

# A Model-based Vehicle Segmentation Method for Tracking

Xuefeng Song      Ram Nevatia

*Institute for Robotics and Intelligence System*

*University of Southern California, Los Angeles, CA 90089-0273, USA*

*{xsong|nevatia}@usc.edu*

## Abstract

*Our goal is to detect and track moving vehicles on a road observed from cameras placed on poles or buildings. Inter-vehicle occlusion is significant under these conditions and traditional blob tracking methods will be unable to separate the vehicles in the merged blobs. We use vehicle shape models, in addition to camera calibration and ground plane knowledge, to detect, track and classify moving vehicles in presence of occlusion. We use a 2-stage approach. In the first stage, hypothesis for vehicle types, positions and orientations are formed by a coarse search, which is then refined by a data driven Markov Chain Monte Carlo (DDMCMC) process. We show results and evaluations on some real urban traffic video sequence using three types of vehicle models.*

## 1. Introduction

Detection and tracking of vehicles is important for many computer vision applications such as traffic analysis and video surveillance, since vehicles are an important class of moving objects in our daily life. When the observation is from a stationary video camera, comparison with a learned background model can be used to extract the moving pixels (*foreground*). These blobs, however, do not necessarily correspond to one moving vehicle respectively, even if we assume that vehicles are the only moving objects; multiple vehicles may merge into one blob in presence of inter-occlusion and a vehicle may split into multiple blobs because of occlusion by scene objects (see Figure-1). To detect, track and classify vehicles from motion foreground blobs, even in presence of these difficulties, is the objective of the work described in this paper.

We focus on videos taken by a single calibrated camera, at a height of a few meters above ground as would be common in surveillance application; here occlusion among vehicles and from static objects such as trees may be significant. We expect the resolution to be such that a medium size car is higher than 20



Figure 1: (a) a sample input frame; (b) motion foreground after background subtraction.

pixels in the image. We assume that there are no large shadows or reflections, or that they have been removed from the motion foreground by a pre-processing step such as in [18].

### 1.1 Previous work

Car detection and classification has been attempted from static images, as in [1][2]; such methods show promising results but motion of vehicles offers significant additional cues. Methods described in [3] [4] utilize knowledge of camera calibration and assume that vehicles are on a known ground plane. They project simplified 3D wireframe vehicle models onto the 2D image to match image gradients for vehicle detection and classification; these methods are applied to unoccluded vehicles only. Methods described in [8] and [9] use 2D shape models to track vehicles in aerial videos where inter-vehicle occlusion is not an issue.

To reason about occlusion between vehicles, [5] [11], and [12] take advantage of contour, feature points, and Markov Random Field representation respectively. However, they all require a specific detection zone (or region of interest) where vehicles must be identified separately before occlusion happens. Pang et al. [13] work on foreground image to detect occlusion and separate merged vehicles; however, their method is sensitive to foreground noise, and can't handle change of vehicle orientation.

Many methods exist to track motion blobs without using shape models. In [6] Cohen *et al* extract tracks by representing blobs in continuous frames as a graph; this helps filter out short-term merging or splitting of

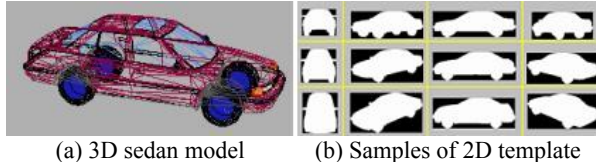


Figure 2: In (b), the three rows are camera tilt angle  $0^\circ$ ,  $15^\circ$ , and  $30^\circ$  respectively; the four columns are vehicle orientation  $0^\circ$ ,  $30^\circ$ ,  $90^\circ$ , and  $120^\circ$  respectively.

blobs; but cannot separate vehicles that are merged over large intervals such as when two inter-occluding vehicles move together. Gupte et al. [7] apply blob tracking for vehicle detection and classification but merging of blobs is a major cause for both detection and classification errors in their system.

## 1.2 Overview of Approach

Our approach is to use generic shape models of classes of vehicles, such as for a sedan or an SUV, to detect, classify and track vehicles from motion foreground blobs. We assume that these blobs are generated by moving vehicles or by noise. Shadows and reflections are assumed to be absent or removed by some pre-processing. Given a binary foreground blob, we formulate the segmentation problem as one of finding the number of vehicles and their associated parameters. We use a generative approach in a Bayesian framework; we form vehicle hypotheses and compute their posterior probabilities by combining a prior distribution on the number of vehicles with measurement of how well the synthesized foreground matches with the observed foreground (*i.e.* the *likelihood*).

Several issues need to be addressed to follow this approach: the nature of models, likelihood and prior distribution functions, and efficient search through the hypothesis space. We summarize our approach below and provide details in the following sections.

We start with 3-D models of a typical member of a class of vehicles. For example, we pick a specific sedan model as a representative of all sedans; this is satisfactory because our aim is not to classify vehicles into specific models and the intra-class differences are not significant for detection. We assume that the camera is calibrated and that it is approximated by an orthographic projection model for observing vehicles whose size is small compared to their distance from the camera. We also assume vehicles move on a known ground plane and that the image x-axis is parallel to this plane (*i.e.* zero “yaw” angle). Thus, the projected shape depends only on the orientation and vertical position in the image of the vehicle. This vertical position can be used to compute a vehicle-centered tilt

angle (*i.e.* the angle between the ground plane and the line from camera center to the vehicle center). For efficient processing, we pre-compute 2D image profiles for a number of different viewpoints; we quantize this space in bins, 72 bins for  $360^\circ$  vehicle orientation range, and 19 bins for  $90^\circ$  tilt angle range, to give a collection of 1,368 2D shape templates (Figure-2 shows a few of them).

We cast the vehicle detection (and classification) problem as being one of finding the set of vehicle types and associated parameters that maximize a posterior probability, *i.e.* compute a MAP solution. It is not practical to conduct a top-down search through the complete space of possible hypotheses to find this solution for an observed foreground. We use a coarse to fine, 2-stage strategy for reducing the search effort as illustrated in figure-3. In the first stage, we take advantage of the observation that all vehicles have a roughly rectangular shape in the motion foreground. We find rectangles in the image which are likely to include a vehicle; the image size of a rectangle is predicted by the image location given the knowledge of the ground plane, vehicle type and camera geometry. Knowledge of static occluding objects is also considered to reduce the occlusion effect from scene objects. The detected rectangles indicate corresponding pre-generated vehicle masks which are then used for a more accurate measurement. In a general case, we search through a set of vehicle orientations to cover all directions; if the direction of motion of the vehicles is constrained, say to be along a known road, we can limit the search for computational efficiency and robustness.

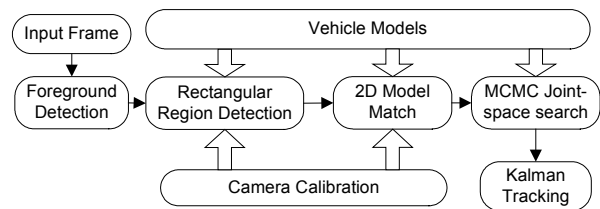


Figure 3: Sketch of our method

A finer search to sample the probability distribution for hypotheses is conducted by a data-driven MCMC (DDMCMC). Such methods have been shown to be effective for human and image segmentation in previous work [14] and [16]. The hypotheses generated in the first stage provide initial proposals. MCMC search includes jump dynamics, where objects may be created, removed, replaced, and diffusion dynamics where variations in parameter values are explored.

We compute posterior probability estimates for vehicles in each frame independently and then use

Kalman filtering to compute associations between the frames. This is not optimal; inter-frame constraints should be incorporated in the search process itself. Nonetheless, even our minimal use of inter-frame information results in large improvements as shown in our results later. Our analysis is largely confined to the use of the silhouettes of the foreground blobs. This is also not sufficient in general as features in the interior of the blobs can be informative. We will also consider these in our future work but believe that our current results are already quite promising and useable for various applications.

The following section gives the formal definition of our approach. And section 3 describes the details about the detection method and the way we apply MCMC. Section 4 presents the tracking method we implemented. The results and evaluation are shown in section 5.

## 2. Bayesian Problem Formulation

Given a binary foreground image  $I$ , assumed to originate from vehicle motion plus noise, we want to answer the following three questions: (1) how many cars are there? (2) Where are they? And (3) what is the type of each car? Formally, we want to find the solution in the space defined by:

$$\Theta = \bigcup_{k=0}^{\infty} (M_1 \times M_2 \times \dots \times M_k), \quad M_i = (t_i, x_i, y_i, o_i) \quad (1)$$

where  $k$  is the number of vehicles and  $M_i$  is a vehicle parameter vector, representing vehicle type ( $t$ ), its center in the image plane ( $x, y$ ), and its orientation in the ground plane ( $o$ ). We can also include the 3D size of vehicles in the parameter vector if the variance of size is not negligible. The solution space is high dimensional and its size (depends on number of vehicles) is not known in advance.

We formulate the problem as computing the maximum a posterior probability  $\theta^*$ , so that

$$\theta^* = \arg \max_{\theta \in \Theta} P(\theta | I) \quad (2)$$

Under Bayes' rule, posterior probability is proportional to the product of prior probability and likelihood,

$$P(\theta | I) \propto P(\theta) \cdot P(I | \theta) \quad (3)$$

We discuss how to calculate these two terms in the following sections.

### 2.1 Prior Probability

Assuming that the occurrence of a vehicle is independence of the others, we define the prior probability of a state is the product of the probability of the number of vehicles and the probabilities of individual vehicles.

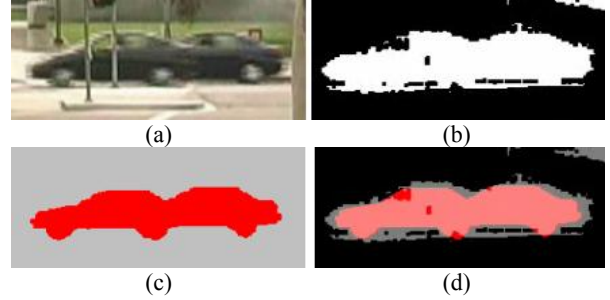


Figure 4: The likelihood based on the match between foreground and synthesized solution image. (a) two connected cars; (b) motion foreground; (c) synthesized image of joint solution; (d) match between foreground and solution.

$$P(\theta) = P(k | N) \cdot \prod_{i=1}^k P(M_i) \quad (4)$$

where  $P(k|N)$  is a Poisson distribution with the sample mean  $N = A_f / A_v$ .  $A_f$  is the overall area of the foreground and  $A_v$  is the average area of one vehicle. The intuition is that more number of vehicles will create a larger area foreground. This term penalizes both too few vehicles and too many vehicles.

$$P(M) = P(t) \cdot P(x, y) \cdot P(o) \quad (5)$$

where  $P(t)$  is the prior probability of the vehicle type. In our implementation, we have three vehicle types, we set  $P(\text{sedan}) = P(\text{SUV}) = P(\text{truck})$ . In general case, we assume the prior probabilities of position  $P(x, y)$  and orientation  $P(o)$  are uniform distributions. However, knowledge of road position and directions, if available, can be added for better performance.

### 2.2 Multi-vehicle Joint Likelihood

When vehicles occlude each other, the image likelihood cannot be decomposed into the product of image likelihood of individual vehicles. We compute the joint likelihood for a given state based on the match between the motion foreground and the synthesized solution image.

We assume that foreground is formed by vehicles and random noise (see figure-4). Denote  $F$  as image foreground,  $V_i$  as the image mask of one vehicle in a solution. Then  $\bigcup_{i=1}^k V_i$  is the ideal foreground of the whole solution. We use the likelihood function described in [14], and given by:

$$P(I | \theta) = \alpha \cdot e^{-(\lambda_1 \cdot E_1 + \lambda_2 \cdot E_2)} \quad (6)$$

where  $E_1 = F - F \cap (\bigcup_{i=1}^k V_i)$  is the set of foreground pixels that are not covered by mask, and  $E_2 = (\bigcup_{i=1}^k V_i) - F \cap (\bigcup_{i=1}^k V_i)$  is the set of mask pixels with no foreground pixels matches,  $\alpha$  is a constant, and  $\lambda_1, \lambda_2$ , are two coefficients to weigh the penalties on the two kinds of errors.

By combining the above likelihood and the prior probability, we get the posterior probability function as:

$$P(\theta | I) \propto P(k | N) \cdot \left( \prod_{i=1}^k P(M_i) \right) \cdot e^{-(\lambda_1 \cdot E_1 + \lambda_2 \cdot E_2)} \quad (7)$$

### 3. Search for MAP

Due to the possibly large parameter space (20 dimensions for 5 cars), it is not possible to conduct an exhaustive search for the MAP solution. A simple greedy search is likely to find only a local maximum. We describe a 2-stage method: in the first stage, coarse hypotheses for presence of vehicles (with associated parameters) are generated; this stage is designed to have high detection rate but also may have many false alarms. These hypotheses provide initial proposals for a second stage DDMCMC method, similar to one described in [14], with jump and diffusion dynamics.

#### 3.1 2-layer Detection for vehicle hypotheses

As mentioned before, we assume ground plane, orthographic projection and zero yaw angle. We describe a vehicle in the image with an 8-parameter vector  $(cx, cy, w, h, rotate, type, orient, tilt)$ .  $(cx, cy)$  is the center of the vehicle at image space.  $(w, h, rotate)$  is rotated bounding rectangle.  $(type)$  is vehicle type.  $(orient)$  is the vehicle orientation relative to camera. And  $(tilt)$  is the angle between ground plane and the line from the camera center to the object center. These eight parameters are not independent. We can derive  $(w, h, rotate, tilt)$  from  $(cx, cy, type, orient)$  by projecting the 3D vehicle model onto the 2D image, with the assumption that camera parameters are known, that the center of the 3D vehicles lies on the projection line from  $(cx, cy)$  and that the variance in size for a vehicle type is not significant. Then  $(cx, cy, type, orient)$  suffice to describe one vehicle. To avoid repeated computation of 3D model projections, we roughly quantize the space  $(cy, type, orient)$ , and learn the corresponding

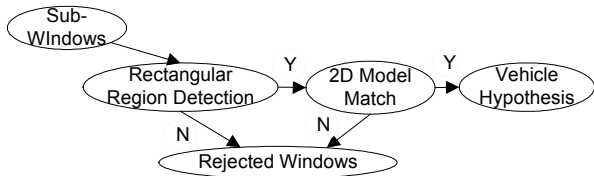


Figure 5: Schematic depiction of 2-layer detection parameters  $(w, h, rotate, tilt)$  by samples; this also accommodates errors in camera calibration. We search in the 4D space  $(cx, cy, type, orient)$  for vehicle hypotheses.

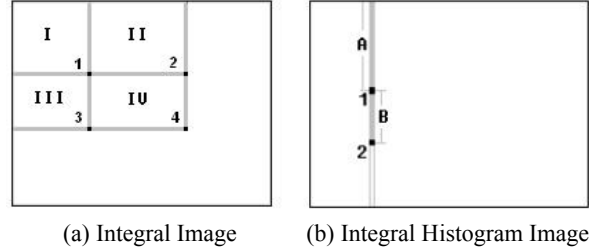


Figure 6: In (b), the value of 1 is sum of pixels on A. The value of 2 is A+B. So, the sum on B is 2-1;

To do this in an efficient way, we adopt a 2-layer detection method. The first layer searches for a rectangular region (not necessarily aligned with the image axes) likely to contain a vehicle, and the second layer uses the projected vehicle models indicated by the detected rectangular regions to match with the foreground for a more accurate evaluation.

#### 3.1.1 Rectangular Region Detection

Given a vehicle  $(cx, cy, type, orient)$ , we retrieve a tilted rectangle  $(cx, cy, w, h, rotate)$ , from a look up table, to approximate the image area of the vehicle. To conduct an efficient match with the image foreground, we use the integral image representation of an intensity image introduced in [10]. The value of a pixel  $(x, y)$  in integral image is the sum of pixel intensities in the rectangle  $(0, 0, x, y)$ . The advantage of this representation is that the intensity sum of any rectangle can be computed with 4 values of the corner pixels in the integral image (see Figure-6(a)). This reduces the computation cost greatly.

As a special case of integral image, we create an integral histogram image to sum the pixels along a row or a column. In the vertical integral histogram image, the value of pixel  $(x, y)$  is the sum of the pixels from  $(x, 0)$  to  $(x, y)$ . With this integral histogram image, the intensity sum of any arbitrary vertical line segment can be computed with two array references (see Figure-6(b)). The horizontal direction works in the same way. It takes only one operation in the integral histogram image to compute the intensity sum on a line segment, while it takes three operations in the normal integral image.

Given a binary foreground image, we compute integral image as  $Int(x, y)$ . Distance map is generated (see figure-7b) from the boundary of foreground image. Vertical and horizontal integral histogram images are computed on distance map, denoted as  $vIh(x, y)$  and  $hIh(x, y)$ . For an arbitrary rectangle  $(x, y, w, h)$ , we calculate the evaluation value  $V$  as below:

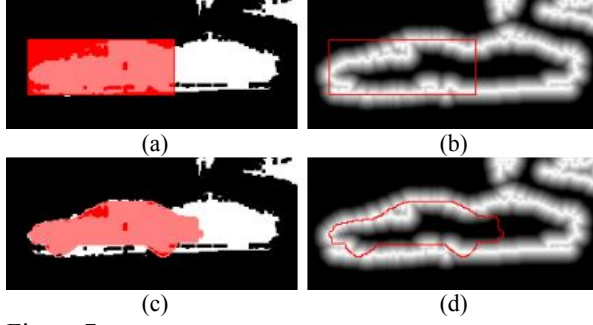


Figure 7: (a) Rectangle region match; (b) Rectangle contour match on edge distance map; (c) 2d mask region match; (d) 2d mask contour match.

$$\begin{cases} V_1 = \text{Int}(x+w, y+h) - \text{Int}(x+w, y) - \text{Int}(x, y+h) + \text{Int}(x, y) \\ V_2 = v\text{lh}(x, y+h) - v\text{lh}(x, y) + v\text{lh}(x+w, y+h) - v\text{lh}(x+w, h) \\ V_3 = h\text{lh}(x+w, y) - h\text{lh}(x, y) + h\text{lh}(x+w, y+h) - h\text{lh}(x, y+h) \end{cases} \quad (8)$$

$$V = V_1 \cdot (V_2 + V_3) / (2 \cdot (w+h) \cdot w \cdot h)$$

As a rectangle is not an accurate approximation for a 2D vehicle appearance, we set a low threshold on  $V$  to achieve a high detection rate.

Figure-8(b) shows the detected rectangles from the foreground of the image in Figure-8(a). In this example, several cars are merged in one blob, but also one car is split into three parts because of the occlusion by scene objects (a tree and a pole). We use knowledge of these occluding scene objects (given by a scene model or by learning) in the following way: when a rectangle covers enough foreground pixels, we treat the inside scene-occluded area (if any) as foreground. In this way, we detect most of the vehicle occluded by scene object with the cost of some more false alarms. In this case, 212 redundant rectangles are found but every vehicle is captured by at least one rectangle.

### 3.1.2 2D Model Match

In the second layer, we use 2D appearance masks to compute a more precise match. From 3D vehicle models, we pre-generate and index the 2D orthographic appearance masks from all viewpoints with quantized parameters (*type, orient, tilt*) as described in section 1.2 earlier. For a detected rectangle, we pick the model with the parameters (*type, tilt, orient*) and match the model with the foreground (see Figure-7(c)(d)) for a more precise evaluation. As for rectangle detection, the evaluation is based on an area match and a contour match. Denote  $V_f$  as the number of foreground pixels matched with model, and  $V_e$  as the intensity sum of pixels on edge energy map along the boundary of the model. We output the detection if  $V_f \cdot V_e$  is larger than a threshold. Finally, we take only the hypotheses with local maximum evaluation values to reduce duplicate

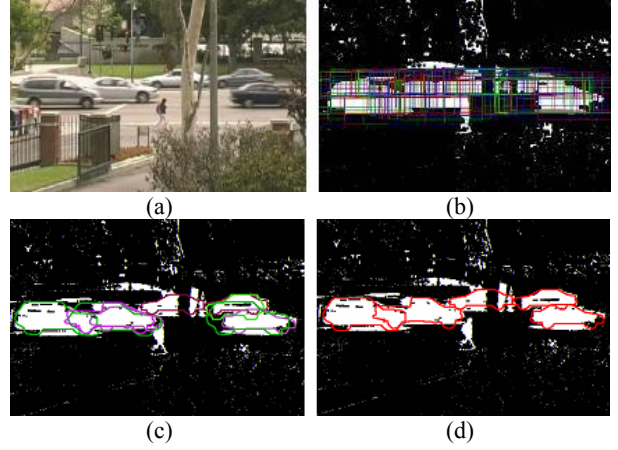


Figure 8: (a) Input image; (b) 217 detected rectangular regions; (c) 9 vehicle proposals after 2d model match; (d) 5 detected vehicles after MCMC search. One sedan is misclassified as a truck, because shadow creates some foreground under the car and part of the car (rear window) is not detected on foreground.

detections (though multiple overlapping hypotheses are still possible). Figure-8(c) shows 9 vehicle hypotheses (for five actual vehicles) after model match.

### 3.2 Markov Chain Monte Carlo

The method described above has high detection rate but also many false alarms. Due to foreground noise and the approximations at the detection level, the parameters of the ‘right’ detection may also not be very accurate. We use the detected hypotheses as initial samples in a data driven MCMC method to search for the joint MAP solution.

As described in [14] [15], the basic idea of MCMC is to design a Markov chain to sample a probability distribution  $P(\theta|I)$ . At each iteration  $t$ , a new state  $\theta'$  is sampled from  $\theta_{t-1}$  based on a proposal distribution  $q(\theta'|\theta_{t-1})$ . The transition intensity is given as

$$p((\theta_{t-1}, k), (\theta', k')) = \begin{cases} \min(1, \frac{P(\theta'|I) \cdot q(\theta_{t-1}|\theta')}{P(\theta_{t-1}|I) \cdot q(\theta'|\theta_{t-1})}), & \text{if } k=k' \\ C \cdot \min(1, \frac{P(I|\theta')}{P(I|\theta_{t-1})}) \cdot p_k \cdot P_k(\theta'), & \text{otherwise} \end{cases} \quad (9)$$

where  $k, k'$  are vehicle numbers in  $\theta_{t-1}, \theta'$  respectively,  $C$  is a normalization factor,  $p_k$  is the prior probability of  $k'$  vehicles, and  $P_k(\theta')$  is the prior probability of  $\theta'$  in the space of  $k'$  vehicles.

Then, the Markov chain accepts the new state with the Metropolis-Hasting algorithm:

$$\theta_t = \begin{cases} \theta', & \text{if } \beta < p(\theta_{t-1}, \theta') \\ \theta_{t-1}, & \text{otherwise} \end{cases} \quad (10)$$

where  $\beta$  is a random number at  $[0, 1)$ .

It has been proven that the Markov chain constructed in this way has a stationary distribution equal to  $P(\theta|D)$ , as long as transition function  $q(\theta'|\theta_{t-1})$  satisfies the requirement of being irreducible and aperiodic.

Denote the state at iteration  $t-1$  as  $\theta_{t-1}=\{k, M_1, M_2, \dots, M_k\}$ . The following Markov chain dynamics apply to  $\theta_{t-1}$  for new state  $\theta'$ . The dynamics correspond to sampling the proposal probability  $q(\theta'|\theta_{t-1})$ .

(1) Vehicle hypothesis addition: Randomly select a proposal  $M=\{t,x,y,o\}$  from vehicle proposals provided by the detection method. A new hypothesis  $M'=\{t,x',y',o'\}$  is generated based on the assigned Gaussian distribution on every parameter (except type).  $\theta'=\{k+1, M_1, M_2, \dots, M_k, M'\}$ .

(2) Vehicle hypothesis removal: Randomly select an existing vehicle hypothesis  $M_i$  in  $\theta_{t-1}$ .

$\theta'=\{k-1, M_1, M_2, \dots, M_{i-1}, M_{i+1}, \dots, M_k\}$ .

(3) Vehicle hypothesis replacement: Randomly select an existing vehicle hypothesis  $M_i$ , and replace  $M_i$  with  $M_i'$ , which is a proposal and has some overlap with  $M_i$ .  $\theta'=\{k, M_1, M_2, \dots, M_{i-1}, M_i', M_{i+1}, \dots, M_k\}$ .

(4) Vehicle type change: Randomly select an existing vehicle hypothesis and randomly change the type of this vehicle. The other parameters are unchanged.

(5) Stochastic vehicle position diffusion: update the position of a vehicle hypothesis in the direction of the gradients under some random noise.

(6) Stochastic vehicle orientation change: update the orientation of a vehicle hypothesis under some random noise.

The first four are referred to as jump dynamics and the rest are referred to as diffusion dynamics. The Markov chain designed in this way is irreducible and aperiodic since all moves are stochastic. Furthermore, redundant dynamics (e.g. *replace* = *remove* + *add*) is added for more efficient traversal in the solution space.

Figure-10 shows the results of the MCMC process on our running examples; these results were obtained after 1000 iterations.

## 4. Tracking of Vehicles

We detect vehicles in each image independently. We predict the parameters of these vehicles in subsequent frames by using a constant acceleration Kalman filter and make associations between detections in consecutive frames. For tracking, each vehicle is represented by a vector  $(id, cx, cy, vx, vy, w, h)$ , representing the id, center position, speed, and size of the bounding rectangle in image. The predicted vehicle descriptions are matched with the new

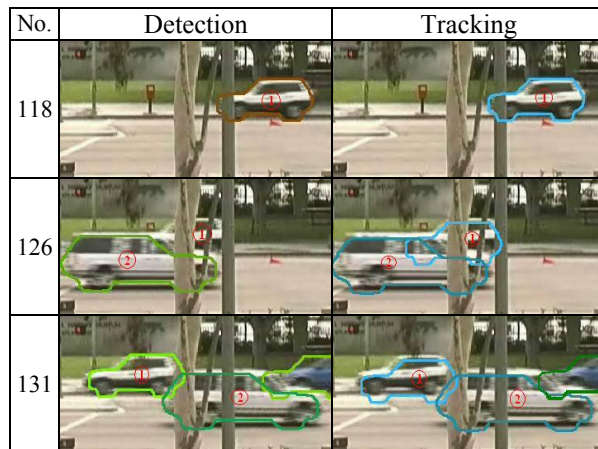


Figure 9: Tracking results. The SUV-1 is missing at frame-126 because of occlusion by Van-2 and scene objects. Tracking can utilize the detections before and after frame-126 to track SUV-1 continuously. Affected by previous frames, the localization of SUV-1 at frame 131 in tracking is a little off.

detections and the position parameters are updated. The match is based on overlap of the bounding rectangles of corresponding vehicles. Unmatched detections are initialized as new vehicles. We also record the continuous time interval for each unmatched vehicle; if it exceeds a threshold value, we remove it from the tracking list. As shown in Figure-9, tracking can effectively recover some missing detections by utilizing spatial and temporal smoothness.

This tracking method is not optimal; it would be preferable to include inter-frame constraints in computing the MAP estimates. We intend to pursue this in our future work but as shown in the following section, this simple tracking described above yields significant improvement in detections and provides reasonable tracking results.

## 5. Experiments and Evaluation

We show results on two different traffic scene videos, (named *cross.avi* and *turn.avi*). For both sequences, the total processing time is about 1 second per frame on a Pentium-IV 2.6GHz computer.

In *cross.avi*, vehicles travel along a road in two known directions. Small shadows, pedestrians, trees, and other noise affect the detected motion foreground. Merging of vehicles in a blob is common: we often see two cars moving together in adjacent lanes in the same direction; cars traveling in opposite directions also merge in the foreground. Up to five vehicles merged together are observed. Merging may persist for several frames or even the whole time that two vehicles are visible. We used 3 common vehicle models (sedan,

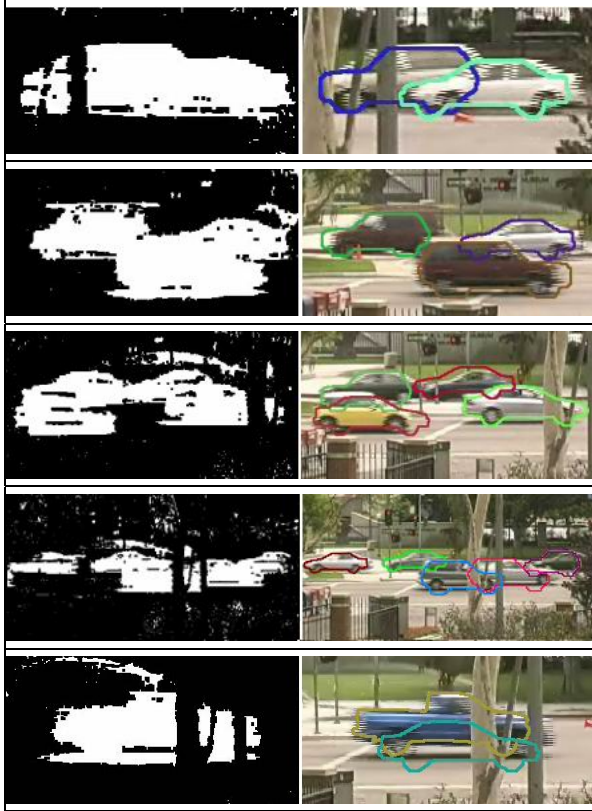


Figure 10: Detection results on side-view vehicle segmentation. The yellow mini van in the third row is detected as a sedan, because we don't have mini van model and sedan matches it best. In the fourth row, the white sedan is classified to SUV because shadow creates a big foreground under it. The right black sedan is classified to SUV because its trunk is invisible. Row 5 is a typical false alarm case. A false alarm sedan is generated to cover the shadow below the blue truck.

SUV, and truck) in our experiments. We used an optical flow method [17] to distinguish the two orientations (moving left or right), and the known orientations in search for vehicle models. Some graphical results are shown in figure-10. Note that all vehicles are detected and the image locations appear to be accurate.

*Turn.avi* is captured at a T-shaped intersection (see Figure-11). Cars make a right turn to merge onto the main street. When the cars are waiting at a stop sign, they could be merged in the detected foreground for 100 frames or more. As the cars coming from right pass by, they also merge with the stopping cars for about 20 frames. To account for turns, we searched vehicle orientation in a 90 degree range.

Some quantitative evaluations are given in Table-1. We evaluate both the detection and the tracking results. In detection evaluation, each frame is processed independently. We define correct detection when a detected vehicle model has an overlap of more

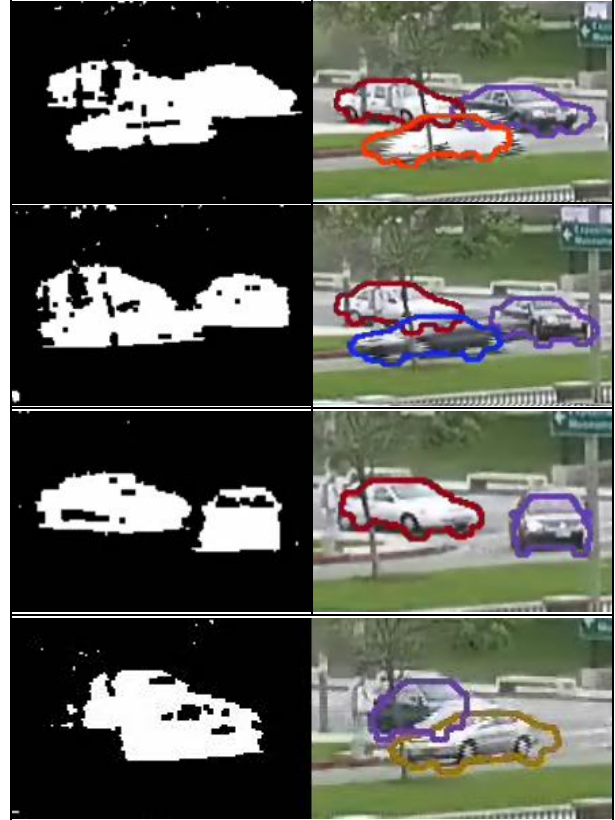


Figure 11: Tracking results on turning vehicles. Through the frames, one black car makes a right turn. The orientation changes are displayed by drawing corresponding contours. The black car in row 4 shows a case of wrong orientation detection caused by occlusion of both the front and the back of the car, leaving little information to distinguish its orientation from the foreground image.

than 80% with the real vehicle. As can be seen, we achieve a high detection rate even occlusion is prevalent (73% vehicle instances are merged with others in *cross.avi*). As we limit our detection on motion foreground, the number of false alarms is small unless many big moving objects appear (e.g. a group of pedestrians). The classification rate is lower, mostly due to error caused by shadows.

To evaluate tracking, we define a track to be "complete" if it covers more than 90% of the frames that the vehicle is in view. Otherwise we call it "incomplete" track. Note one vehicle could create more than one incomplete track. A "false alarm track" is where no actual vehicle corresponds to the tracked vehicle for more than half of its duration. It can be seen that most vehicles are tracked completely, with few fragments and false alarms.

## 6. Summary and future work

Table-1: Evaluation on vehicle segmentation

|                         |                     | cross.avi | turn.avi |
|-------------------------|---------------------|-----------|----------|
| Single Frame Detection  | Frames              | 810       | 543      |
|                         | Vehicle Instances   | 1451      | 980      |
|                         | Merged Vehicles     | 1059      | 304      |
|                         | Detection           | 1346      | 953      |
|                         | Detection Rate      | 92.8%     | 97.2%    |
|                         | False Alarms        | 7         | 20       |
|                         | Classification Rate | 77.1%     | N/A      |
| Video Sequence Tracking | Single Vehicles     | 35        | 5        |
|                         | Complete Tracks     | 31        | 5        |
|                         | Incomplete Tracks   | 6         | 0        |
|                         | False Tracks        | 2         | 1        |

This paper presents a vehicle segmentation approach for solving the vehicle-tracking problem in crowded situations. The approach is composed of a bottom-up detection method and a top-down analysis process. The combination takes advantage of the efficiency of detection method to guide the global optimization process. The experiments demonstrate the effectiveness on several cluttered video sequences. The method does not require manual initialization or for vehicles to appear un-occluded for some time before they can be tracked.

Our current association method is relatively simple. Appearance based association should improve the performance. Combing detection and tracking in multiple frames should improve performance even more. One limitation of our approach is that we have not incorporated explicit reasoning of shadows and reflections. These could be handled in the top-down generative phase. But for efficiency, we would also need to devise lower level segmentation methods; purely photometric methods are effective to some extent but will likely need to be enhanced by some geometric reasoning as well. We intend to explore this in our future work.

## Acknowledgement

This research was funded, in part, by the Advanced Research and Development Activity of the U.S. Government under contract #MDA-904-03-C-1786.

## References

[1] Rajagopalan, P. Burlina, and R. Chellappa. Higher order statistical learning for vehicle detection in images. In Proc. IEEE Intl. Conf. Computer Vision, vol.2, pp. 1204-1209, 1999

[2] H. Schneiderman and T. Kanade. A statistical model for 3d object detection applied to faces and cars. In Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol.1, pp.1746-1751, 2000.

[3] G. D. Sullivan. Model-based vision for traffic scenes using the ground-plane constraint. Phil. Trans. Roy. Soc. (B), vol. 337, pp. 361-370, 1992.

[4] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. International Journal of Computer Vision, 10(3): 257-281, 1993

[5] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In Proc. European Conf. On Computer Vision, pages A: 189-196, 1994

[6] I. Cohen, and G. Medioni, Detecting and Tracking Moving Objects for Video Surveillance. Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol.2, pp. 2319-2326, 1999

[7] S. Gupte, O Masoud, R. Martin, N. P. Papanilolopoulos. Detection and Classification of Vehicles. IEEE Trans. On Intelligent Transportation Systems, vol.3, no.1, 2002

[8] T. Zhao, R. Nevatia. Car Detection in low resolution aerial image. IEEE Intl. Conf. Computer Vision, 2001.

[9] Z. Kim, J. Malik. Fast Vehicle Detection with Probabilistic Feature Grouping and its Application to Vehicle Tracking. In Proc. IEEE Intl. Conf. Computer Vision, 2003

[10] P. Viola, and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In IEEE Conf. on Computer Vision and Pattern Recognition, 2001

[11] S. Kamijo, Y. Matsushita, K Ikeuchi, M. Sakauchi. Occlusion robust tracking utilizing spatio-temporal markov random field model. In Proc. IEEE 15th Intl. Conf. Pattern Recognition, vol. 1, 2000, pp. 140-144

[12] F. Oberti, S. Calcagno, M. Zara, C. Regazzoni. Robust Tracking of Humans and Vehicles in Cluttered Scenes with Occlusion. International Conference on Image Processing, 2002.

[13] C. C. Pang, W. L. Lam, H. C. Yung, A Novel Method for Resolving Vehicle Occlusion in a Monocular Traffic-image Sequence. IEEE Trans. on Intelligent Transportation Systems. vol.5, no.3, Sept. 2004.

[14] Tao Zhao, Ram Nevatia. Bayesian Human Segmentation in Crowded Situations. IEEE Conf. On Computer Vision and Pattern Recognition, 2003.

[15] W.R. Gilk, S. Richardson, D.J. Spiegelhalter, Chapter 13, Markov Chain Monte Carlo in Practice.

[16] Z. W. Tu and S. C. Zhu, Image Segmentation by Data-Driven Markov Chain Monte Carlo, IEEE Trans. On PAMI, vol.24, no.5, 2002

[17] Lucas, B., and Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision, Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI), pp. 674-679, 1981

[18] A. Prati, I. Mikic, M.M. Trivedi, and R. Cucchiara. Detecting Moving Shadows: Algorithms and Evaluation. IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.25, no.7, 2003