, each pixel in the image is modeled as a pixel process; each process consists of a mixture of adaptive Gaussian distributions. The distributions with least variance and maximum weight are

isolated as the background. The probability that a pixel of a particular distribution will occur at a time  $t$  is determined by

$$P(X_t) = \sum_{i=1}^k \omega_{i,t} * \eta * (X_t, \mu_{i,t}, \Sigma_{i,t}) \quad (1)$$

where  $\omega_{i,t}$  is the estimate of weight of the  $i^{th}$  Gaussian in the mixture at time  $t$  and  $\mu_{i,t}$  and  $\Sigma_{i,t}$  are its mean and the covariance. Each pixel is compared to the existing distribution until a match is found. When no match is found, the least probable distribution is replaced by the recent distribution. The weights of all the distributions are then updated as

$$\omega_{k,t} = (1 - \alpha) * \omega_{k,t-1} + \alpha * M_{k,t} \quad (2)$$

where  $\alpha$  is the learning rate and  $M_{k,t}$  is 1 for matched distribution, and 0 otherwise. The  $\mu$  and  $\sigma$  remain unchanged for unmatched distributions while they are updated for the matched distribution as

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t \quad (3)$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \quad (4)$$

where  $\rho = \alpha * \eta(X_t | \mu_k, \sigma_k)$  is the learning factor for the current distributions. The Gaussians having the least variance and maximum weight form the background in the scene. In other words, the models with a high  $\omega/\sigma$  value correspond to the background.

## 2.1 Connected Components

The individual regions are extracted using a two-pass connected region extraction method. A raster scan of the image is performed in the first stage coding the image as runs. In the next step, the relation between runs in different rows is established. Various statistics of each blob such as position, centroid, moments, size, elongation, and aspect ratio are also computed.

## 3 Oriented Bounding Box Computation

The oriented bounding box for each blob is computed by using a method of Principal Component Analysis called vector quantization. The covariance matrix for the blobs is given as:

$$\begin{bmatrix} M_{20} & M_{11} \\ M_{11} & M_{02} \end{bmatrix} \quad (5)$$

where  $M_{i,j}$  is the  $i, j^{th}$  order moment. Diagonalising  $M$ , gives

$$M = \Delta^T . D . \Delta \quad (6)$$

where,  $\Delta = [ v_1 \ v_2 ]$  and  $D = \begin{bmatrix} e_1 & 0 \\ 0 & e_2 \end{bmatrix}$ , which are the eigenvectors and eigenvalues respectively.

If  $e_1 > e_2$ , we choose  $v_1$  as the principal axis with elongation  $2.e_1$ . The angle made by the principal axis with respect to the x-axis of the image is also computed. Similarly,  $v_2$  is chosen as the second principal axis with elongation  $2.e_2$ .

## 4 Tracking Methodology

Whole body tracking is used for tracking the objects in the image. Bounding rectangles are used to model the vehicles and pedestrians in a scene. Each bounding box is associated with attributes such as position, area, elongation, velocity, pedestrian/vehicle flag, age, etc. The pedestrian/vehicle flag is just a number which indicates whether a blob is a pedestrian or a vehicle. The blobs are computed from motion segmentation and connected component extraction as explained in the previous section on motion segmentation. In a typical scene, blobs do not always have one-to-one correspondence with pedestrians and vehicles. As a result, a many-to-many relationship is assumed between the blobs and the entities in the image. This relation is updated temporally based on the observed behavior of the blobs.

### 4.1 Blob Tracking

The blob tracking method is similar to Masoud's pedestrian tracker [2]. A set of blobs occurring in a frame  $i$  are related to the blobs occurring in the frame  $i-1$  using an undirected bipartite graph  $G(V_i, E_i)$  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.

In each frame, the association of each new blob in the  $i^{th}$  frame is sought with the blobs in the  $(i-1)^{th}$  frame. Thus, tracking is equivalent to constructing a new association graph for every new frame. With each frame, the blobs are allowed to split, merge, appear, and disappear. To simplify graph computation, the following constraints are imposed:

**Parent Constraint** From one frame to another, a blob may not participate in a split and merge operation at the same time.

**Locality Constraint** Two blobs (or vertices) can be connected only if they have a bounding box overlap area which is at least half the size of the bounding box of the smaller blob.

The graph computation is done in two steps. In the first step, a graph in which no node violates the locality constraint is constructed. There can be only one such graph. In the second step, valid graphs are computed by eliminating the ones that do not satisfy parent structure constraint. There could be several such graphs. Instead of enumerating several valid graphs and then picking up the least cost graph as in Masoud’s method [2], only one valid graph is computed. The graph re-computation is done only for the nodes that do not satisfy the parent structure constraint. For each node the algorithm checks for the above constraint. Violation of constraint occurs when either of the following occurs:

- a given parent node has multiple children and one or more of them has more than one parent.
- a parent has one child and the child has more than one parent and one or more of its other parents is splitting.

In any of the above cases, the algorithm computes the cost of node connection for the given parent-child and the other parent-child nodes whose participation with respect to the given child node caused the violation. The parent-child connection having the least cost is allowed to keep the connection while the connection with the rest is eliminated. The connection that results in least variation in size of the child from that of the parent has the least cost, while the one that results in maximum size variation across frames results in maximum cost. Thus, child nodes that do not have any parent which represents maximum size variation (say, when a blob disappears and reappears again) will have maximum cost. This algorithm is not guaranteed to find the optimal solution all the times. However, for our purposes, the graph computation is fairly accurate. This is thanks to the very high frame rate as a result of which there are very few connections that could possibly cause violation of the constraint or hardly any case with a highly interconnected graph. In such a case, using a single step method though sub-optimal is faster.

## 4.2 Bounding Box Overlap Computation

Bounding box overlap computation is complicated owing to the fact that the boxes are oriented

in arbitrary directions with respect to each other. Hence overlap computation is done by considering the rectangles as polygons. Polygon clipping algorithms can be used for computing the overlap. Instead, we use a simple two-step method. In the first step, overlaps between bounding rectangles formed by corner points of the two boxes are computed. This step is done just to eliminate unrelated blobs to save computation. In the next step, each corner point of one rectangle is considered with respect to the other and the number of points inside the other is found. Using these points, the intersecting points are computed. The same step is repeated considering the corner points of the other rectangle with respect to the first. After eliminating the redundant points, the polygon area is computed from the remaining points after ordering. This corresponds to the overlap area.

## 4.3 Classification

The principal axis angles estimated for computing the oriented bounding boxes along with certain other attributes are used to classify each blob as a vehicle or a pedestrian. The tracked blobs inherit the same classification as the parent blob. This serves as a preliminary classifier and works well in scenes with little or no clutter.

## 5 Results

Our tracking system was tested on images in different weather conditions such as sunny, cloudy, snow etc. The tracking results are shown in Figure 1. The tracker was very successful in tracking the objects even in very cluttered scenes. The algorithm was run on a Pentium II PC. The system had a frame rate of 12-15 fps depending on the clutter in the scene. The system was able to successfully classify in the absence of occlusions and shadows. Tracking is also affected by the presence of shadows in the scene.

## 6 Shortcomings and Future Work

The tracking performance deteriorates due to shadows. Stauffer’s method handles static shadows very well. However, moving shadows cannot be identified. The invariance of texture in a region where a shadow is cast can be used as a cue to identify shadows. We are currently investigating techniques for identifying such regions.



Figure 1: Tracking sequence under different lighting conditions, (cloudy, sunny with clouds and snow.)

Classification of the objects in the image is not accurate in the presence of occlusions, shadows or cases where objects are very close together, such as a group of pedestrians. Future improvements include using techniques like modelling the motion of vehicles and pedestrians [6, 7] in order to produce a better classifier.

## 7 Conclusions

A real-time tracking system for monitoring vehicles and pedestrians under different lighting conditions and in the presence of clutter is presented. Image segmentation is performed by modelling each pixel in the image as a mixture of Gaussian distributions. The system learns the model of the background with time and updates the background with changing lighting conditions.

The system has been used successfully to track vehicles and pedestrians using oriented bounding boxes in outdoor environments. A single camera mounted at an arbitrary position in the scene is used to track the images. The system achieves real-time performance in different lighting and weather conditions.

## 8 Acknowledgement

This work has been supported by the ITS Institute at the University of Minnesota, the Minnesota Department of Transportation, and the National Science Foundation through grant # CMS-0127893.

## References

- [1] C. Smith, C. Richards, S. A. Brandt and N. P. Papanikolopoulos, "Visual Tracking for intelli-

gent vehicle-highway systems", *IEEE Trans. on Vehicular Technology*, vol. 45, no. 4, pp. 744-759, Nov. 1996.

- [2] O. Masoud, "Tracking and analysis of articulated motion with application to human motion", Ph.D. Thesis, Dept. of Computer Science and Engineering, Univeristy of Minnesota, 2000.
- [3] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time Tracking", *Proc. Computer Vision and Pattern Recognition Conf. (CVPR '99)*, June 1999.
- [4] C. Stauffer and W. Eric L. Grimson, "Learning patterns of activity using real-time tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, August 2000.
- [5] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, July 1997, vol. 19, no. 7, pp. 780-785.
- [6] A. Selinger and L. Wilson, "Classifying objects as rigid or non-rigid without correspondences", *DARPA Image Understanding Workshop (IUW)*, Monterey, CA, November 1998, pp. 341-348.
- [7] H. Fujiyoshi and A. Lipton, "Real-time human motion analysis by image skeletonization", *IEEE Workshop on Applications of Computer Vision (WACV)*, Princeton, NJ, October 1998, pp. 15-21.