Proceedings of the IEEE ITSC 2006
2006 IEEE Intelligent Transportation Systems Conference
Toronto, Canada, September 17-20, 2006

Realtime Lane Tracking of Curved Local Road

Zu Kim, *Member, IEEE*

Abstract— **A lane detection system is an important component of many intelligent transportation systems. We present a robust realtime lane tracking algorithm for a curved local road. First, we present a comparative study to find a good realtime lane marking classifier. Once lane markings are detected, they are grouped into many lane boundary hypotheses represented by constrained cubic spline curves. We present a robust hypothesis generation algorithm using a particle filtering technique and a RANSAC (RANdom SAmple Concensus) algorithm. We introduce a probabilistic approach to group lane boundary hypotheses into left and right lane boundaries. The proposed grouping approach can be applied to general part-based object tracking problems. It incorporates a likelihood-based object recognition technique into a Markov-style process. An experimental result on local streets shows that the suggested algorithm is very reliable.**

I. Introduction

Detecting and localizing lanes from a road image is an important component of many intelligent transportation systems applications. There has been active research on lane detection, and various algorithms have been suggested [1], [2], [3], [4], [5], [6], [7]. Due to a realtime constraint and then slow processor speed the lane markings have been detected based only on their intensity values or simple gradient changes, and many of their results were shown in straight roads and/or highways with clear lane markings or with an absence of obstacles on the road. Some exceptions of detecting curved roads are found in [5] and [7]. However, neither of the algorithms works in realtime and the results are shown on a limited scenarios with one or no obstacles present.

We present a realtime lane detection algorithm which shows robust results on various lane marking types and various difficult scenarios including lane changes, emerging/ending/merging/splitting lanes, intersections, and distractions by leading vehicles and markings on the road.

The lane curvature information is especially useful for a collision warning system. A collision warning system takes readings from various sensors, such as radar, LIDAR, and accelerometer, detects objects in front of the vehicle, estimates the collision times to the detected objects, and generates warnings for the driver. A difficulty occurs when there is a curved road in front. Without knowing the road curvature, the system cannot distinguish an object on a road from one on a sidewalk. One false alarm scenario is shown in Fig. 1. Without the information on the road curvature, the system may generate a false alarm for the postbox on the

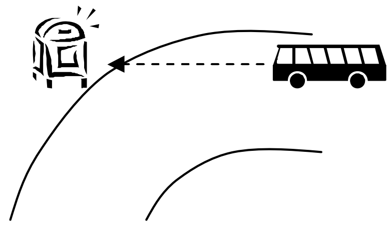
Fig. 1. A false alarm scenario of a collision warning system. Without knowing the lane curvature, the system will generate a false alarm for the postbox.

sidewalk. Our goal is to detect boundaries of the lane and to estimate their rough curvatures.

One possible difficulty in applying vision-based lane detection result to a collision warning system is that a small calibration error (a change of camera's pitch angle) can cause a large error in distance [8]. However, this is not a critical problem. For example, we can apply a sensor fusion technique of combining vision-based obstacle detection and active-sensor based obstacle detection to deal with it [9].

As an alternative to a vision-based approach, one may use a global positioning system (GPS) and 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, the curve at an off-ramp can cause false warnings but the GPS-based systems suffer from discriminating whether the vehicle entered an off-ramp or not.

There are several technical challenges of the vision-based lane detection and tracking:

- The algorithm should work in realtime.
- When we introduce the road curvature, the search space becomes quite large, and robust and fast detection is difficult.
- We deal with a local road, while much of the previous work has focused on highways with regular lane markings and a small curvature. There are various types of roads and lane markings, and a simple intensity- or gradient- based lane marking detection algorithm may not work.
- There are many distracting features such as traffic directives written on the road, as well as leading vehicles.
- Many of the previous approaches apply constant-width lane models. However, they cannot effectively deal with various situations such as an emerging lane, merging lanes, and splitting lanes at on/off ramps.

Our algorithm follows the "hypothesize and verify" paradigm. In the "hypothesize" step, lower level features are

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

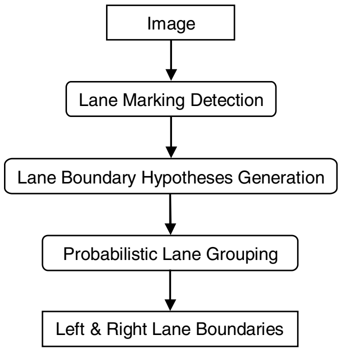
Image
Lane Marking Detection
Lane Boundary Hypotheses Generation
Probabilistic Lane Grouping
Left & Right Lane Boundaries

Fig. 2. The flow diagram of the algorithm.

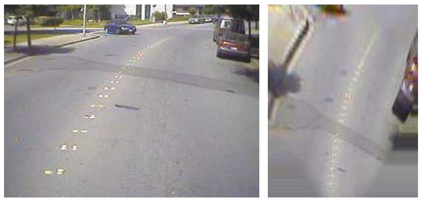

Fig. 3. An example image and the rectified image.

grouped into many higher level feature hypotheses, and they are filtered in the "verify" step to reduce the complexity of the higher level grouping. Figure 2 shows the flow diagram. First, the image is *rectified* assuming that the ground is flat. An example image and the rectified image are shown in Figure 3. In the rectified image, possible lane marking pixels are detected. Detected lane marking pixels are, then, 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 RANSAC (RANdom SAmple Consensus) algorithm (for detection) is introduced to robustly find lane boundary hypotheses. 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 a uniform width) to deal with various situations such as on/off ramps or an emerging lane.

Section II describes our lane marking detection approach including a comparative study on various classification methods. There is a lack of such comparative study in realtime vision system research. 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 conclusion in Section VI.

II. Lane Marking Detection

Sample road images are shown in Figure 4. Most previous algorithms simply look for "horizontal intensity bumps" to detect lane markings. It shows reasonably good performance in many cases, but there are difficult situations when the lane markings are not clearly visible (for example, yellow

Fig. 4. Example road images.

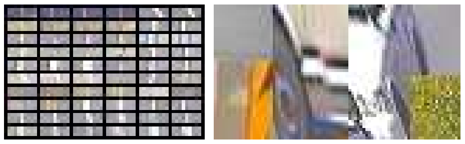

Fig. 5. Example image patches of lane markings and non-markings.

center markings have similar grayscale intensity to the gray road pixels) or when the image quality is poor. It also suffers from false detections by leading vehicles and textures on road and sidewalk. To find a better (but fast enough) classification algorithm, we apply a machine learning technique on a larger number of features: the RGB values of the pixels in 9×3 windows (total 81 features). Applying a stereo algorithm, [3], [4], can further improve the performance, but we focus on a monocular color image in this paper.

From a video sequence, we have gathered image patches of 565 lane markings and 11893 non-markings. Figure 5 shows example image patches. We observe a variety in colors, textures, and width.

We compare the classification performance and the computation requirement of various classifiers on our data set. We compare the following classifiers:

- **Artificial Neural Networks (ANN):** We compare two-layer neural networks with various numbers of hidden nodes. Training an ANN requires relatively significant computation, but the actual classification time is relatively small. When there are n features (inputs) and m hidden nodes, it requires $n+m$ weighted summation and m sigmoid function calculation to classify a hypothesis ($n = 81$ in our case).
- **Perceptron:** Perceptrons provide the simplest and fastest classification. The classification is performed by linear weighted summation and the parameters are learned by an iterative (but fast) learning algorithm.
- **Naive Bayesian Classifiers (NBC):** Naive Bayesian classifiers show good classification performances in spite of its unrealistic conditional independence assumption. We compare the discrete and the unimodal Gaussian representations of the conditional probability. For the 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 to compute the exponential function for n times. However, we can avoid calling the exponential function by using log likelihood instead of the actual probability. In fact, for both representations, it is necessary to use the log likelihood to minimize the numerical errors, especially when the number of features is large. For the discrete

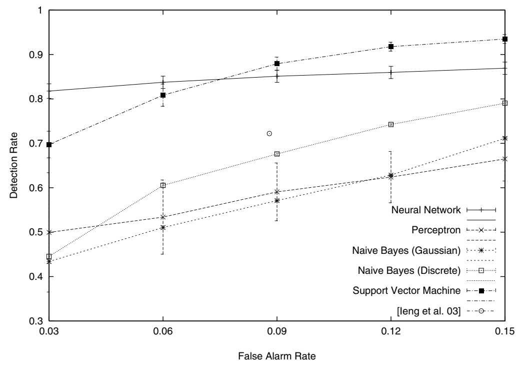

Fig. 6. The classification performance of the classifiers.

NBC, we can pre-calculate the logarithms of all the probability table entries to save the computation.

- **Support Vector Machine (SVM):** During the last decade, support vector machines have rapidly gained popularity. They provide a good framework for incorporating kernel methods. We test the second order polynomial kernel, which requires the smallest computation. Learning requires relatively significant computation, but it is bounded in polynomial time. The classification invovlves a large number of multiplications ($O(mn)$ where m is the number of support vectors). The number of support vectors which is at least $n+1$. However, when the data is not clearly separable (in the transformed feature space) or when a small tuning parameter is given, the number of support vectors can be much greater than n.
- **Ieng et al. 2003:** In [6], Ieng et al. suggested a fast and simple lane marking detection algorithm. It simply finds the intensity bump of a pre-defined pixel width, which is reported to show a good performance.

Details on most of the above classifiers can be found in many of the machine learning literatures, for example, in [10].

Figure 6 shows the classification performances of the presented classifiers. We follow the evaluation scheme presented in [11]. We repeated stratified 5 fold cross-validation for 10 times, and show the ROC curves with the confidence intervals. For all the classifiers, we obtained the ROC curves by changing only the threshold values (no re-learning with different parameters).

For all the classifiers, we applied various parameters, and chose the best ones. For ANN, we compared the ones with 5 and 15 hidden nodes. For the discrete naive Bayesian network, we used 7-level discretization. The SVM was learned with the tuning parameter 100.0. We see that the support vector machine and the neural networks show the best performances.

We also compared the classification computation for the classifiers. We have applied the classifiers on 52×159 images, and summarized the computing time in Table I. The algorithms ran on a Pentium III 1GHz machine. We optimized all the classification algorithms to bring maximum performance. Our decision based on the presented result is to use an ANN with 5 hidden nodes which showed a good classification performance with small computation.

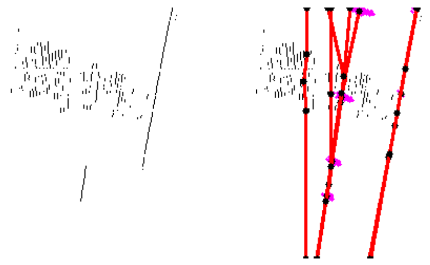

Fig. 7. (a) Detected lane marking pixels, and (b) the *selected hypotheses* from the particle filtering/RANSAC algorithm.

TABLE I
COMPUTATION TIME OF THE CLASSIFIERS.

Classifier	Classification time
ANN (5 hidden nodes)	60 ms
ANN (15 hidden nodes)	180 ms
Perceptron	10 ms
Gaussian NBC	60 ms
Discrete NBC	70 ms
SVM	1.2 sec
Ieng et al.	Under 10 ms

III. LANE BOUNDARY HYPOTHESES GENERATION WITH PARTICLE FILTERING AND RANSAC

Once possible lane marking pixels are detected (an example is shown in Figure 7a), they are grouped into cubic spline curves of 5 control points. A spline is a smooth piecewise polynomial function, which is widely use to represent a curve. Various spline representations have been proposed and we use a cubic spline. In a cubic-spline representation, a point p on a curve between i-th and $(i+1)$-th control point is represented as:

$$\begin{aligned} p &= (x_i(t), y_i(t)), \text{where} \\ x_i(t) &= a_i + b_i t + c_i t^2 + d_i t^3, \\ y_i(t) &= e_i + f_i t + g_i t^2 + h_i t^3, \end{aligned}$$

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

A cubic-spline curve has a useful property for robust fitting that the control points are actually on the curve. We use this property to apply a RANSAC (RANdom SAmple Concensus) algorithm [12] to curve fitting. A RANSAC algorithm is a robust fitting algorithm that has been successfully 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, cubic spline fitting is much faster, but it suffers from unexpectedly irregular shapes, in general. Our algorithm uses a cubic spline for fast fitting, but imposes additional constraints that $y_i(t)$ is monotonic and the curvature be reasonable.

Our RANSAC fitting procedure is as follows. First, lane markings are grouped into line segments. A hypothesis is generated from a random set of one, two, or three line segments. The number of the line segments is also determined randomly for each hypothesis. A single line segment gives a straight line, a set of two line segments gives a curve with a single curvature (roughly), and a set of three line segments gives a more complicated curve. Three of the five control points are determined from these line segments. The positions of the first and the last control point are determined by extrapolating the first and the last line segment. The first control point is always on the lower boundary of the rectified image, and the last control point is the upper-most lane marking pixel along the extrapolated curve.

Once a curve hypothesis is generated, its validity is tested. First, a simple geometrical test is performed to see whether the curve is aligned with the original line segment. Then, the second test is performed based on its lane marking pixel support:

$$\text{CurveScore} = (1-\lambda)\sum_{m} 1,$$

where m is a supporting lane marking pixel on the curve, and λ is a penalty for the minimum description length (MDL) criteria. In our implementation, $\lambda = 0.1$ for a hypothesis generated from a set of two line segments and $\lambda = 0.2$ for a hypothesis from three line segments.

In our implementation, 500 random hypotheses are generated given a frame. Once hypotheses are generated and validated, an overlap analysis is performed and a small number of non-overlapping (but may be partially overlapping) hypotheses are finally selected.

The above algorithm does not take advantage of a strong temporal coherence of the lane boundaries. We combine it with a tracking algorithm based on a particle filtering technique. For particle filtering, we model vehicle's motion (rotation and translation) by Gaussian distributions. Given a randomly selected motion, a lane boundary hypothesis is generated by moving the previously detected lane boundary's control points according to the motion. The position of the new control point is slightly adjusted when there is a lane marking pixel nearby. A number of hypotheses are generated (100 per each left and right boundaries, in our implementation) and scored based on the supporting lane marking pixels.

The final position of a control point is estimated by the weighted sum over all the hypothesized positions, where the weights are the scores of the corresponding curve hypotheses. Due to the vehicle's forward motion, all the control points will eventually move closer. Therefore, we need to adjust the position of the first control point that it does not disappear from the image. We enforce that the first control point's position be always on the boundary of the image (the adjusted position is found by interpolating the curve). For the last control point, a search is performed on the extrapolated curve position. If there is any lane marking pixel above the currently determined last control point, it is selected as the new last control point. In addition, the second control point is also examined to see if its position is too low (if it keeps going down, it will eventually collide with the first control point). If its position is too low, it is removed and a new control point is generated between the fourth and the fifth (last) control point.

The *selected lane boundary hypotheses* from the RANSASC algorithm and the particle filtering process are shown in Figure 7b. Up to five hypotheses per lane boundary (left/right) are selected including the ones from the particle filtering process. The particles are also shown as clouds of points nearby the control points.

IV. Probabilistic Grouping of Lane Boundaries

Our next goal is to choose the best pair from the *selected hypotheses* shown in Figure 7b. We apply probabilistic reasoning for decision making. The available evidence for the reasoning includes the lane marking support of each lane boundary hypothesis (see Section III), compatibility of the two boundary hypotheses, and the temporal coherance. In fact, this is a typical object tracking problem. This one is focused on part-based tracking while most of the previous work focused on a single part tracking. In this section, we introduce a new formulation for object tracking which fits especially 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, x, to denote certain assignments taken by those variables. We mostly deal with binary classification, where the assignments are true or false. For a multi-nomial random variable X, the statement $P(X = \phi)$ is used as a shorthand for $P(X = \text{missing})$. A set of multiple variables or assignments are denoted by boldface letters, such as $\mathbf{X}$ and $\mathbf{x}$.

Typical temporal reasoning models, such as dynamic Bayesian networks [13], use a posterior probability to select the best hypothesis from pre-defined candidates:

$$\arg\max_{x} P(X = x|\mathbf{e}), \tag{1}$$

where X is the target random variable and $\mathbf{e}$ is observed evidence (current and past).

However, many of the object recognition approaches use the maximum likelihood estimate [14], [15]. Their goal is to find:

$$\arg\max_{X} P(X = \text{true}|\mathbf{e}), \tag{2}$$

where X is a random variable for a specific hypothesis. 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 pre-determined) number of 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 such that selecting one does not exactly mean rejecting another. In addition, it is much easier to model a binary classification problem than a multinomial classification one especially for learning.

We introduce a dynamic Bayesian network-style formulation using the likelihood estimates. 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 $P(\mathbf{x}|\mathbf{e_c}, \mathbf{e_t}, \mathbf{e_p})$ for all possible combinations of the part hypotheses, $\mathbf{x}$. Then,

$$P(\mathbf{x}|\mathbf{e_c}, \mathbf{e_t}, \mathbf{e_p}) = \alpha P(\mathbf{e_c}|\mathbf{x}) P(\mathbf{x}|\mathbf{e_t}, \mathbf{e_p}), \tag{3}$$

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

To estimate $P(\mathbf{e_c}|\mathbf{x})$ given an image frame is well-studied in the object recognition field. The challenge is to estimate $P(\mathbf{x}|\mathbf{e_t}, \mathbf{e_p})$. Applying the Markov assumption,

$$P(\mathbf{x}|\mathbf{e_t}, \mathbf{e_p}) = \sum_{\mathbf{h}} P(\mathbf{x}|\mathbf{H} = \mathbf{h}, \mathbf{e_t}) P(\mathbf{H} = \mathbf{h}|\mathbf{e_p}), \tag{4}$$

where $\mathbf{H}$ is a random variable for the previous object and $\mathbf{h}$ represents an individual hypotheses ($\Sigma_{\mathbf{h}} P(\mathbf{H} = \mathbf{h}) = 1$)). 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 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 a 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 is the likelihood of the hypotheses, not $P(\mathbf{H} = \mathbf{h}|\mathbf{e_p})$. To deal with this, we estimate $P(\mathbf{H} = \mathbf{h}|\mathbf{e_p})$ from the previous likelihood:

$$P(\mathbf{H} = \mathbf{h_i}|\mathbf{e_p}) = \frac{P(\mathbf{H_i} = \text{true}|\mathbf{e_p})}{\sum_j P(\mathbf{H_j} = \text{true}|\mathbf{e_p})}, \tag{5}$$

where $\mathbf{H_i}$ is for a previously *selected hypothesis* (cf. Section III). We also consider the probability of mis- or false detection of some or all of the object. Therefore, $\mathbf{H_i}$ may represent mis-detection.

For $P(\mathbf{x}|\mathbf{H} = \mathbf{h_i}, \mathbf{e_t})$, we assume that the individual parts' tracking histories are independent to each other. When $\mathbf{x}$ (as well as $\mathbf{H_i}$) consists of individual parts $x^1, \ldots, x^n$,

$$P(\mathbf{x}|\mathbf{H_i} = \mathbf{h_i}, \mathbf{e_t}) = \prod_j P(x^j|H_i^j = h_i^j, \mathbf{e_t}). \tag{6}$$

When h_i^j is not a mis-detection,

$$\begin{aligned} P(x^j|H_i^j = h_i^j, \mathbf{e_t}) &= P(x^j|h_i^j = \text{true}, \mathbf{e_t}) \\ &= \beta_j P(\mathbf{e_t}|x^j, h_i^j) P(x^j|h_i^j), \end{aligned} \tag{7}$$

where β_j is a normalizing constant, $P(\mathbf{e_t}|x^j, h_i^j)$ can be learned by examples, and $P(x^j|h_i^j)$ is assumed to be constant (prior probability given that there is a previously detected part). When h_i^j represent a mis-detection ($H_i^j = \phi$), $P(x^j|H_i^j = \phi, \mathbf{e_t})$ is assumed to be a constant, which is the prior probability of an emerging part.

In addition, we also need to know the probability of the j-th part of the object being mis-detected in the current frame (denoted by ϕ_j). In this case, we need to separately estimate $P(\phi_j|H_i^j = h_i^j, \mathbf{e_t})$, which is set to constant – either $P(\phi_j|H_i^j = \phi)$ or $P(\phi_j|H_i^j = \text{detected})$.

B. Application to Lane Boundary Grouping

In the lane boundary grouping case, $\mathbf{X} = (L, R)$, where L is for the left and R is for the right lane boundary hypotheses. For $P(\mathbf{e_c}|l, r)$, 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 the lane marking support score introduced in Section III. For $\mathbf{e_{lr}}$, we examine the average lane width and the *lane compatibility score*. In fact, the lane width can be slightly increasing or decreasing (within an image) due to the presence of up-/down-hills or changing road width. We assume that the lane width can be linearly 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. The *lane compatibility score* is inverse proportional to the maximum residual distance.

For $P(\mathbf{e_t}|l, l_{\text{prev}})$ and $P(\mathbf{e_t}|r, r_{\text{prev}})$ the maximum distance between a curve and a sampled point of the other curve is used.

Note that there is no hidden variable and all the probability distribution (including the background models, such as $P(\mathbf{e_c}|L = \text{false})$) can be directly learned from positive and negative examples. We assumed that most of the above conditional probability distributions follow a unimodal Gaussian model. One exception is on the lane width. To enforce a minimum and maximum lane width we use a discrete distribution instead of a Gaussian one.

It is very tedious to manually give a large number of positive and negative examples. Therefore, a semi-supervised 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. Rejected hyoptheses are considered as negative examples.
3) The lane detection algorithm is once again applied with the new parameters.

V. Experimental Results

Most of the previous work present the experiments only on a small number of "demo" images. In [7], the result is shown on a larger number of image frames, but on a specific scenario where the lane is detected in the first frame and tracked throughout the entire sequence. In contrast, we present experimental results on two recorded video clips (MPEG) of 176×120 image resolution. The total computation was about $30 \sim 70$ms per frame on an Intel® Pentium® M 2GHz processor, and we ran the algorithm for 10 frames per second. The computation is similar on a higher resolution video (352×240) because most of the computation is done on the fixed sized rectified image. However, increased resolution will increase the performance, in general, because we can obtain a rectified image of a better quality. Currently, a Pentium® M processor of up to 1.8GHz is available for embedded computing (PC/104).

The first video clip was 1 minute and 39 seconds in length (about 1000 frames) and in over 600 frames of them one or two lane boundaries were visible. The second video clip was 2 minutes and 8 second long. The video clips include various different lane types, emerging/ending/merging lanes, intersections, lane changes, significant obstacles (leading vehicles), and distracting markings on the road. In addition, the image resolution and quality (quatersize of MPEG) is generally poorer than those have been used in the previous work.

Example results are shown in Figure 8 and Figure 9. No noticeable false alarms were detected in the first video clip. The robust lane marking detection procedure removed many of the false lane markings, and the probabilistic grouping procedure removed many of the false hypotheses which did not have enough temporal correlation.

Some detection failures were reported due to a lack of lane marking support. In the first video clip, single lane boundaries were temporarily misdetected in 8 cases of total 23 frames and both of the lane boundaries were misdetected for one case of total 32 frames (in 20 frames of them, only a single lane boundary was visible).

The overall detection rate is similar to that of a state-of-the-art non-realtime algorithm [7]. Taking into account that our example contains much more complicated scenarios including intersections, lane changes, and obstacles, this is a very promising result.

In the second video clip, two false alarm cases were generated by competing lane hypotheses, and one false tracking was caused by a leading vehicle and its shadow cast on a lane boundary. In the both of the false alarm cases, the correct hypotheses were temporarily rejected due to weaker lane marking supports and temporary misdetection of the right lane boundaries. In addition, one significant misconfiguration (detecting a curve as a straight line) and several cases of slight misconfiguration were reported. All the false alarm cases and the misconfiguration cases are shown in Figure 9.

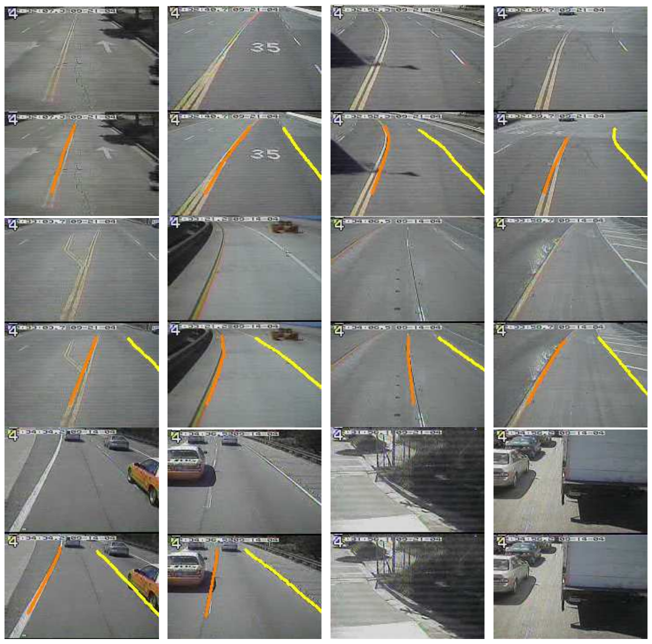
Fig. 8. Example detection results.

The suggested algorithm also work on video images with low illumination. Preliminary results on dark video images are shown in Figure 10. Note that the same parameters as for the above result were used. More misdetections were observed in this case since the lane marking classifier was learned with daytime video images. However, it still shows the robustness of our approach. The mistdetection rate can be significantly reduced by re-learning the lane marking classifier with dark video images.

The resulting video clips can be downloaded at http://path.berkeley.edu/∼zuwhan/lanedetection. 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 raw video clips are also provided for future comparative studies.

VI. Conclusion

We introduced a robust realtime lane detection algorithm for curved local roads. We first presented a comparative study on the lane marking classification performance and the computing cost. Our grouping algorithm is based on the "hypothesize and verify paradigm. We introduced a novel approach to combine lane detection and tracking based on the particle filtering and RANSAC technique. We also presented a probabilistic grouping approach for part-based object tracking which can incorporate a likelihood-based object recognition algorithm into a Markov framework. A promising result was presented. Our algorithm showed a

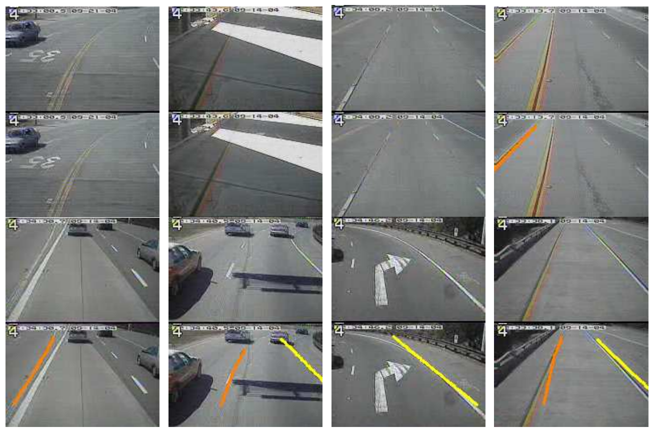

Fig. 9. Example misdetection and false alarms. Only three cases of significant false alarms or tracking failure were reported in two video clips of about 2200 total frames.

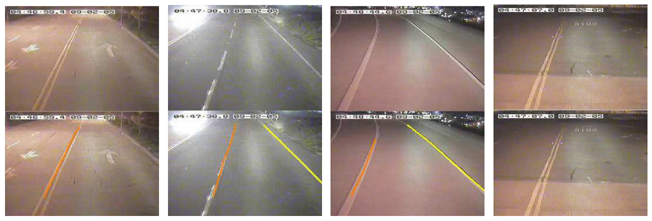

Fig. 10. Example results on images with low illumination. The right most on is a misdetection case.

comparable result to a state-of-the-art non-realtime algorithm on video clips of more complicated scenarios. The future work will be applying the suggested tracking and grouping framework to other part-based tracking applications such as human body tracking.

REFERENCES

[1] D. Pomerlean, "RALPH: Rapidly adapting lateral position handler," in *Proceedings of the Intelligent Vehicles 1995 Symposium*, 1995, pp. 506–511.

[2] M. Bertozzi and A. Broggi, "Real-time lane and obstacle detection on the GOLD system," in *Proc. IEEE Intelligent Vehicles*, 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 Intelligent Vehicles*, 1996, pp. 207–212.

[4] H. Hattori, "Stereo for 2d visual navigation," in *Proc. IEEE Intelligent Vehicles Symposium*, 2000, pp. 31–38.

[5] M. Beauvais and S. Lakshmanan, "Clark: a heterogeneous sensor fusion method for finding lanes and obstacles," *Image and Vision Computing*, vol. 18, no. 5, pp. 397–413, April 2000.

[6] S.-S. Ieng, J.-P. Tarel, and R. Labayrade, "On the design of a single lane-markings detector regardless the on-board camera's position," in *Proc. IEEE Intelligent Vehicles Symposium*, 2003, pp. 564–569, 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 and Vision Computing*, vol. 22, pp. 269–280, 2004.

[8] "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.

[9] Z. Kim, "Realtime obstacle detection and tracking based on constrained delaunay triangulation," in *IEEE Intelligent Transportation Systems Conference*, 2006.

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2001.

[11] 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.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. of the ACM*, vol. 24, pp. 381–395, 1981.

[13] T. Dean and K. Kanazawa, "A model for reasoning about persistence and causation," *Computational Intelligence*, vol. 5, no. 3, pp. 142–150, 1989.

[14] 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.

[15] Z. Kim and J. Malik, "Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking," in *Proc. IEEE Intl. Conf. Computer Vision*, vol. 1, 2003, pp. 524–531.