A Point-based Tracking Algorithm for Vehicle Trajectories in Complex Environment

Lu Shengnan[1,2], Song Huansheng[1], Cui Hua[1], Wang Guofeng[3]

1. Chang'an University, Xian, Shaanxi, 710064, China
2. Xi'an Shiyou University, Xian, Shaanxi, 710065, China
3. China Highway Engineering Consulting Corporation, Beijing, 100097, China
lushengnan@xsyu.edu.cn

***Abstract*—In this paper, a point-based tracking algorithm is presented, which can be used in traffic jams and complex weather conditions. The main approaches for tracking vehicle trajectories are based on accurately segment for the moving vehicles, while uneven illumination, shadows and vehicle overlapping are difficult to handle. The main contribution of this paper is to propose a point tracking algorithm for vehicle trajectories without a difficult image segmentation procedure. In the proposed algorithm, feature points are extracted using an improved Moravec algorithm. A specially designed template is used to track the feature points through the image sequences. Then trajectories of feature points can be obtained, while unqualified track trajectories are removed using decision rules. The experiment results show that the algorithm is robust enough for vehicle tracking in complex weather conditions.**

Keywords-Vehicle Trajectories; Tracking; Corner Detection;

I. INTRODUCTION

Video cameras have been deployed for a long time for traffic and other monitoring purposes, because they provide a rich information source for human understanding. Video analytics may now provide added value to cameras by automatically extracting relevant information[1]. This way, computer vision and video analytics become increasingly important for intelligent transport systems (ITSs).

Vehicle behavior analysis and incident prediction from video analytics have emerged as an active and challenging research area. The work focuses on a higher level of scene understanding than vehicle detection and tracking of vehicles in images and video. In order to analyze vehicle behaviors and incident prediction, the motion trajectories of vehicles are important foundation. A trajectory is typically defined as a data sequence, consisting of several concatenated state vectors from tracking, meaning an indexed sequence of positions and velocities over a given time window [2]. Using a time window of 1 s, for example, can mean trajectories consisting of 25–30 samples, depending on the frame rate of the camera.

The use of vehicle trajectories to characterize and learn vehicle behaviors has emerged in the past few years. Researches have studied the estimation of vehicle trajectories. Hu *et al.* [3] constructed the 3-D vehicle models in advance with the calibrated image sequences captured by a stationary camera, and then the 3-D trajectories of vehicles formed by tracking the 3-D vehicle models. Buzan *et al*. [4] employed an Extended Kalman Filter(EKF) for tracking the blobs, which were got after segmentation and represent the moving objects. Saunier *et al*.[5] used the robust Kanade-Lucas-Tomasi feature tracker to get the vehicle trajectories. A multiple filter setup was used by Barth and Franke[6], where each three-dimensional object point was assigned an individual Kalman filter and was tracked over time for trajectories. A vehicle was represented by such a point cloud and segmented based on a cuboid model. Yokoyama and Poggio[7] combined edge features and optical flow for object tracking with active contours. In [8], a variational Gaussian mixture modeling was used to classify and predict the long-term trajectories of vehicles. From the above mentioned studies, it is noticed that the main approaches for tracking vehicle trajectories were based on accurately segment for the moving vehicles[9,10]. As usual, image segmentation is a difficult problem in computer vision, especially under various complex traffic environments, such as cloudy, rainy and snowy. The trajectories extraction under various environments will meet many difficulties such as illumination variations, shadow effects and vehicle overlapping problems that appear in traffic jams[11].

In this paper, we make an effort to abandon the idea of tracking objects as a whole but instead selecting the most common low-level features on vehicles for tracking without segmentation. Point features have the advantage of efficient locating and easy tracking. Hence, we suggested a new tracking algorithm for vehicle trajectory extraction using the point features. The significance of the algorithm lies in its temporal processing for vehicle trajectory extraction, and accordingly, it works well for solving many problems of vehicle tracking in traffic jams and complex weather conditions such as in sunny, rainy, cloudy days or at night.

This paper is organized as follows: Section 2 addresses the general principle of proposed algorithm. We introduce the detailed algorithm in Section 3 and Section 4, experimental results and conclusions are given in Section 5 and Section 6.

II. GENERAL PRINCIPLE OF PROPOSED ALGORITHM

The block diagram of the proposed algorithm is depicted in Figure 1. Considering the vehicle appearance to be rigid, corner points on a vehicle surface keep their positions stable over time so that they provide coherent motion information of vehicles. In the proposed algorithm, the moving vehicle candidate is obtained through accumulating frame difference.

978-1-4799-4261-9/14 $31.00 © 2014 IEEE
DOI 10.1109/ISDEA.2014.24

This step is to eliminate unnecessary corners and reduce the calculating amount. Then, corner points of the moving vehicles are detected and qualified corners are selected with the purpose of keeping their positions stable. After selecting corners for the vehicles, tracking is used to measure vehicle paths in video sequence. Considering that vehicle motions can be treated as uniform linear motions between consecutive frames, vehicle corners will be predicted by Kalman filter, which will reduce searching range and calculating amount. Second, a searching area around the predicted corner in the current frame and a matching template area around the corner in the previous frame are established. Then a full search matching is carried out with the searching area and matching template. Finally, we will get the motion trajectory of every corner, while irregular track trajectories are removed.

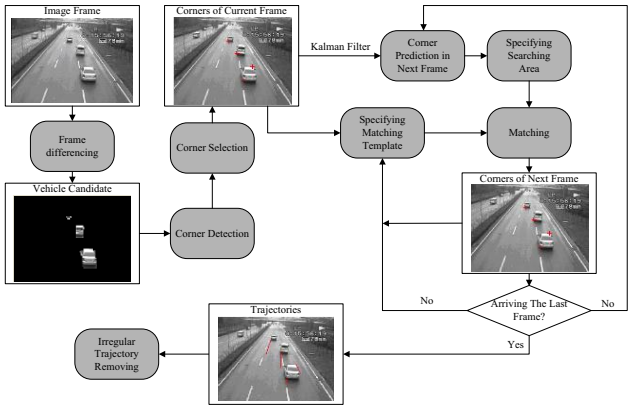


Figure 1. The block diagram of the proposed algorithm.

III. FEATRUE POINTS EXTRACTION

A. Corner Detection

Corner points are usually detected at occluding contours of vehicles (two sides of the vehicles or the junction of vehicle windows and body), however, corner points formed by vehicle and background do not physically exist and are extremely unstable during the relative motion of vehicles and background scenes. We use the Moravec detector for detecting corners in the video frames. The Moravec detector calculates the Sum of Squared Differences (SSD) between a detected pixel and its adjacent pixels on horizontal direction, vertical direction and diagonal direction. We improve the method for reducing noise interference. An image patch defined as 6*8 are measured by taking the SSD between the image patch and a shifted version of itself. Then the smallest value of the eight SSDs will be saved as the point(x,y) eigenvalue. If the point eigenvalue is bigger than a threshold value, this point is signed as a corner. Figure 2 shows the results of corner detection using the above mentioned approach with different scenarios and viewing angles.

B. Corner Selection

We have noticed that several corners are detected on one

Figure 2. Corner detection for different scenarios and viewing angles.

vehicle, which may lead to error tracking and unnecessary calculate amount. Ideally, only one or a small number corners are detected on an moving object. However, the corners distribution is usually intensive in a partial region. To get better corners, which will keep their positions stable, is very critical for the following target tracking. The steps for selecting better corners are listed as follows:

Step 1. All the corners are sorted in descending order according to its eigenvalue. This result will be the scanning order, which is saved as *point[i]*, where *i* values from *1* to *N, N* is the total number of corners. *Point[1]* is the first selected corner, which is saved in *save_point[1]*.

Step 2. Scan the next unchecked *point[i]*, and calculate the distances between *point[i]* and all the selected corners saved in *save_point*, as shown in(1), where *length[j]* is the distance value between *point[i] and save_point[j]*, *m* is the total number of selected corners.

Step 3. The shortest distance, which is denoted as *length*, can be derived by checking the minimum *length[j]* for all *j* in (2), shown below. If the *length* is bigger than threshold *L*, *point[i]* is a selected better corner, which is saved in *save_point[i]*, and *m* increases to *m+1*. If the *length* is smaller than threshold *L*, *point[i]* is delected from corners, which indicates that a better corner has been selected around *point[i]*.

$$length[j] = \sqrt{\begin{matrix}(point[i].x - save\_point[j].x)^2 \\ +(point[i].y - save\_point[j].y)^2\end{matrix}} \qquad j = 1,2,\cdots,\mathrm{m} \tag{1}$$

$$length = \min\{length[j], j = 1,2,\cdots \mathrm{m}\} \tag{2}$$

Step 4. All the corners saved in *point[i]* should repeatedly be checked from step 2 until all corners are evaluated.

Figure 3 shows the proposed approach applied to different cases and viewing angles.

Figure 3. Corner selection for different scenarios and viewing angles.

IV. VEHICLE TRACKING

Vehicle Tracking is to provide correspondences between the regions of consecutive frames based on the corners, which are generated in every video frame. First, vehicle corners are predicted by Kalman filer. Then, a full search patch match method is exployed in a fixed size area based on predicted corners. Finally, we will get the motion trajectory of every corner. These methods are introduced in this section.

A. Corner Predition

The tracking process of Kalman filter is governed by the two important components: a state transition model and an observation model. In order to meet the rough location of corners requirement in the next frame in real time, a state transition model, which models the temporal correlation of state transition between two consecutive frames is only used in this approach.

The state variable *X(k)* is modeled by the corners location parameters *(x,y)* and velocity components (v_x, v_y) at time *k*, where *(x,y)* are corners coordinate paremeters, and (v_x, v_y) are the corners of the horizontal and vertical translation parameters. State transition $A_{k,k-1}$ from time *k-1* to *k* is defined as（3）,where $A_{k,k-1}$ is a transition matrix.

An update model is modeled by corners location in every frame, where *pos[k]* are the corners coordinate parameters value at *k*, and *pos[k].x*, *pos[k].y* are the horizontal and vertical corners location parameters, as shown in (4). If the *k* is equal to *0* or *1*, that is a new corner is detected or a corner is tracked for once time, the velocity of state variable is *0* and corner location parameters are the corner location being detected for the first time. If *k* is more than *1*, the velocity of state variable is the average difference of the corner location parameters on three consecutive frames.

$$X^T(k) = A_{k,k-1} X^T(k-1)$$

$$X(k) = \left[x, y, v_x, v_y\right], \; A_{k,k-1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$X(k) = \begin{cases} [pos[0].x, pos[0].y, 0, 0] & k = 0 or k = 1 \\ [pos[k].x, pos[k].y, \left\lfloor \frac{pos[k].x - pos[k-2].x}{2} \right\rfloor, & \\ \left\lfloor \frac{pos[k].y - pos[k-2].y}{2} \right\rfloor] & k \geq 2 \end{cases} \tag{4}$$

B. Search Area Setting

Searching area is a square centering on a predicted corner in current frame. A matching template, established by the real corner in previous frame as center, moves at each position of the searching area, as shown in Figure 4.

We have noticed that fixed-size searching area and matching template are not suitable for the whole image, because vehicle displacement and relative size are different between close shot and long shot in the image. So, a dynamic strategy is adopted for the searching area and matching template. The Region of Interest(ROI) is divided into four parts with equal length, as shown in Figure 5(a), the different size of searching area and matching template, which is decided by the size of each part, are applied in the four parts. With the size of four parts decreasing, the size of searching area and matching template is proportional to shrink. For example, if the size of searching area is defined as $M \times N$ and the size of matching template as $m \times n$ in part 1, the size of searching area will be $M \times N \times CD/AB$, where *AB* and *CD* are widths of part1 and part2, the size of matching template will be $m \times n \times CD/AB$. The rest parts can be done in the same manner. The results is shown as Figure 5(b).

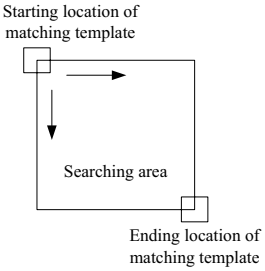
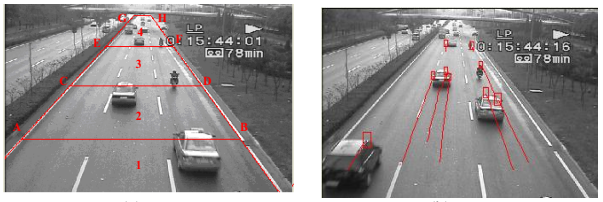


Figure 4. Searching area for matching.



(a) (b)

Figure 5. Searching area and matching template setting. (a) ROI is divided into four parts with equal length. (b)Results of searching for different template.

C. Matching Template Extraction And Update

The matching template is established centering on the corner. In order to reduce calculate amount, the most likely pixels of corners are selected for matching. First, the matching template is divided into 5×5 blocks with the same size, as shown in Figure 6, where block1 lies the center of the matching template, block2 to block 9 are core blocks and block 10 to block 25 are surface blocks. Second, every block eigenvalue is calculated. Next, the matching template is consist of block 1, the three largest eigenvalue blocks in core blocks and two largest eigenvalue blocks in surface blocks. Figure 7 shows the results of matching template found out in accordance with the above methods.

With the movement of vehicles far from camera, the vehicle pixels included in matching template is decreasing, so the structure of matching template need to be updated. We have divided the ROI into four parts. When the tracked corners pass across the boundary of two parts, the matching template is updated. In the above method of matching template extraction, 6 blocks are selected to represente the matching template, which is established in part1. With the detected corner entering into part2, the minimum eigenvalue block in surface blocks is removed, that is there are 5 blocks representing the matching template. Corresponding, the number of blocks will decrease by 1 with the corner acrossing the following parts. In part4, there are 3 blocks in the matching template, block 1 and the two largest eigenvalue blocks in core blocks.

On the other hand, the neighbourhood pixels of the same corner of one vehicle in every frame are changing, so the

matching template need to be constantly updated. The steps for the matching template updating are listed as follows:

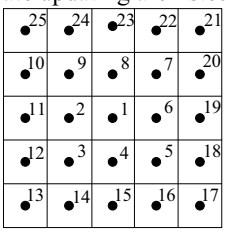
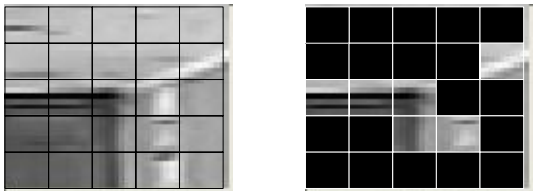


Figure 6. A matching template is divided into 5×5 blocks.



(a) (b)

Figure 7. A proposed matching template.(a)A matching template is divided into 25 blocks. (b)Results of matching template extraction.

Step 1. If a corner is first detected in the *k* frame, its location is saved in *point[0]*, and initialize the matching template centered on the corner. After matching, a new corner is found in the *k +1* frame, which is saved in *point[1]*. Meanwhile, we define two variables *i* and *j*, where *i* is the last time of matching template updating, *j* is the current detected corner. In this step, *i*=0 and *j*=1.

Step 2. If the chessboard distance of *point[i]* and *point[j]*, shown below in (5), is greater than threshold, go to step 3, otherwise go to step 4. *(x,y) and (s,t)* are the coordinates of *point[i]* and *point[j]*, respectively.

$$ChessDis(point[i], point[j])=\max(|x-s|,|y-t|) \quad (5)$$

Step 3. The matching template is updated centered on the corner *point[j]* in the *k* frame, and *i=j,j=j+1*. Searching a new corner in *k+1* frame, which is saved in *point[j]*, then go to step 5.

Step 4. *j=j+1*. Searching a new corner in *k+1* frame, which is saved in *point[j]*.

Step 5. *k=k+1*, and go to step 2.

D. Corner Matching

A function measuring the degree of similarity or dissimilarity between the template and the portion of searching area is taken for matching. The searching rule is based on (6), shown below. In (6), (x_c,y_c) is the best predicting position in the searching area, where *x* is the searching index for the horizontal of searching area, and *y* is the searching index for the vertical of searching area, *f(i+x,j+y)* is the pixel value of the searching area and *g(i,j)* is the pixel value of the matching template. Finally, join the (x_c, y_c) of every frame up to complete motion trajectories.

$$SAD(x,y)=\sum_{i=1}^{m}\sum_{j=1}^{n}\left|f(i+x,j+y)-g(i,j)\right|$$
$$(x_c,y_c)=\arg\min_{x,y}(SAD(x,y)) \quad (6)$$

E. Irregular Trajectories Removing

An ideal track trajectory is a uniform change and smooth curve. However, a few track trajectories are not regular, which may influence the following vehicle behavior analysis. There are two major kinds of irregular track trajectories. One is that corner of background is mistakely selected as vehicle corner because the corner is on the edge of vehicle and vehicle color is similar to background color, as shown in Figure 8(a), pointed out by the arrow. Another is that the location offsets of detected corners change greatly, and the track trajectory is not smooth, as shown in Figure 8(b), pointed out by the arrow. These irregular track trajectories should be deleted before vehicle behavior analysis. So, two testing methods are adopted for irregular track trajectories. For the first case, five consecutive frames are selected to calculate the offset of tracked corners in horizontal and vertical directions, if the offset is smaller than threshold, the track trajectory is removed and stop tracking this corner. For the second case, arc length and chord length of *N* consecutive frames are calculated, shown below in (6), where *N* is the number of consecutive frames, *k* is the total number of tracked corners. (x_k,y_k) is the corner coordinate parameters value at *k*. Then, if the ratio of arc length against chord length is more than threshold, this track trajectory should be removed. Figure 8(c) and (d) show the results of removing irregular track trajectories according to the above two approaches, respectively.

$$Chord=\sqrt{(x_k-x_{k-N})^2+(y_k-y_{k-N})^2} \quad Arc=\sum_{i=k-N+1}^{k}\sqrt{(x_i-x_{i-1})^2+(y_i-y_{i-1})^2} \quad (6)$$

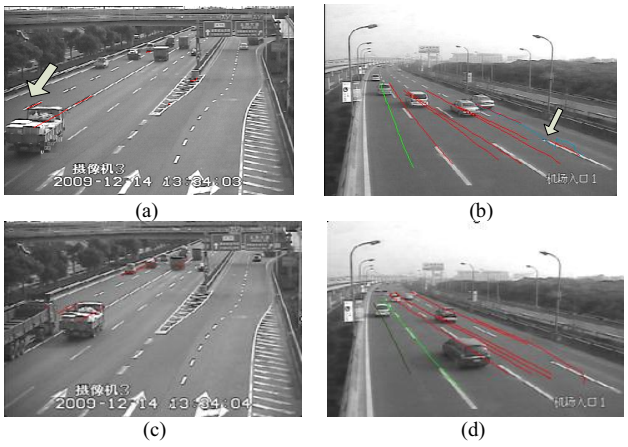
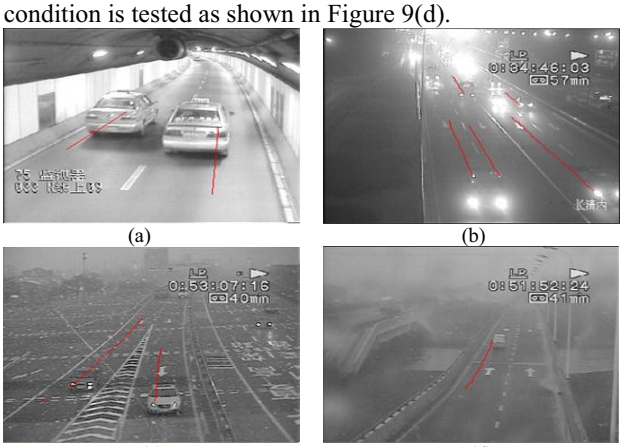


Figure 8. Irregular track trajectories and results of removing them. (a)Irregular track trajectory formed by mistakely selected corners. (b)Irregular track trajectory with not smooth curve. (c)The result of (a) removing the irregular track trajectory. (d) The result of (b) removing the irregular track trajectory.

V. EXPERIMENTAL RESULTS

The performance of the algorithm is assessed on a variety of video sequences, including different traffic and weather conditions. These scenarios are tested on a Windows XP platform with a Pentium 4 3.2-GHz central processing unit (CPU) and 2-GB random access memory (RAM). The size of each image is 720×288, and the

sampling frequency is 25 fps. The proposed algorithm is implemented with Visual C++ on a raw video format. Figure 9 shows the results of vehicle tracking with different conditions. In Figure 9(a) and (b), there are two different illumination conditions: one is in the tunnel with a stable light source, and another is at night. Figure 9(c) shows vehicles moving at high speeds on a snowy day. The misty condition is tested as shown in Figure 9(d).



(a) (b) (c) (d)

Figure 9. Testing scenarios under various conditions. (a)in the tunnel with stable light. (b) Nighttime. (c)Snowy day. (d) Misty day.

We prepared five sequences with different weather conditions, including sunny, nighttime, snowy, misty and stable light in the tunnel. The lengths of the sequences and the results of vehicle motion trajectories with these five sequences are shown in Table I. A true trajectory is a real reflection of the vehicle motion trajectory. Accuracy rate is the result of true trajectories compared with total trajectories in one sequence. The overall results are satisfying, with an average percentage of accuracy rate of 92.5%. Many errors occur in the far distance, where small feature points tracking inaccuracies imply larger error on the world coordinates. Other errors are caused by camera jitter. There are more problems induced by vehicle-light at nighttime.

TABLE I
TRACKING RESULTS FOR EACH SCENARIO

Scenarios	Length(Frames)	True trajectories/ Total trajectories	Accuracy Rate%
Sunny Day	5258	3698/3911	94.6
Night Time	1344	883/956	92.4
Snowy Day	2179	543/595	91.3
Misty Day	3965	1228/1325	92.7
Tunnel	2041	896/976	91.8

VI. CONCLUSIONS

In this paper, a point-based tracking algorithm for vehicle trajectories is proposed. Point features have the advantage of being well localized and easy to track, and they are used for forming vehicle motion trajectories. The proposed algorithm works well in solving problems of vehicle tracking in complex weather conditions like sunny, rainy and cloudy days as well as at night. In order to reduce computation amount and ensure the algorithm performance, an improved Moravec algorithm is used in feature points extraction. Then trajectories can be obtained by tracking the feature points through the image sequences with a specially designed template, and, at the same time unqualified track trajectories are dropped by decision rules. Based on the experimental results, the tracking accuracy is quite good in traffic jams and complex weather conditions. In future work, we will improve the tracking accuracy for the rain or nighttime condition and analyse vehicle behaviors on the basis of trajectories.

REFERENCES

[1] N. Buch, S. A. Velastin and J. Orwell. "A review of computer vision techniques for the analysis of urban traffic," *IEEE Trans. Intell. Transp. Syst*., vol. 12, No. 3, pp. 920-939, Sep. 2011.

[2] S.Sivaraman and M.M.Trivedi. " Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking,and Behavior Analysis". *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1773–1796, Dec. 2013.

[3] W. M. Hu, X. J. Xiao, D. Xie, T. N. Tan, and S. Maybank, " Traffic accident prediction using 3D model based vehicle tracking," *IEEE Trans. Veh. Technol*., vol. 53, no. 3, pp. 677-694, 2004.

[4] D. Buzan, S. Sclaroff, and G. Kollios, "Extraction and clustering of motion trajectories in video," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, Cambridge, U.K., 2004.

[5] N. Saunier and T. Sayed, "A feature-based tracking algorithm for vehicles in intersections," in *Proc. 3rd Can. Conf. Comput(CRV'06)*, Canada,2006.

[6] A. Barth and U. Franke, "Where will the oncoming vehicle be the next second" in *IEEE Intelligent Vehicles Symposium*, 2008.

[7] M. Yokoyama and T. Poggio, "A contour-based moving object detection and tracking," *in 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Oct. 2005, pp. 271–276.

[8] J.Wiest, M. Hoffken, U. Kresel, and K. Dietmayer, "Probabilistic trajectory prediction with Gaussian mixture models," in *Proc. IEEE IV Symp*.,Jun. 2012, pp. 141–146.

[9] C. Hermes, C. Wohler, K. Schenk, and F. Kummert, "Long-term vehicle motion prediction," in *Proc. IEEE Intell. Veh. Symp*., Jun. 2009, pp. 652–657.

[10] T. Gindele, S. Brechtel, and R. Dillmann, "A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments," in *Proc. 13th Int. IEEE Conf. ITSC*, Sep. 2010, pp. 1625–1631.

[11] S. Sivaraman, B. T. Morris, and M. M. Trivedi, "Learning multi-lane trajectories using vehicle-based vision," in *Proc. IEEE Int. Conf. Comput. Vision Workshop*, 2011, pp. 2070–2076.