Realtime Obstacle Detection and Tracking Based on Constrained Delaunay Triangulation

Zu Kim, *Member, IEEE*

***Abstract*— We introduce a realtime vehicle detection and tracking algorithm for in-vehicle video images. Although various vehicle detection approaches have been proposed, it is difficult to find a fast and reliable algorithm for realtime applications, such as for vehicle collision warning. We introduce a realtime appearance-based vehicle detection approach. It uses constrained Delaunay triangulation to make image evidence collection for multiple overlapping hypotheses computationally efficient. We also propose an integrated detection and tracking approach where redundant detection and tracking can compensate each other's shortcomings. The experiment was performed on various video clips of highways and local roads with various traffic and illumination conditions, and the resulting performance is competitive compared to those of other active sensors.**

I. Introduction

In the past decade, there has been a significant improvement in the field of object detection and recognition. However, most of object detection algorithms use computationally expensive feature sets and algorithms such as Lowe's scale invariant features [1]. Although the computing power has been improved significantly, such approaches still require too much computation for realtime applications.

Realtime vehicle detection is an important research area and has useful applications such as for a vehicle collision warning system. A collision warning system informs the driver on obstacles on the road and warns a possible collision. Vision-based detection and tracking algorithm can be very helpful to improve the collision warning performance. For example, many LIDAR sensors have long latencies of as much as 400ms or even more due to mechanical scanning and data pre-processing. It limits the system's ability to quickly respond on leading vehicle's sudden movement. On the other hand, computer vision systems can provide much less latency when the processing time is small enough. In addition, computer vision algorithms can have much higher lateral positioning and tracking accuracy.

Vision-based vehicle detection can also help improve the collision warning performance when it is combined with a lane detection algorithm. In collision warning applications, it is important to know whether the obstacle is in the same lane with the ego-vehicle or not. While a vision-based lane detection system can show a good performance, the calibration of camera and active sensors can be problematic. For example, most vision algorithms estimate the distance to a curve by assuming flat ground which is often wrong due to hills and vehicle's fluctuation. Such an error can be very large when the target is further than 30 meters away [2]. If we use vision-based obstacle detection (with a sensor fusion), the vision-based lane detection can be more useful because it is easier to tell whether an object is in the same lane or not. In addition, it can help improving the lane detection performance by eliminating localization errors caused by vehicle's texture.

Z. Kim is with the California PATH, University of California, Berkeley, CA 90732, USA (http://path.berkeley.edu/∼zuwhan).

Based on such needs, we introduce a realtime vehicle detection and tracking approach. The detection is based on texture analysis and we use constrained Delaunay triangulation for fast image feature gathering. We also introduce an integrated object detection and tracking framework where detection and tracking results can compensate each others' shortcomings. We first review related work in Section II and present our approach. The detection is presented in Section III with experimental results. The tracking algorithm is presented in Section IV, and the conclusion will be in Section V.

II. Related Work and Proposed Approach

There have been various efforts to detect vehicles in realtime or non-realtime [3]. Many of the generic obstacle detection work also deal with images of vehicles [4]. While such algorithms can give good detection rate even for images with complicated backgrounds, most of them are not even close to realtime.

Most of the previous realtime vehicle detection efforts are based on motion or stereo analysis. The motion-based approaches use the fact that the image motion of an obstacle (roughly on a vertical plane) is different from that of the ground (on a horizontal plane) [5]. However, such a difference is usually much smaller than a pixel and it takes at least a second of tracking to distinguish the differences between them [5]. Various stereo algorithms have been also proposed. However, the resolution of stereo depth is significantly limited. This is problematic especially with the collision warning applications which require to detect obstacles of usually 30 $\sim$ 100 meter away. Increasing stereo depth accuracy introduces another problem of recovering camera parameters which may change due to vehicle's vibration.

With an increased computing power, realtime appearance-based vehicle detection became feasible. In [6], Sun et al. applied a set of Gabor filters to vehicle hypotheses classification. While it shows a good classification performance in realtime (10Hz in an embedded computer), the overall performance is heavily dependent on its histogram-based hypotheses generation algorithm, which may not generate good hypotheses in an urban environment with many distracting features.

The computational bottleneck of the generic object detection approach is on collecting features. While a large number of features are required to achieve a good classification performance, the computational cost to gather them is far beyond the realtime. One popular approach proposed by Viola and Jones [7] combines a cascade classifier with a large number of simple (binary) convolution masks for realtime object detection. It shows a good performance for face detection where the target objects have a common shape, but has not been applied for a vehicle detection task.

Our approach is a complementary one to [7] because it allows to use somewhat complicated texture features. Since vehicles' colors and shapes largely vary, texture features, such as the numbers of lines and corners in the region can be more effective in classifying vehicles. To apply such features, image segmentation is a crucial step to avoid computational redundancy. However, image segmentation is a difficult problem which requires a significant amount of computation. In this paper, we propose to apply constrained Delaunay triangulation (CDT) for realtime image segmentation.

The target image is first segmented by applying CDT and object hypotheses are generated by grouping them. We adopt the "hypothesize and verify" paradigm to first generate many overlapping hypotheses then validating them. However, the computation to gather the image statistics for classification is still manageable because the statistics are not gathered directly from the pixels but from larger image chunks (triangles).

In addition to the vehicle detection, we present a probabilistic framework to combine vehicle detection and tracking. In this approach, both the tracking and the detection algorithms help improve the performance of each other. For example, the tracking helps eliminate false detections while redundant detections over frames compensate tracking failures such as drifting.

III. Realtime Vehicle Detection

We use simple texture features for the vehicle classification: density of corner features, density of horizontal line features, density of vertical line features, and intensity variance of the region. In fact, these four features alone provide good classification as will be shown later in this section. Applying more complicated features [1] or a feature selection technique [6] may improve the final classification result, which is not the focus of this paper.

Even though we use simple feature sets, collecting statistics for many (as many as a hundred) overlapping hypotheses is computationally expensive. Therefore, we segment images into small pieces (also referred as *superpixels*) to efficiently gather such information. In the following subsections, we present fast image segmentation based on constrained Delaunay triangulation, our hypotheses generation procedure, and the hypotheses verification (classification) algorithm. We also present detection results.

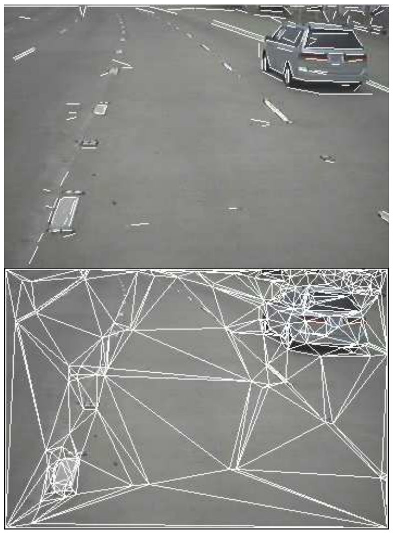

Fig. 1. An example line detection and constrained Delaunay triangulation result.

A. Image Segmentation

Most image segmentation algorithms require too much computation for realtime processing or fast algorithms suffer from leaking that makes grouping difficult. Since our interest is not on finding an accurate segmentation of objects, but on finding small superpixels to reduce the computation, we do not need to use such complicated segmentation algorithms. Instead, we use a constrained Delaunay triangulation (CDT) which was recently recognized as a useful preprocessing step for image segmentation [8] [9].

In CDT-based segmentation, image segments are obtained by finding line segments and applying CDT to find a proper triangulation. We use a Canny edge detector [10] followed by line grouping and fitting to obtain line segments. For CDT it is important to make sure that no resulting line segments intersect each other, and all the curves should be properly broken into straight line segments that the fitting error is small enough. We adopted a line detection approach proposed by Guru et al. [11] to break curves then performed another round of line segmentation to ensure all the curves are properly broken. The resulting line segments (longer than 5 pixels) and the triangulation results are shown in Figure 1.

B. Hypotheses Generation

A vehicle hypothesis is generated from two horizontal line segments. First, horizontal lines are selected among triangle edges. Given a horizontal line (a candidate for vehicle's baseline), the vehicle hypothesis' rough 3-D position is estimated by applying camera calibration parameters. Then, we can estimate the vehicle's rough size in the image (given a fixed vehicle size in the world coordinates). The next step is to perform a search for possible top lines (based on the estimated vehicle size in the image). The top line should be roughly parallel to vehicle's baseline. The top line candidates for an example baseline candidate are shown in Figure 2a.

Given two lines, the triangles between them are collected.

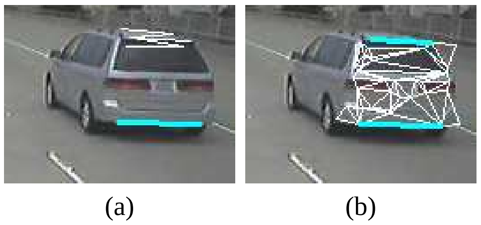

Fig. 2. (a) The top line candidates of an example baseline candidate. (b) A triangle grouping result given two horizontal line segments.

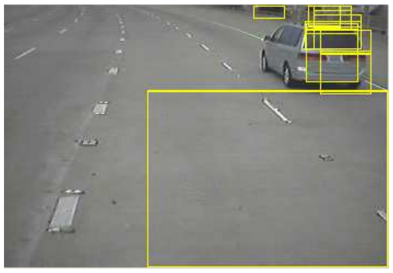

Fig. 3. Total 25 hypotheses were generated by the triangle grouping procedure.

Starting from the triangle right above the baseline candidate, an exhaustive search is performed to find all the triangles which has at least one vertex inside the bounding box of the two lines (the *member triangles*). An example grouping result is shown in Figure 2b. In Figure 3 the bounding boxes of all 25 hypotheses generated from the example image are shown.

C. Hypotheses Verification

Four texture features are used to verify a hypothesis.

- A **corner density** is obtained by averaging the eigen-energy [12] over all the pixels in the hypothesis region.
- A **horizontal line density** is obtained by averaging $\max(|\partial I/\partial x| - \alpha|\partial I/\partial y|, 0)$ over all the pixels in the region, where α is a constant and a larger value of α will give more emphasize on strictly horizontal lines and disregard angled lines. In our implementation $\alpha = 3$. $\partial I/\partial x$ and $\partial I/\partial y$ were obtained by applying 3×3 Sobel masks.
- A **vertical line density** is obtained by averaging $\max(|\partial I/\partial y| - \beta|\partial I/\partial x|, 0)$ over all the pixels in the region, where $\beta = 3$ in our implementation.
- A **standard deviation** of the region is also used.

Although all of the above features are computationally cheap to obtain, repeatedly gathering such information for a large number of overlapping hypotheses is computationally expensive for realtime processing. Therefore, we first pre-calculate required statistics for each triangle, then obtain the feature values of a hypothesis by summing up the statistics of its member triangles.

Since the first three are the average values, the overall average of the region is obtained by weighted summing the triangles' averages with respect to their areas. For the standard deviation, the sum of the intensity values and the square sum of them are kept for each triangle. Then, the standard deviation of a hypothesis is given by

$$\sqrt{\frac{\sum_t S_2(t)}{A} - (\frac{\sum_t S(t)}{A})^2}, \tag{1}$$

where t is a member triangle, $S(t)$ is the sum of the intensity values of the triangle t, $S_2(t)$ is the square sum, and A is the area of the hypotheses (the sum of the areas of the member triangles).

For classification, we gathered 708 learning data where 103 of them were positive. We tested a neural network (with two hidden nodes), a nave Bayesian classifier (with 7 level discretization), and a support vector machine (SVM) with a polynomial kernel. We followed the evaluation scheme presented in [13]. We repeated stratified 5 fold cross-validation for 10 times to obtain the ROC curves with the confidence intervals shown in Figure 4. For all the classifiers, we obtained the ROC curves by changing only the threshold values (no re-learning with different parameters). We see that the SVM shows slightly better performance than other classifiers.

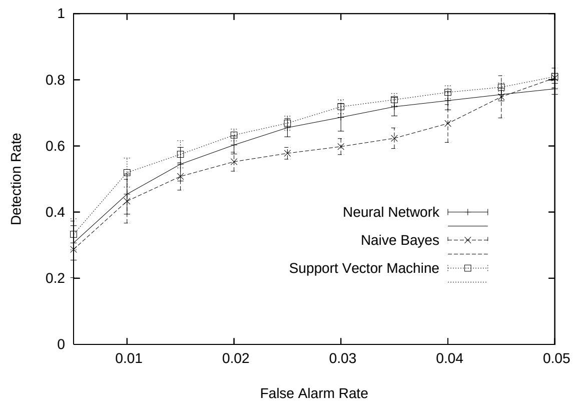

Fig. 4. The classification performance of a neural network, a nave Bayesian classifier, and a support vector machine (SVM). We chose the SVM for the classification.

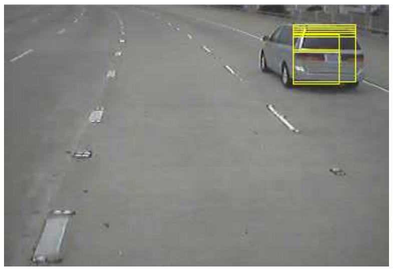

Fig. 5. The *selected hypotheses* obtained by applying the support vector machine classifier. Among the overlapping hypotheses the largest one is finally chosen.

Each and every hypothesis is evaluated based on the SVM's output (the *hypothesis score*) and hypotheses with high scores are selected (the *selected hypotheses*). The selected hypotheses of the example image are shown in Figure 5. We see that there are many overlapping hypotheses. Among them we choose the largest one because our feature sets cannot distinguish a large hypothesis covering a whole vehicle from a small hypothesis which is a part of it. However, it can effectively filter out over-sized hypotheses because the background (road) does not usually have enough texture, which decreases the overall texture density.

D. Detection Results

It is difficult to show an objective detection performance in numbers because the results will be very much dependent on how to choose the examples. However, we can get a rough idea by examining the classification performance shown in Figure 4. The result is surprisingly good considering that we only used simple features. In addition, the learning data contains more "challenging" cases (high scoring false hypotheses) than what usual images give because we added such hypotheses on purpose to induce sharper classification performance.

In fact, the classification result of Figure 4 does not count the mis-detections in the hypotheses generation procedure. When the bottom or top lines are not detected (and are not generated by CDT either), the vehicle hypothesis is not generated. However, our approach generates many more hypotheses than most of other approaches and the loss in the hypotheses generation procedure is small.

Example detection results are shown in Figure 6. Both good and bad detection examples are shown in the figure. The detection was performed on various video clips with different traffic and illumination conditions and image resolutions, but the same parameters were applied to all. We see that our simple feature sets work very well even in a very complex scenes. However, it still has limitations as a realtime algorithm, and mis-detections and false detections occur as shown in the last row of the figure. In the next section, we present a tracking framework that can effectively compensate such limitations.

IV. Integration of Detection and Tracking

Detected targets are tracked based on a template matching algorithm. The target image is resized to the template image size (15 pixels/meter in our implementation), and a cross-correlation matching is applied on a search window (11×7 pixels in our implementation). However, the correlation matching alone cannot effectively deal with tracking failures such as drifting and tracking failures caused by vehicle's orientation changes or calibration errors from camera's pitch angle changes due to vehicle's movement.

To deal with such problems and to compensate detection errors, we introduce a probabilistic approach to integrate detection and tracking. We use the temporal redundancy of the video to suppress the false detections and use redundant detection (in every frames) to compensate the tracking errors.

A. Probabilistic Reasoning

The probability of a target hypothesis x being valid given a set of evidence $\mathbf{E_c}$, (evidence from the current frame) $\mathbf{E_t}$ (*transitional* evidence), and $\mathbf{E_p}$ (evidence from the past frames) is given by the following equation (applying Bayes' rule followed by marginalization):

$$\begin{aligned}P(x|\mathbf{E}_c,\mathbf{E}_t,\mathbf{E}_p) &= \alpha P(\mathbf{E}_c|x)P(x|\mathbf{E}_t,\mathbf{E}_p)\\ &= \alpha P(\mathbf{E}_c|x)[P(x|h,\mathbf{E}_t)P(h|\mathbf{E}_p)+P(x|\bar{h},\mathbf{E}_t)P(\bar{h}|\mathbf{E}_p)],\end{aligned} \quad (2)$$

where α is the normalizing constant and h is the target hypothesis in the previous frame. Two assumptions are used here. First, $\mathbf{E}_c$, $\mathbf{E}_t$ and $\mathbf{E}_p$ are independent to each other. Second, current frame's hypothesis is independent to the history given the previous frame's hypothesis (the Markov assumption).

We also get from the Bayes' rule,

$$P(x|h,\mathbf{E}_p) = \beta P(\mathbf{E}_p|x,h)P(x|h), \quad (3)$$

where $\beta = 1/[P(\mathbf{E}_p|x,h)P(x|h)+P(\mathbf{E}_p|\bar{x},h)P(\bar{x}|h)]$ (the normalizing constant).

We use two evidence features to estimate $P(\mathbf{E}_c|x)$: the hypothesis score (E_{cs}, the output of the SVM classifier) and the re-detection (E_{cr}). The hypothesis score is obtained in the same way as in the detection procedure except that we do not use triangles but directly collect the evidence in the bounding box. It does not take much computation since there are not many targets being tracked. We also examine whether the target is re-detected in the current frame or not. Many false hypotheses are generated because of an accidental alignment of triangles. In such a case, they are not usually re-detected. We assume that the two evidence features, E_{cs} and E_{cr}, are independent to each other: $P(\mathbf{E}_c|x) = P(E_{cs}|x)P(E_{cr}|x)$. The distribution for $P(E_{cs}|x)$ and $P(E_{cs}|\bar{x})$ can be estimated from the learning dataset used in the detection. In our implementation, we discretized E_{cs} into 7 levels. $P(e_{cr}|x)$ is the overall detection rate and $P(e_{cr}|\bar{x})$ is the false alarm rate.

For $\mathbf{E}_p$, we use a single feature: the template matching score (the square of the cross correlation). To get the distribution of $P(E_p|x,h)$, we collected template matching scores of some valid tracking results. We assumed that $P(E_p|\bar{x},h) = P(E_p|x,\bar{h}) = P(E_p|\bar{x},\bar{h})$ and estimate their distributions by collecting the template matching scores of the false matches in the same tracks as for $P(E_p|x,h)$. In our implementation we discretized E_p into 7 levels. The prior, $P(x|h)$, were given manually, and $P(\bar{h}|\mathbf{E}_p)$ is the posterior from the previous frame.

For a newly detected target only a single evidence feature, E_{cs}, is available. Therefore, we applied a simple Bayes rule to estimate the initial posterior probability: $P(x|E_{cs}) = \alpha P(E_{cs}|x)P(x)$, where $P(x)$ is the detection precision.

B. Tracking Framework

For a robust tracking result, we allow at most two overlapping hypotheses kept tracked at the same time and compete with each other. The overall tracking strategy is as follows: First, each target from the previous frame is tracked by correlation matching and the respective $P(x|\mathbf{E}_c,\mathbf{E}_t,\mathbf{E}_p)$ is estimated. Then, an overlap analysis is performed to distinguish *active targets* from *backup candidates*. The active targets are the ones finally used and the backup candidates are

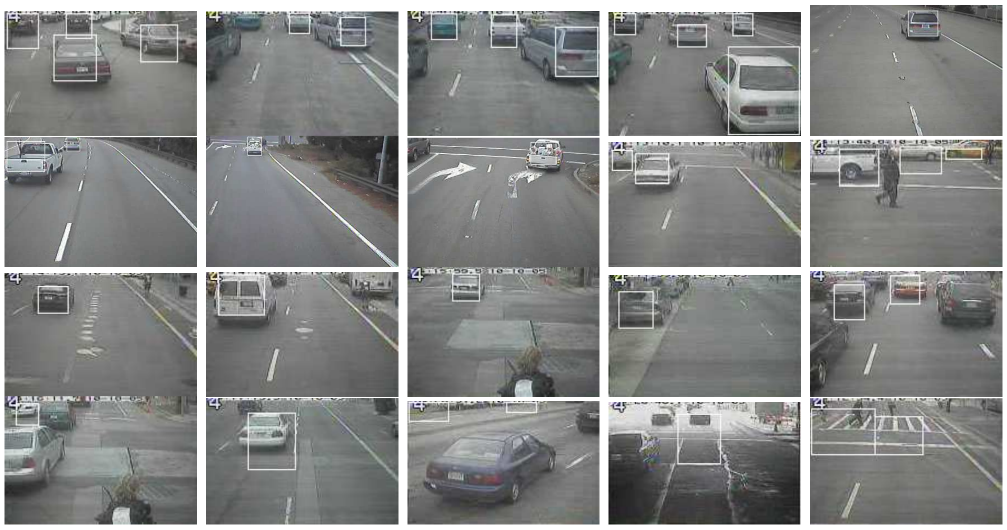

Fig. 6. Example detection results. Both good and bad (in the last row) results are shown.

tracked background for robust tracking. The next step is to perform another round of overlap analysis among the backup candidates to ensure that no backup candidates overlap each other (we allow at most two overlapping hypotheses including the active targets).

Then for each newly detected target, the initial posterior, $P(x|E_{cs})$, is estimated. If there is an overlapping backup candidates, the two posterior probabilities are compared to choose one. Otherwise, it is initially set as a backup candidate. For robustness, a newly detected target remains as a backup candidate for the first two frames, then may become an active one if it survives.

1) Results: The tracking result is very promising. False detected targets were effectively filtered out. For example, in a cluttered scene with multiple leading vehicles, false detection can occur in every two or three frames. The SVM's detection precision in the learning dataset is about 75% given 80% detection rate (where we set the threshold around). Therefore, we roughly get one false alarm per three correct detections. However, most of such false alarms were filtered out because they did not have enough feature support in the next frame, were not re-detected, or did not get a good correlation score (for example, when it was a mixture of a target and a textured background).

At the same time, the tracking failures were compensated by redundant detection. When a matching score of a target was low or when there was a drifting (thus have a low feature support), the target was replaced by its backup candidate which had a higher posterior probability. In addition, misplaced initial detections were repositioned in the same manner. Figure 7 presents examples of tracking failures or localization errors being replaced by better new detections in the following frames.

The test was performed on various video clips (the same parameters were applied to all) and an excerpt video clips containing both good and bad results can be downloaded at http://path.berkeley.edu/∼zuwhan/obstacledetection. The

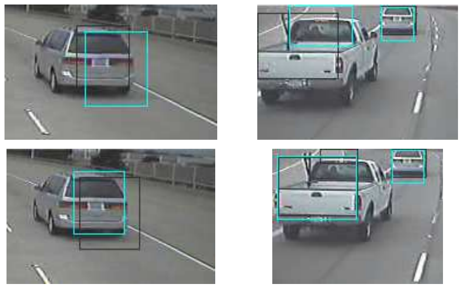

Fig. 7. At most two overlapping tracking hypotheses compete with each other. The *active targets* (cyan) are for the final use and the *backup candidates* (dark gray) are for background tracking. The example results show inferior active targets (due to tracking failure or mis-positioning in the initial detection) being replaced by the backup candidates of higher posterior probabilities.

resulting detection performance is competitive compared to those of active sensors such as LIDAR and radar. For future comparative study an example video and our result on it can also be downloaded at the same website.

The overall computation for detection and tracking was under 100ms for a 352×240 resolution video clips and about 30ms for 174×108 clips on a Pentium III 1GHz processor. Currently, a Pentium M processor of up to 1.8GHz is available for embedded computing (PC/104). Most computation were spent on the detection.

V. Summary and Conclusion

We introduced a realtime appearance-based vehicle detection algorithm based on constrained Delaunay triangulation (CDT). The proposed use of CDT provides an efficient way of collecting image features for a large number of overlapping hypotheses which could otherwise require a serious amount of computation. We also presented a probabilistic integration of detection and tracking that redundant detection in consecutive frames can compensate tracking failures and temporal redundancy for tracking can compensate detection errors such as false detection or localization errors.

The resulting detection performance is competitive compared to those of other sensors such as radar and LIDAR while the latency is significantly smaller and the lateral tracking performance is more accurate. Our future work includes:

- Enhance the detection performance further by applying more advanced features and a feature selection approach.
- Integrate the vehicle detection algorithm with an existing lane detection system.
- Find a proper sensor fusion approach to maximize the benefit of each sensor. For example, we can use distance measured by active sensors to properly locate the detected targets or we can pre-filter target locations by the vision algorithm to reduce the sensor latency.

References

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Intl. Conf. Computer Vision*, 1999, pp. 1150–1157.

[2] "Automotive collision avoidance system field operational test," U.S. Department of Transportation, National Highway Traffic Safety Administration, Final Program Report DOT HS 809 886, 2005.

[3] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: a review," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, 2006.

[4] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 264–271.

[5] R. Okada, Y. Taniguchi, K. Furukawa, and K. Onoguchi, "Obstacle detection using projective invariant and vanishing lines," in *Proc. IEEE Intl. Conf. Computer Vision*, vol. 1, 2003, pp. 330–337.

[6] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection using evolutionary gabor filter optimization," *IEEE Trans. on Intelligent Transportation Systems*, vol. 6, no. 2, 2005.

[7] P. Viola and M. Jones, "Rapid real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[8] Q. Wu and Y. Yu, "Two-level image segmentation based on region and edge integration," in *Proc. VIIth Digital Image Computing: Techniques and Applications*, 2003.

[9] X. Ren, C. C. Fowlkes, and J. Malik, "Scale-invariant contour completion using conditional random fields," in *Proc. 10th Int'l. Conf. Computer Vision*, 2005.

[10] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679–698, 1986.

[11] D. S. Guru, B. H. Shekar, and P. Nagabhushan, "A simple and robust line detection algorithm based on small eigenvalue analysis," *Pattern Recognition Letter*, vol. 25, no. 1, pp. 1–13, 2004.

[12] W. Forstner and E. Gulch, "A fast operator for detection and precise location of distinct points, corners, and centers of circular features," in *Proc. Intercommision Conf. on Fast Processing of Photogrammetric Data*, 1987, pp. 281–305.

[13] Z. Kim and R. Nevatia, "Expandable Bayesian networks for 3-d object descriptions from multiple views and multiple mode inputs," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 6, pp. 769–774, 2003.