Robust Lane Detection and Tracking in Challenging Scenarios

ZuWhan Kim, *Member, IEEE*

***Abstract*—A lane-detection system is an important component of many intelligent transportation systems. We present a robust lane-detection-and-tracking algorithm to deal with challenging scenarios such as a lane curvature, worn lane markings, lane changes, and emerging, ending, merging, and splitting lanes. We first present a comparative study to find a good real-time lane-marking classifier. Once detection is done, the lane markings are grouped into lane-boundary hypotheses. We group left and right lane boundaries separately to effectively handle merging and splitting lanes. A fast and robust algorithm, based on random-sample consensus and particle filtering, is proposed to generate a large number of hypotheses in real time. The generated hypotheses are evaluated and grouped based on a probabilistic framework. The suggested framework effectively combines a likelihood-based object-recognition algorithm with a Markov-style process (tracking) and can also be applied to general-part-based object-tracking problems. An experimental result on local streets and highways shows that the suggested algorithm is very reliable.**

***Index Terms*—Collision warning, computer vison, lane detection, part-based object tracking.**

I. Introduction

DETECTING and localizing lanes from a road image is an important component of many intelligent-transportation-system applications. There has been active research on lane detection [1]–[9], and a wide variety of algorithms of various representations (including fixed-width line pairs, spline ribbon, and deformable-template model), detection and tracking techniques (from Hough transform to probabilistic fitting and Kalman filtering), and modalities (stereo or monocular) have been proposed.

Due to a real-time constraint and, then, slow processor speed, the lane markings have been detected based only on simple gradient changes, and much of the older work has presented results on straight roads and/or highways with clear lane markings or with an absence of obstacles on the road.

Many commercial lane-detection systems are available and show good performance in many challenging road and illumination conditions. However, they do not provide lane-curvature information but just lane positions to deliver robust results. Although lane positions are sufficient for some applications, such as lane-departure warning, there are other applications which require lane-curvature information.

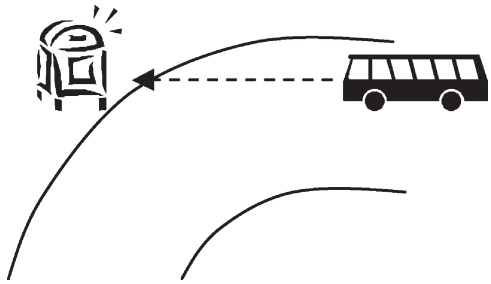

Fig. 1. False-alarm scenario of a collision-warning system. Without knowing the lane curvature, the system will generate a false alarm for the postbox.

For example, a collision-warning system can generate false alarms when the lane curvature is not known. An example scenario is shown in Fig. 1. Without knowing the road curvature, the system cannot distinguish objects on the sidewalk (e.g., the postbox) from the objects on the road, and it may generate a false alarm. As an alternative to a vision-based approach, one may use a global-positioning system (GPS) with a geographic-information system (GIS). However, the GPS has a limitation on the spatial and temporal resolution, and detailed information is often missing or not updated frequently in GIS. For example, it is important to detect the road curvature at an off-ramp because it can generate a false-collision warning, but most GPS-based systems suffer from even discriminating whether the vehicle entered an off-ramp or not.

Recent efforts deal with curved roads [5], [7]–[9], and robust detection results on challenging images, such as distracting shadows or a leading vehicle, have been reported. Some of them work in real time, and some do not.

We present a real-time lane-detection-and-tracking system which is distinguished from the previous ones in the following ways.

1) It uses more sophisticated lane-marking-detection algorithm (than gradient- or intensity-bump-based detection) to deal with challenging situations, such as worn lane markings and distracting objects/markings, for example, at an intersection and on a road surface.
2) It detects the left- and right-lane boundaries separately, whereas most of the previous work uses a fixed-width lane model. As a result, it can handle challenging scenarios such as merging or splitting lanes and on- and off-ramps effectively.
3) It combines lane detection and tracking into a single probabilistic framework that can effectively deal with

Manuscript received December 27, 2006; revised April 19, 2007, July 14, 2007, and July 31, 2007. The Associate Editor for this paper was U. Nunes.

The author is with California Partners for Advanced Transit and Highways, University of California, Berkeley, Richmond, CA 94804-4698 USA (e-mail: zuwhan@berkeley.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2007.908582

1524-9050/$25.00 © 2008 IEEE

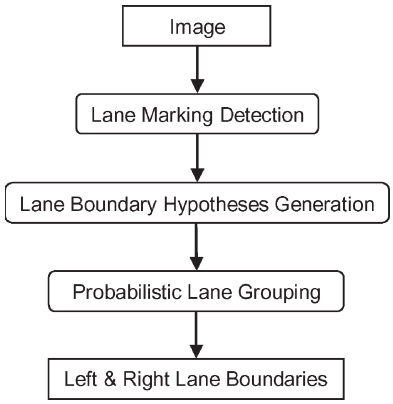

Fig. 2. Flow diagram of the algorithm.

Fig. 3. Example image and a rectified image.

lane changes, emerging, ending, merging, or splitting lanes. Much previous work has focused on lane tracking and usually uses a time-consuming detection algorithm to initialize the tracking. We introduce a fast and robust lane-detection algorithm that can be applied in every frame in real-time.

Our algorithm follows the "hypothesize and verify" paradigm. In the "hypothesize" step, lower level features are grouped into many higher level feature hypotheses, and they are filtered in the "verify" step to reduce the complexity of the higher level grouping. Fig. 2 shows the flow diagram. First, the image is rectified, assuming that the ground is flat.[1] An example image and the rectified image are shown in Fig. 3. Possible lane-marking pixels are detected in the rectified image. Then, the detected lane-marking pixels are grouped into lane-boundary hypotheses. A lane-boundary hypothesis is represented by a constrained cubic-spline curve. A combined approach of a particle-filtering technique (for tracking) and a RANdom SAmple Consensus (RANSAC) algorithm (for detection) is introduced to robustly find lane-boundary hypotheses in real-time. Finally, a probabilistic-grouping algorithm is applied to group lane-boundary hypotheses into left- and right-lane boundaries. Note that we generate left- and right-lane-boundary hypotheses separately (unlike much of the previous work which has a lane model of uniform width) to deal with various scenarios such as on/off-ramps or an emerging lane.

In Section II, a comparative study of both classification performance and computation time on various lane-marking-classification methods is presented. In Section III, we present our approach to hypothesize lane boundaries. The probabilistic-grouping algorithm is proposed in Section IV. Experimental results are presented in Section V, and we present the summary and future work in Section VI.

[1]However, a nonflat case is also addressed at a later stage (lane-boundary grouping).

Fig. 4. Example road images.

II. Lane-Marking Detection

Sample road images are shown in Fig. 4. Many of the previous algorithms simply look for "horizontal intensity bumps" to detect lane markings, which shows reasonably good performance in many cases, but it cannot distinguish false intensity bumps caused by leading vehicles and road markings/textures from weak lane markings. For example, worn yellow markings often have similar grayscale intensity to the road pixels. In addition, we sometimes need to deal with a poor image quality, for example, when we need to postprocess an MPEG data.

To deal with such problems, we apply machine learning. We applied various classifiers to the lane-marking-detection task and present a comparative analysis. Since the size of the lane marking changes dramatically with respect to its distance from the car, we need to normalize them to apply a standard classifier. Therefore, we first rectify the original image, as shown in Fig. 3. When we assume that the ground is flat (for this stage only), we can apply a plane homography to find an image rectification. A point (x, y) on the rectified image corresponds to the point (u, v) in the original image, where

$$\begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix} = \mathbf{H} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

and $\mathbf{H}$ is a homography matrix. A homography matrix can easily be obtained by applying a simple external camera calibration with four reference points. Details on the plane homography can be found in many computer-vision textbooks, for example [10].

When a plane homography is given, image rectification is done in the following manner: For each pixel (x, y) of the rectified image, its correspondence (u, v) on the original image is obtained. Since u and v are not integer numbers in most cases, the pixel value of the rectified image is calculated by linearly

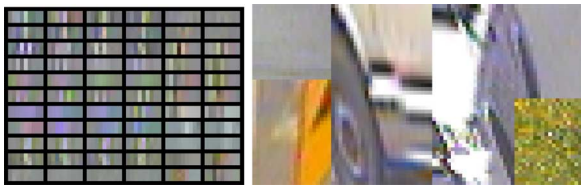

Fig. 5. Example image patches of lane markings and nonmarkings.

interpolating the intensity values of the four neighboring pixels (by flooring and ceiling the u and v) in the original image.

Once we have a rectified image, a lane-marking classifier is applied on a small image patch around each and every pixel. A typical width of the lane marking on the rectified images are about three pixels. Therefore, raw pixel values of a 9×3 window were used as inputs (total 27 features for a grayscale image and 81 RGB values for a color image). To find a suitable classification algorithm, we tested various classifiers.

Applying a stereo algorithm [3], [4] can further improve the lane-marking-detection performance, but we focus on a monocular image in this paper.

For learning, we have gathered image patches of 421 lane markings and 11 124 nonmarkings. Fig. 5 shows example image patches. We observe a variety in colors, textures, and width. We compared the classification performances and the computation requirements of various classifiers on the data set. The following classifiers were considered.

1) **Intensity-Bump Detection:** Intensity-bump detection is the most popular method in the lane-detection literature. It is the simplest and fastest detection method, and it can also be applied to nonrectified images. We use an implementation by Ieng *et al.* [6]. We applied various values for the gradient threshold (s_0) to control the tradeoffs between the detection rates and the false-alarm rates.
2) **Artificial Neural Networks (ANNs):** We tested two-layer neural networks with various numbers of hidden nodes. Training an ANN (a back-propagation algorithm was applied in our experiment) requires significant computation, but the actual classification time is relatively small. When there are n features (inputs) and m hidden nodes, it requires nm multiplications, $nm + m$ additions, and m sigmoid-function calculations to classify a hypothesis ($n = 27$ or 81 and $m = 7$ in our examples).
3) **Naive Bayesian Classifiers (NBC):** NBCs show good classification performances, in spite of their unrealistic conditional independence assumption. We compare the discrete and the unimodal Gaussian representations of the conditional probability. For both representations, the learning time is linear to the number of the examples (fastest). A discrete NBC requires very little computation for classification. The Gaussian representation requires computation of the exponential function n times. However, we can avoid calling the exponential function by using a logarithm of the probability instead of the actual one. In fact, for both representations, it is necessary to use a logarithm to minimize the numerical errors, particularly when the number of features is large. For the discrete NBC, we can precalculate the logarithms of all the probability table entries to save computation.
4) **Support Vector Machine (SVM):** During the last decade, SVMs have rapidly gained popularity. They provide a good framework for incorporating kernel methods. We tested the second-order polynomial kernel, which requires the smallest computation. Learning requires significant computation, but it is bounded in polynomial time. The classification involves a large number of multiplications: $O(mn)$, where m is the number of support vectors. The number of support vectors is at least $n + 1$, and it can be much greater when the data is not clearly separable (in the transformed feature space) or when a small tuning parameter is given. For training, we used the implementation by Collobert *et al.* [11] (SVMTorch) with the tuning parameter of 100.

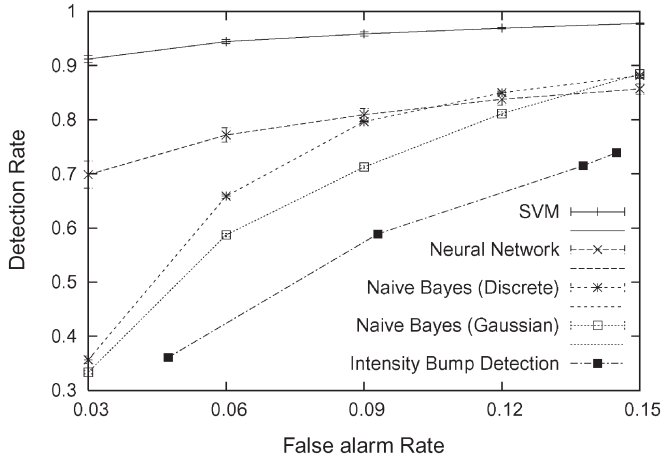

Fig. 6. Classification performance of the classifiers.

Details on most of the above classifiers can be found in the machine-learning literature, for example, in [12].

Fig. 6 shows the classification performances of the presented classifiers. We followed the evaluation scheme presented in [13]. We repeated stratified fivefold cross-validation ten times and showed the receiver operating-characteristic (ROC) curves with the confidence intervals. For all the classifiers, we obtained the ROC curves by changing only the threshold values (no relearning with different parameters).

For all the classifiers, we applied various parameters and chose the best ones. For ANN, we compared the ones with seven, 10, and 15 hidden nodes, but we present the result of the one with seven hidden nodes because it is the fastest, whereas the performances among them are not significantly different. For the discrete naive Bayesian network, we used seven-level discretization. The SVM was learned with the tuning parameter of 100.0.

We observe that all the classifiers show superior performance than the intensity-bump detector. In fact, intensity-bump detectors introduce too many false alarms, given an acceptable detection rate. Therefore, applying any of the above classifiers will deliver much better lane-detection performance. The SVM

TABLE I
COMPUTATION TIME OF THE CLASSIFIERS

Classifier	Classification time
Intensity Bump Detector	∼ 10 ms
ANN (7 hidden nodes)	∼ 100 ms
Gaussian NBC	∼ 25 ms
Discrete NBC	∼ 60 ms
SVM	∼ 2.2 sec
Cascade	∼ 25 ms in worst cases

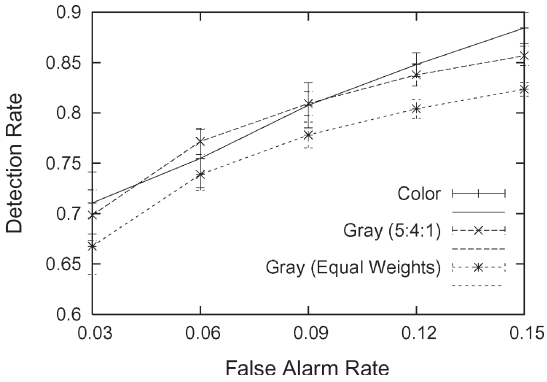

Fig. 7. Classification performance with neural networks when directly using color pixels (81 features), gray-level pixels with 5 : 4 : 1 weights, and gray-level pixels with equal weights. Using a gray image with 5 : 4 : 1 weights gives competitive performance to that of using a color image.

shows far better performance than any other classifiers, and then, the ANN follows.

We also compared the classification computation for the classifiers. We have applied the classifiers on images of a 70 × 250 size and summarized the computing time in Table I. The algorithms ran on an Intel Core 1.83-GHz processor. For fair comparison, all the classification algorithms were implemented in C++ inline functions and optimized to bring maximum performance.

Unfortunately, the SVM was not fast enough for real-time classification, and we chose to use the ANN. To further reduce the computation time, we applied a cascade classification: First, a simple gradient detector and an intensity-bump detector with loose (low) threshold values are successively applied to quickly filter out nonlane markings, and then, the ANN classifier is applied to the remaining samples (much smaller in number). As shown in Table I, it significantly reduces the classification time.

We used gray-level lane-marking images for the above classification result and the computation-time analysis. The gray-level images were generated by weighted-summing RGB values (0.5 for red, 0.4 for green, and 0.1 for blue) to better detect worn yellow lane markings. Applying such weights outperformed the equal-weight conversion, as shown in Fig. 7. We have tested various different weight combinations, and the proposed weights showed the best performance. One may apply the classifier directly to the color pixels (total 81 features), but it introduces too much computation in image-rectification classification, whereas it does not improve the performance significantly, as also shown in the Fig. 7.

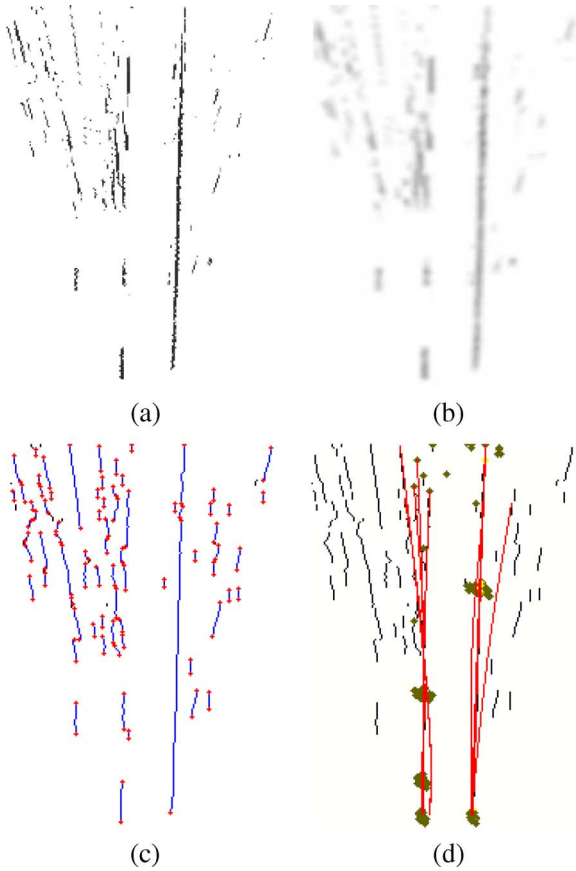

Fig. 8. (a) Detected lane-marking pixels. (b) Smoothed lane-marking score. (c) Line-segment grouping. (d) Selected hypotheses from the particle-filtering/RANSAC algorithm.

III. LANE-BOUNDARY-HYPOTHESES GENERATION WITH PARTICLE FILTERING AND RANSAC

Once possible lane-marking pixels are detected [an example is shown in Fig. 8(a)], they are grouped into uniform cubic-spline curves of two to four control points. Splines are smooth piecewise polynomial functions, and they are widely used in representing curves. Various spline representations have been proposed, and we use a cubic spline among them. In a cubic-spline representation, a point p on a curve between the ith and $(i+1)$th control point is represented as

$$p = (x_i(t), y_i(t))$$

where

$$x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$$
$$y_i(t) = e_i + f_i t + g_i t^2 + h_i t^3$$

where the parameters $a_i, \ldots, h_i$ are uniquely determined by the control points so that the curve is smooth. $(x_i(0), y_i(0))$ is the ith control point, $(x_i(1), y_i(1))$ is the $(i+1)$th control point, and $0 \leq t \leq 1$.

A cubic-spline curve enables fast fitting, because the control points are actually on the curve. We use this property to apply a RANSAC algorithm [14]. A RANSAC algorithm is a robust

fitting algorithm that has successfully been applied to various computer-vision problems. In [7], Wang *et al.* used a B-spline curve to represent a curved road. In a B-spline representation, control points reside outside of the curve, and its fitting procedure requires a significant number of iterations. On the other hand, the uniform cubic-spline fitting is much faster, but it may result in irregular curves when the spacing between the control points are not even. However, such irregular curves can be filtered out by applying a RANSAC technique while still maintaining the computation manageable. We also impose additional constraints to the cubic spline to prevent unreasonable hypotheses from being formed: $y_i(t)$ should be monotonic for all i, and the maximum curvature should be smaller than 0.05.

Our RANSAC fitting procedure is as follows. First, lane-marking points are grouped into line segments. Since the lane-marking detection result is noisy [Fig. 8(a)], we apply Gaussian smoothing [Fig. 8(b)] followed by nonmaxima suppression to remove noise. Then, we follow a line-grouping approach proposed in [15] to generate line segments, as shown in Fig. 8(c).

The next step is to generate hypotheses. Each hypothesis is generated from a random set of one, two, or three line segments. First, for each hypothesis, it is randomly determined whether to use two, three, or four control points. A spline curve with two control points represents a line, three for an approximate arc, and four for a more complex curve. Increasing numbers of control points increases the representational power, but at the same time, it decreases the robustness of the fitting. The use of a RANSAC algorithm makes it easier to handle this tradeoff because many hypotheses of various numbers of control points are generated and then evaluated by a single scoring function (presented in Section IV-B), which can handle this tradeoff.

A straight line of two control points is generated from a random set of one or two line segments. Whereas a single-line segment is sufficient to make a straight-line hypothesis, we also use a pair of line segment for robust fitting. Note that we still need hypotheses from single line segments, because sometimes, only one line segment is detected for a lane boundary, for example, for solid lane markings.

An approximate arc of three control points is generated from a random set of two line segments, and a more complicated hypothesis of four control points is generated from a random set of three line segments.

For robust hypothesis generation, some constraints were imposed. For example, the first (nearest) line segment should be close enough, for example, within 15 m, to the vehicle. In addition, the two nearby line segments should be on a common arc of a reasonable curvature within a small error bound.

The first (nearest) control point is forced to be on the bottom of the rectified image. Its position is computed by extrapolating the fitted arc of the first two line segments. The last control point is set to the end point of the farthest line segment. It is important to force the control points in the middle to be evenly spaced, because uneven spacing of control points makes the uniform spline fitting unstable. Therefore, the rest of the control points are chosen from the end/center points of the line segments, where the spacings are as even as possible: For an approximate arc with three control points, (x_1, y_1), (x_2, y_2), and (x_3, y_3), the second control point (x_2, y_2) is chosen to maximize $\min(d_2, d_1)/\max(d_2, d_1)$, where $d_i = y_{i+1} - y_i$. For a complex curve with four control points $(x_1, y_1), \ldots, (x_4, y_4)$, the second and the third control points are chosen to maximize $\min(d1/d2, d2/d3)/\max(d1/d2, d2/d3)$.

In our implementation, at most 100 random hypotheses are generated for each frame. Once hypotheses are generated and validated, an overlap analysis is performed. The hypotheses are evaluated based on their lane-marking support, and a curve penalty scores. Details on the above scores are presented in Section IV-B. As a result, a small number of nonoverlapping (but may partially be overlapping) hypotheses are finally selected.

Another set of hypotheses are generated by applying a tracking algorithm. Our tracking algorithm is based on a particle-filtering technique [16]. Other tracking techniques, such as Kalman filtering or active contours (SNAKE), can also be used within this framework. However, we chose a particle-filtering algorithm over the Kalman filter to prevent the result from being biased too much on the predicted vehicle motion but to give more weight to the image evidence. Due to vehicle's vibration, including pitch change, the motion of the lane boundaries in world (vehicle) coordinates is not smooth enough to be properly modeled by a Kalman filter. Active contours (such as the one used in [7]) may provide a good tracking result, but it requires too much computation as compared to the proposed particle-filtering algorithm.

For the particle filtering, the vehicle's motion (rotation and translation) was modeled by Gaussian distributions for simplicity, but the scoring function is carefully designed to prevent the result from being dictated by this model. Our particle-filtering algorithm is as follows: 1) Given a randomly selected motion, a pair of lane-boundary hypotheses is generated by moving the previously detected lane boundary's control points according to the motion; 2) the position of the last control point is adjusted to a lane-marking pixel near the extrapolated curve; 3) a number of hypotheses are generated (50 left and right boundaries, in our implementation) and scored based on the supporting lane-marking pixels; and 4) finally, the control-point position of the final boundary hypothesis is set as the weighted sum over all the hypothesized positions, where the weights are based on the lane-marking support scores of the corresponding curve hypotheses. It is very important that the weights be able to sharply distinguish good hypotheses from relatively bad hypotheses to prevent the result from being dictated by the initial motion model. Therefore, the weights were exaggerated by applying a sharp sigmoid function with the mean being its curve score in the previous frame.

Due to the vehicle's forward motion, all the control points move closer to the vehicle. Therefore, we need to adjust the position of the first control point so that it does not disappear from the image. We enforce that the first control point's position will always be on the bottom of the image. The adjusted position is calculated by interpolating the curve. The second control point is also examined to see if its position is too low, because if it keeps going down, it will eventually collide with the first control point. In addition, the uneven spacing of the control points will make the uniform spline fitting unstable. Therefore, when the position of the second control point is too

low, it is removed, and a new control point is inserted before the last control point.

The selected lane-boundary hypotheses from the RANSAC algorithm and the particle-filtering process are shown in Fig. 8(d). The particles are also shown as clouds of yellow/green dots near the control points, where the color of a dot represents the weight (the yellower, the higher). In our implementation, up to five hypotheses per lane boundary (left/right) are selected, including the ones from the particle-filtering process.

IV. Probabilistic Grouping of Lane Boundaries

Our next goal is to choose the best pair from the selected hypotheses shown in Fig. 8(d). We apply probabilistic reasoning for decision making. We use three types of evidence features: lane-marking support of the boundary hypotheses, compatibility of the two boundary hypotheses, and the temporal coherence. In fact, this is a typical object-tracking problem. Whereas most of the previous work tracked an object as a whole, our problem requires part-based tracking. In this section, we introduce a new formulation for object tracking, which fits particularly well with the "hypothesize and verify" paradigm of object detection.

A. Probabilistic Grouping for Part-Based Tracking

We use capital letters, such as X, to denote random variables and lowercase letters, such as x, to denote certain assignments taken by those variables. We also consider missing parts, and the statement $P(X = \phi)$ is used as a shorthand for $P(X =$ missing$)$. A set of multiple variables or assignments are denoted by boldface letters, such as $\mathbf{X}$ and $\mathbf{x}$. For example, in the lane-tracking application, $\mathbf{X} = \{L, R\}$, where L is for the left and R is for the right lane-boundary hypotheses.

Typical temporal reasoning models, such as dynamic Bayesian networks [17], use a posterior probability to select the best hypothesis from predefined candidates

$$\arg\max_{x_i} P\left(X = x_i|\mathbf{e}\right) \tag{1}$$

where X is the target random variable, x_i represents a specific candidate, and $\mathbf{e}$ is observed evidence (such as lane-marking scores or how well the tracking result fits to the motion model—see Section IV-B for more details). Note that

$$\sum_{x_i} P\left(X = x_i|\mathbf{e}\right) = 1. \tag{2}$$

However, many of the object-recognition approaches use the maximum-likelihood estimate [18], [19]. Their goal is to find

$$\arg\max_{X_i} P\left(X_i = \text{true}|\mathbf{e}\right) \tag{3}$$

where X_i is a random variable for a specific hypothesis. In this case, the summation of the posterior probabilities (2) does not have any meaning, but

$$P\left(X_i = \text{true}|\mathbf{e}\right) + P\left(X_i = \text{false}|\mathbf{e}\right) = 1. \tag{4}$$

There are various reasons to use the maximum-likelihood estimates. First of all, many object-recognition algorithms use the "hypothesize and verify" paradigm, and the goal is to choose the best one from a large (and not predetermined) number of overlapping hypotheses. In fact, it makes more sense to use the likelihood estimates because the generated hypotheses are not really disjoint. Many of them are overlapping or share common parts, and selecting one does not exactly mean rejecting another. In addition, it is much easier to model a binary-classification problem than a multiclass one particularly for learning.

We introduce a combination of the two approaches. Our approach uses the likelihood estimates, but its temporal reasoning is based on dynamic Bayesian network's formulation. For each hypothesis, $\mathbf{X}_i$, our goal is to estimate $P(\mathbf{x}_i|\mathbf{e})$. $\mathbf{X}_i$ consists of n parts $X_i^1, \ldots, X_i^n$. We use likelihood estimates here, and x_i^j stands for either $X_i^j =$ true or $X_i^j =$ false. In addition, we consider a part being misdetected or missing, and we use the notation x_i^j to also denote $X_i^j = \phi$.[2] For example, we estimate $P(L_2 = \text{true}, R_1 = \text{true}|\mathbf{e})$, as well as $P(L_2 = \text{true}, R = \phi|\mathbf{e})$, for lane tracking.

When we assume that the evidence variable $\mathbf{e}$ is a set of three types of independent evidence variables, $\mathbf{e} = (\mathbf{e_c}, \mathbf{e_t}, \mathbf{e_p})$, where $\mathbf{e_c}$ is a set of evidence collected in the current frame, $\mathbf{e_p}$ is a set of evidence collected in the past, and $\mathbf{e_t}$ is a set of transitional evidence (such as temporal correlation). Then, we want to know $P(\mathbf{x}_i|\mathbf{e_c}, \mathbf{e_t}, \mathbf{e_p})$ for each and every possible combinations of the part hypotheses. Applying Bayes' rule

$$P\left(\mathbf{x}_i|\mathbf{e_c}, \mathbf{e_t}, \mathbf{e_p}\right) = \alpha P\left(\mathbf{e_c}|\mathbf{x}_i\right) P\left(\mathbf{x}_i|\mathbf{e_t}, \mathbf{e_p}\right) \tag{5}$$

where $\alpha = 1/P(\mathbf{e_c})$ is a normalizing constant.

Estimating $P(\mathbf{e_c}|\mathbf{x}_i)$, given an image frame, is well-studied in the object-recognition field. The challenge is to estimate $P(\mathbf{x}_i|\mathbf{e_t}, \mathbf{e_p})$. To reduce complexity, we apply the Markov assumption that the history is conditionally independent given a previous state

$$P\left(\mathbf{x}_i|\mathbf{e_t}, \mathbf{e_p}\right) = \sum_{\mathbf{h}_k} P\left(\mathbf{x}_i|\mathbf{H} = \mathbf{h}_k, \mathbf{e_t}\right) P\left(\mathbf{H} = \mathbf{h}_k|\mathbf{e_p}\right) \tag{6}$$

where $\mathbf{H}$ is a random variable for the previous object, and $\mathbf{h}_k$ represents individual hypotheses or missing parts: $\sum_{\mathbf{h}_k} P(\mathbf{H} = \mathbf{h}_k) = 1$.

Note that we not only use the previously detected object but also use many of the previously rejected hypotheses. There are two reasons for this. First, there are cases in which some hypotheses are generated, but none of them are strong enough to be accepted. It could be a false alarm, (noise) or an object of a relatively weak evidence support. We want to choose an object, which might have relatively weaker image support $(\mathbf{e_t})$ but is consistently observed over time, rather than a false alarm, which might have stronger $\mathbf{e_t}$ at one frame but came out of nowhere (weak temporal support). Using the information on the rejected hypotheses can deal with this problem. Second, when an object is in a transitional stage (for example, a lane

[2]Note that it is still a binary, not ternary, classification problem in its nature. The expression $X_i^j = \phi$ is used for a notational convenience, but it does not mean that its value is ϕ.

boundary is split into two lines at the off-ramp), two or more strong hypotheses might be competing. The original hypotheses will eventually get weaker and weaker image support but will still have strong temporal support. The emerging one will get stronger image support but no temporal support unless the rejected hypotheses of previous frames are taken into an account.

The problem we face is that what we estimate from the previous frame are likelihood estimates of the hypotheses, not $P(\mathbf{H}=\mathbf{h}_k|\mathbf{e_p})$, except when $h_k=\phi$. To deal with this, we estimate $P(\mathbf{H}=\mathbf{h}_k|\mathbf{e_p})$ from the previous likelihood estimates. Some members of $\mathbf{h}_k$ may represent missing parts, and we separate missing parts from the detected parts $\mathbf{H}=(\mathbf{D},\mathbf{M})$, where $\mathbf{D}$ represents the detected parts, and $\mathbf{M}$ represents the missing parts. We use the following approximation:

$$\begin{aligned} P(\mathbf{H}=\mathbf{h}_k|\mathbf{e_p}) &= P(\mathbf{D}=\mathbf{d_k},\mathbf{M}=\phi|\mathbf{e_p}) \\ &\approx P(\mathbf{D}\neq\phi,\mathbf{M}=\phi|\mathbf{e_p})\frac{P(\mathbf{D}_k=\text{true}|\mathbf{e_p})}{\sum_i P(\mathbf{D}_i=\text{true}|\mathbf{e_p})} \end{aligned} \tag{7}$$

where $\mathbf{D}_i$ is all the selected hypotheses in the previous frame (cf. Section III). $P(\mathbf{D}\neq\phi,\mathbf{M}=\phi|\mathbf{e_p})$ can be obtained from the previous likelihood estimates. For example

$$P\left(L\neq\phi,R=\phi|\mathbf{e}\right)=P\left(R=\phi|\mathbf{e}\right)-P\left(L=\phi,R=\phi|\mathbf{e}\right) \tag{8}$$

where $P(R=\phi|\mathbf{e})=P(L_k=\text{true},R=\phi|\mathbf{e})+P(L_k=\text{false},R=\phi|\mathbf{e})$ for any L_k.

For $P(\mathbf{x}_i|\mathbf{H}=\mathbf{h}_k,\mathbf{e_t})$, we assume that the individual parts' tracking histories are independent from each other

$$P\left(\mathbf{x}_i|\mathbf{H}=\mathbf{h_k},\mathbf{e_t}\right)=\prod_j P\left(x_i^j|H^j=h_k^j,\mathbf{e_t}\right). \tag{9}$$

When $h_k^j\neq\phi$

$$\begin{aligned} &P\left(x_i^j|H^j=h_k^j,\mathbf{e_t}\right) \\ &\quad=P\left(x_I^j|H_k^j=\text{true},\mathbf{e_t}\right) \\ &\quad=\beta P\left(\mathbf{e_t}|x_i^j,H_k^j=\text{true}\right)P\left(x_i^j|H_k^j=\text{true}\right) \end{aligned} \tag{10}$$

where β is a normalizing constant, $P(\mathbf{e_t}|x_i^j,h_k^j)$ can be learned by examples, and $P(x_i^j|H_k^j=\text{true})$ is assumed to be a constant (for all i). When $h_k^j=\phi$, $P(x_i^j|H_k^j=\phi,\mathbf{e_t})$ is assumed to be a constant for all i.

Since x_i^j can also be missing, we have four constants for each part, which works as system parameters: $P(x^j|H^j\neq\phi)$, $P(X^j=\phi|H^j\neq\phi)$, $P(x^j|H^j=\phi)$, and $P(X^j=\phi|H^j=\phi)$. Since $P(x^j|X^j=\phi)=0$, $P(x^j|H^j\neq\phi)=P(x^j|X^j\neq\phi)P(X^j\neq\phi|H^j\neq\phi)$ and $P(x^j|H^j=\phi)=P(x^j|X^j\neq\phi)P(X^j\neq\phi|H^j=\phi)$. Therefore, we only have three independent parameters.

1) $P(x^j|X^j\neq\phi)$: A prior probability of a part hypothesis given that the part is detected (or exists). It roughly determines the overall detection rate. In our lane-tracking system, it was set to 0.999 for all parts.
2) $P(X^j\neq\phi|H^j=\phi)$: The probability of an emerging part. The larger the parameter is, the more spontaneously the system reacts to emerging hypotheses and noises. However, when the detection performance is good enough, it is good to give a reasonably large weight to this because redundant detection compensates tracking failures. In our lane-tracking system, it was set to 0.1 for all parts.
3) $P(X^j=\phi|H^j\neq\phi)$: The probability of a disappearing part. The smaller the parameter is, the more the system depends on the tracking. However, making it zero can cause a false detection or localization error when the part is disappearing. In our lane-tracking system, a very low number $(1e^{-8})$ was assigned for all parts, because tracking is much more robust than detection in most cases.

B. Application to Lane-Boundary Grouping

In the lane-boundary-grouping case, $\mathbf{X}=\{L,R\}$. To get $P(\mathbf{e_c}|l,r)$, we separate the evidence variable $\mathbf{e_c}$ into the ones that are independent to each other ($\mathbf{e_l}$ for the left hypotheses and $\mathbf{e_r}$ for the right ones) and the dependent ones ($\mathbf{e_{lr}}$). Then, $P(\mathbf{e_c}|l,r)=P(\mathbf{e_l}|l)P(\mathbf{e_r}|r)P(\mathbf{e_{lr}}|l,r)$.

For $\mathbf{e_l}$ and $\mathbf{e_r}$, we use lane-marking support scores and curve penalties. The lane-marking support score is obtained from the smoothed lane-marking-score image [Fig. 8(b)]

$$\text{LaneMarkingSupport}=\sum_p I(p)$$

where p is a pixel on the lane-boundary hypothesis, and $I(p)$ is the smoothed lane-marking score (the intensity of the pixel). In addition to the lane-marking-support score, a curve-penalty score is used to prevent overfitting. A penalty is imposed when the direction of the curve is changed without lane-marking support

$$\text{CurvePenalty}=\sum_{p_1,p_2}\left|[y(p_2)-y(p_1)]\left[\frac{dx}{dy}(p_2)-\frac{dx}{dy}(p_1)\right]\right|$$

for all points pair p_1 and p_2 where there is no lane-marking support in between them.

When $P(\mathbf{e_l}|l)$ or $P(\mathbf{e_r}|r)$ is estimated from the lane-marking support and curve-penalty scores, we can also calculate the posterior probability of a lane-boundary hypothesis given these scores $P(l|\mathbf{e_l})$ or $P(r|\mathbf{e_r})$ by applying Bayes' rule. These posterior probabilities are used in the hypotheses-generation stage for hypotheses selection (Section III).

For $\mathbf{e_{lr}}$, we examine the average lane width, the change of the lane width, and a lane-incompatibility penalty. Note that the lane width can slightly be increasing or decreasing (within an image) due to the pitch movement of the vehicle, presence of up/down-hills and/or changing road width. We assume that the lane width can linearly be increasing or decreasing at a small ratio. Given a lane-boundary pair, the lane width is sampled at various distance and fit into a linear equation to

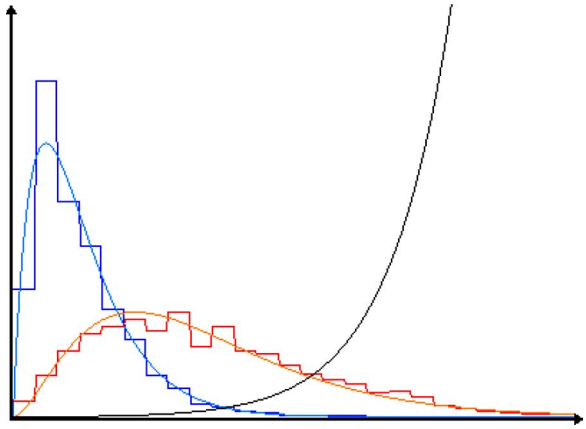

Fig. 9. Analyzing grouping parameters. For the lane-marking-support score, gamma distributions fit well with both positive and negative examples. When fitting parametric distributions, we need to also examine the likelihood (black curves) to see if it is reasonable. For example, it is supposed to be monotonically increasing for this variable.

obtain the average lane width and the change of the lane width. The lane-incompatibility penalty is the maximum residual distance.

For $P(\mathbf{e_t}|l, l_{\text{prev}})$ and $P(\mathbf{e_t}|r, r_{\text{prev}})$, we examine the angle change, the lateral movement, the shape difference (obtained in a similar manner to the lane-incompatibility penalty), and the difference in length (to impose a penalty for a short hypothesis).

There is no hidden variable. Therefore, all the probability distributions [including the background models, such as $P(\mathbf{e_l}|L_i = \text{false})$] can directly be learned from positive and negative examples. It is very tedious to manually give a large number of positive and negative examples. Therefore, a semisupervised-learning approach was applied to estimate the parameters. The procedures are as follows.

1) The probability-distribution parameters are given manually to generate a reasonable result. This is not very difficult because all the probability distributions are very intuitive.
2) Automatic-detection results are used as a ground truth to estimate the parameter. It is not easy to collect the negative hypotheses because many of the rejected hypotheses are not really negative hypotheses but just ones with lower likelihood. To deal with this problem, we can first apply a loose threshold at the hypotheses-selection stage that "bad" hypotheses can also be generated. Then, the hypothesis of the lowest likelihood is chosen as a negative example. The negative hypothesis should have a significantly lower likelihood than that of the misdetection.
3) The lane-detection algorithm is once again applied with the new parameters. Go to the step 2), if necessary.

We applied various parametric distributions on the collected examples, such as Gaussian, half-Gaussian, gamma, and discrete distributions, by examining the positive- and negative-learning data collected in step 2). An example for the lane-marking-support score is shown in Fig. 9. Gamma distributions fit well to both positive and negative examples. We also need to carefully examine that the likelihood is reasonable. For example, using a pair of Gaussian distributions with different means and variances will result in nonmonotonic likelihood. Since most of the evidence variables are scores or penalties, using a nonmonotonic-likelihood function for such variables will generate classification outliers.

Fig. 10. Horizon detection of the CHEVP algorithm [7] can easily be distracted by ground markings and leading vehicles. In our implementation (for the comparison purpose), we did not use the horizon voting but estimated it from the calibration parameters to get better results.

V. Experimental Results

We first present an experimental result on the detection algorithm only (without tracking). For comparison, we implemented the Canny/Hough Estimation of Vanishing Points (CHEVP) algorithm. In [7], Wang *et al.* introduced the CHEVP algorithm to initialize their B-spline SNAKE tracking algorithm. The CHEVP algorithm performs the following: 1) It detects the line segments using the Hough transformation on a vertically segmented image; 2) performs voting to find the horizon; 3) finds the lane-boundary candidates in each segment of the image based on the detected horizon; and 4) fits them into a spline curve. The algorithm was implemented such that it shows similar results on all the images presented in the algorithm website (http://www.ntu.edu.sg/home5/ps2633175g/chevp.htm).

Some minor modifications were made to the original CHEVP algorithm to improve the result on our test video clip. For example, the horizon detection is sensitive to noise by leading vehicles and/or nonlane markings on the road (Fig. 10). Instead of detecting the position of the horizon in each frame, we derived it from the calibration parameters. Although the actual horizon positions slightly vary according to the vehicle's pitch movement, it is still much more accurate and robust to use a fixed one than to detect them in every frame.

For our algorithm, we only used the RANSAC-based lane-boundary detection and the probabilistic grouping based only on the evidence of the current frame. Both algorithms were tested on a short video clip of 352 $\times$ 240 image resolution (total of 923 frames). Our detection algorithm showed correct detection in over 80% of the frames. Misalignments (mostly minor) occurred in less than 4% of the image frames, and in the rest of them (14%), detection failed. For CHEVP, we applied a more liberal measure for the misalignment because, in the work of Wang *et al.* [7], the final result was refined using a SNAKE fitting. Even ignoring the minor misalignments, the result was still not comparable to our results. The detection was roughly okay only in 52% of the image frames, where there were significant misalignments in 11% of them and misdetections in 37%. The Hough transformation failed to grab many of the dashed lane markings [Fig. 11(a)], whereas picking up false lines resulted in misalignments

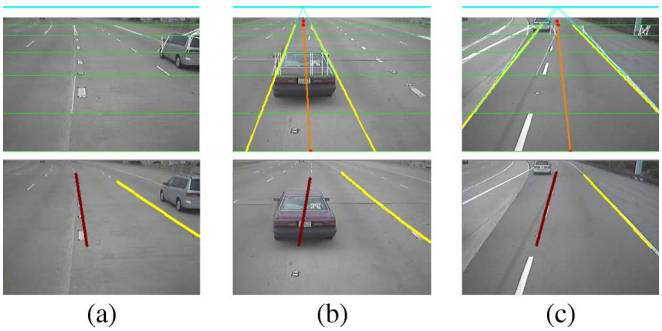

Fig. 11. (Upper) Misdetection and false detections by the CHEVP algorithm. (Lower) Our RANSAC algorithm correctly found the lane boundaries from the same images. For the CHEVP algorithm, the orange line is the detected lane center, and the yellow lines are the boundaries. For the RANSAC algorithm, the orange line is the left boundary, and the yellow line is the right boundary.

[Fig. 11(b) and (c)]. The resulting video clips can be downloaded at http://path.berkeley.edu/~zuwhan/lanedetection/.

We now present the experimental result on the complete algorithm (both detection and tracking). We tested the algorithm on a 3-min video clip taken from a normal bus route. The video clip is in an MPEG format of 176 × 120 image resolution. The video quality and the resolution is much worse than that of the previous one and those used in most of the previous works. In addition, it contains all the difficult scenarios, including lane changing, merging/emerging/splitting lanes, and ending lanes (at intersections). In addition, the overall lane-marking condition is much poorer. For example, the contrast between some of the worn lane markings and the worn road is very low.

The proposed algorithm was run at 10 frames/s with a total of 1951 frames. An average computation time (including the operating-system overhead, such as virtual-memory usage but not including video-reading/decoding time) was under 60 ms/frame on an Intel Pentium 4 3-GHz processor. The program was implemented in C++ on Microsoft Windows using the OpenCV library. The computation time did not increase significantly (still under 60 ms) on a higher resolution video (352 × 240), because most of the computation was done on the rectified images of fixed size. Therefore, whenever available, we can use a higher resolution video to increase the performance—better performance is expected with a higher resolution video because we can obtain a rectified image of better quality. Note that some of the previous lane-detection algorithms do not work in real-time. For example, Wang *et al.* [7] require up to 4 s for detection and 2 frames/s for tracking on Intel Pentium III.

Example results are shown in Figs. 12 (good detection results) and 13 (bad detection results). Since there is no absolute ground truth and it is difficult to quantize the result, we categorized the bad detections into four categories: misdetecting both sides of the lane boundaries, misdetecting one of two lane boundaries, false alarms or major misalignments, and minor misalignments.

It is difficult to determine good detections from minor misalignments from major misalignments, but the results in Fig. 13 show our guidelines. Fig. 13(k) is a minor misalignment, and Fig. 13(l)–(o) are major misalignments.

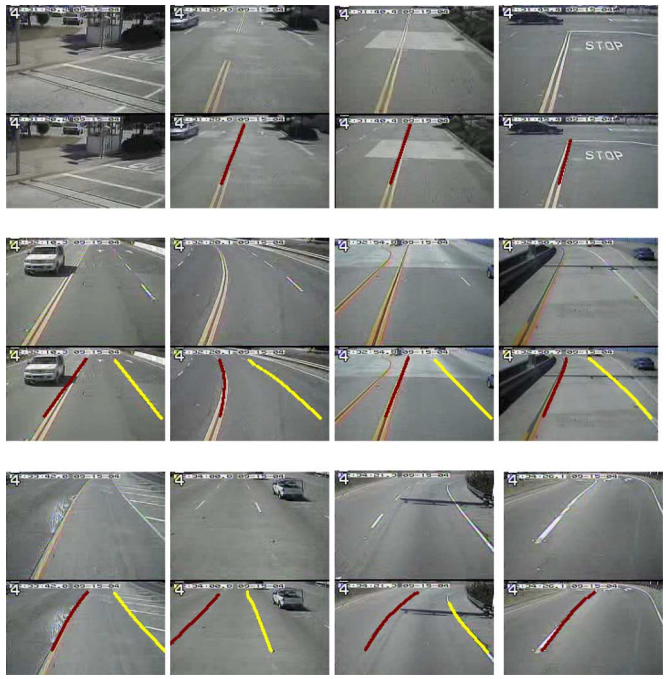

Fig. 12. Example detection results.

There was only one false-alarm case, which was caused by another valid lane boundary [Fig. 13(p)]. However, it was eventually taken over by the correct one due to our probabilistic-tracking framework that combines redundant detection with tracking. Another example of correcting false detection is shown in Fig. 14. A misalignment was corrected in the next frame. In fact, these correction abilities distinguish our method from most of the previous algorithms, which are heavily dependent on temporal filtering.

Most of the major misalignment was caused by the low image quality. For example, MPEG compression side effects sometimes cause false alarms in lane-marking detection, and they can easily cause misalignments at the end of the curve. Most of the misdetection was caused by lane-marking-detection failure or lack of actual lane markings. Many of the lane-marking-detection failures occurred because of steep curves and/or because of nonflat ground. Further research is required to improve the lane-marking detection in such cases.

Some misdetection was caused by strong shadows by overpasses, as shown in Fig. 13 (first row). Such a failure is due to the limitation of low-end autoiris cameras. Due to the high contrast between the shadow and nonshadow areas, lane markings inside the shadow have a very low contrast. Note that the suggested algorithm can fairly handle other challenging illumination conditions (Fig. 15).

Another major case of spontaneous misdetection is due to fast lane changes (first two of the third row). Since the lane-boundary movement is unusual in such cases, some of the correct lane-boundary hypotheses with weak lane-marking supports were rejected.

The quantitative result is summarized in Table II. The resulting video clips can be downloaded at the same website (http://path.berkeley.edu/~zuwhan/lanedetection/). In the video

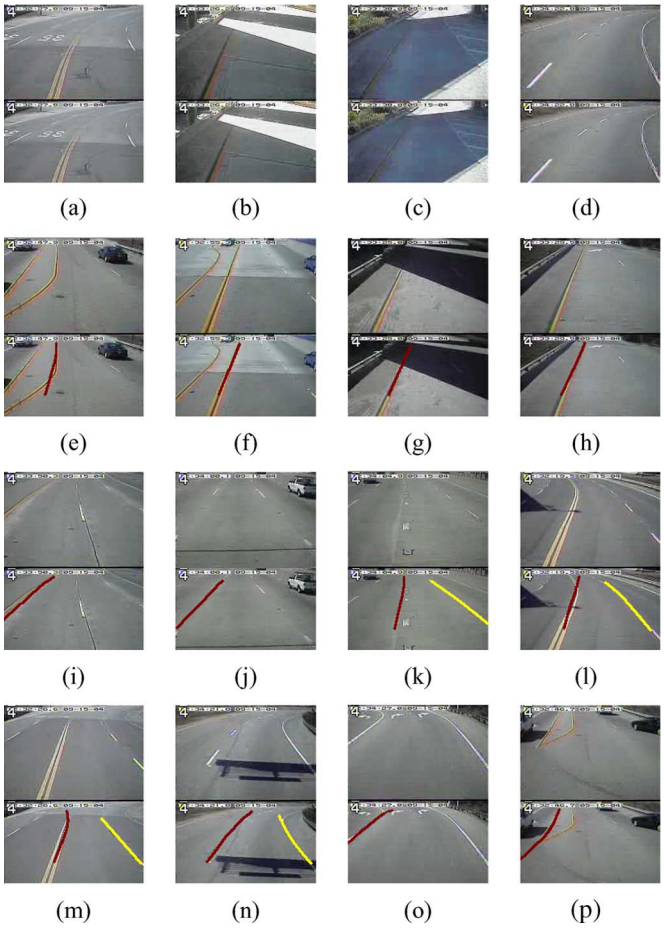

Fig. 13. (a)–(j) Example misdetections, (k)–(o) misalignments, and (p) a false alarm.

Fig. 14. Misalignment corrected by redundant detections in the following frames.

clip, thin red curves on the left are the selected hypotheses, and the thick orange and yellow curves are the detected left- and right-lane boundaries. The unprocessed video clip is also provided for future comparative studies.

Additional results on other challenging video clips (including the one used in the detection experiment) are shown in Fig. 15. It shows that the proposed algorithm works robustly on various distractions (competing nonlane-marking lines or leading vehicles) and on low illumination. Note that the right-lane boundaries of the low-illumination images are not lane markings but curbs. Whether curbs can be detected or not depends on the application—detecting a single lane boundary is sufficient in many applications, including the ones for collision warning. The result on these additional video clips can also be downloaded at the same website.

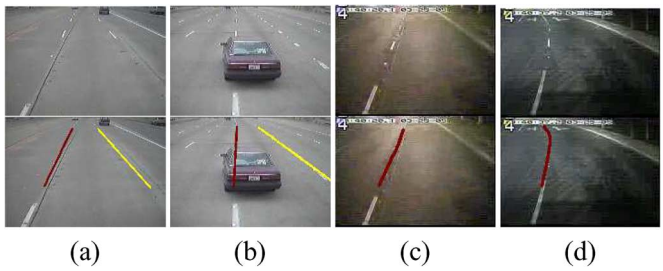

Fig. 15. Results on additional video clips. The proposed algorithm works robustly on challenging situations. Distraction by (a) a competing nonlane-marking line and (b) a leading vehicle and (c) and (d) low illumination.

TABLE II
MIS- OR FALSE-DETECTION COUNTS FROM 1951 FRAMES

Misdetection (both)	10 cases, 95 frames
Misdetection (one)	17 cases, 110 frames
False alarm / major misalignments	6 cases, 24 frames
Minor misalignments	3 cases, 6 frames

Fig. 16. Preliminary sensor-fusion result of vision-based lane-and-obstacle-detection results, LIDAR, and vehicle navigation information. The obstacles' relative positions with respect to the lane are estimated. The thick red line is a projected vehicle path.

VI. SUMMARY AND FUTURE WORK

We introduced a robust real-time lane-detection-and-tracking algorithm for local roads and freeways of various challenging scenarios. We first presented a comparative study on the lane-marking-classification performance and the computing cost. We introduced a robust real-time lane-detection algorithm based on RANSAC, and it was combined with a particle-filtering-based tracking algorithm by a probabilistic grouping framework. The suggested grouping framework can generally be applied to other part-based object detection and tracking problems.

The result was promising. Future work will integrate it with a vision-based obstacle-detection algorithm, for example [20], for a collision-warning system. A preliminary result is shown in Fig. 16. By applying a sensor-fusion algorithm to combine vision-based lane and obstacle detection results with other sensor information, the collision-warning performance can significantly be increased.

REFERENCES

[1] D. Pomerleau, "RALPH: Rapidly adapting lateral position handler," in *Proc. Intell. Veh. Symp.*, 1995, pp. 506–511.
[2] M. Bertozzi and A. Broggi, "Real-time lane and obstacle detection on the GOLD system," in *Proc. IEEE Intell. Veh.*, 1996, pp. 213–218.

[3] C. J. Taylor, J. Malik, and J. Weber, "A real-time approach to stereopsis and lane-finding," in *Proc. IEEE Intell. Veh.*, 1996, pp. 207–212.
[4] H. Hattori, "Stereo for 2D visual navigation," in *Proc. IEEE Intell. Veh. Symp.*, 2000, pp. 31–38.
[5] M. Beauvais and S. Lakshmanan, "Clark: A heterogeneous sensor fusion method for finding lanes and obstacles," *Image Vis. Comput.*, vol. 18, no. 5, pp. 397–413, Apr. 2000.
[6] S.-S. Ieng, J.-P. Tarel, and R. Labayrade, "On the design of a single lane-markings detectors regardless the on-board camera's position," in *Proc. IEEE Intell. Veh. Symp.*, 2003, pp. 564–569. [Online]. Available: http://www-rocq.inria.fr/ tarel/iv03.html
[7] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using B-snake," *Image Vis. Comput.*, vol. 22, no. 4, pp. 269–280, Apr. 2004.
[8] Y. Zhou, R. Xu, X. Hu, and Q. Ye, "A robust lane detection and tracking method based on computer vision," *Meas. Sci. Technol.*, vol. 17, no. 4, pp. 736–745, Apr. 2006.
[9] R. Labayrade, J. Douret, and D. Aubert, "A multi-model lane detector that handles road singularities," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2006, pp. 1143–1148.
[10] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[11] R. Collobert, S. Bengio, and J. Mariethoz, "Torch: A modular machine learning software library," Institut Dalle Molle d'Intelligence Artificielle Perceptive, Tech. Rep. IDIAP-RR 02-46 2002. [Online]. Available: http://www.torch.ch/
[12] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York: Springer-Verlag, 2001.
[13] Z. Kim and R. Nevatia, "Expandable Bayesian networks for 3D object description from multiple views and multiple mode inputs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 6, pp. 769–774, Jun. 2003.
[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.
[15] Z. Kim, "Geometry of vanishing points and its application to external calibration and realtime pose estimation," Inst. Transp. Stud., Res. Rep. UCB-ITS-RR-2006-5, Univ. Calif, Berkeley, CA, 2006.
[16] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.
[17] T. Dean and K. Kanazawa, "A model for reasoning about persistence and causation," *Comput. Intell.*, vol. 5, no. 3, pp. 142–150, Aug. 1989.
[18] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2003, vol. 2, pp. 264–271.
[19] Z. Kim and J. Malik, "Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, vol. 1, pp. 524–531.
[20] Z. Kim, "Realtime obstacle detection and tracking based on constrained Delaunay triangulation," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2006, pp. 548–553.

ZuWhan Kim (S'00–M'01) received the B.S. and M.S. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, and the Ph.D. degree from the University of Southern California, Los Angeles.

He is currently a Research Engineer with California Partners for Advanced Transit and Highways (California PATH), University of California, Berkeley. His primary research areas are in computer vision and pattern recognition. He has multidisciplinary research experience with a wide variety of academic fields, including intelligent transportation systems, geographic information systems, and cognitive science and robotics. He authored the vehicle detection and tracking algorithm for the Next Generation Simulation (NGSIM) program of the Federal Highway Administration.

Dr. Kim is a member of the IEEE Computer Society and Association for Computing Machinery.