“© 2007 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

Robust Lane Detection in Urban Environments

Stephan Sehestedt, Sarath Kodagoda, Alen Alempijevic, Gamini Dissanayake

***Abstract*— Most of the lane marking detection algorithms reported in the literature are suitable for highway scenarios. This paper presents a novel clustered particle filter based approach to lane detection, which is suitable for urban streets in normal traffic conditions. Furthermore, a quality measure for the detection is calculated as a measure of reliability. The core of this approach is the usage of weak models, i.e. the avoidance of strong assumptions about the road geometry. Experiments were carried out in Sydney urban areas with a vehicle mounted laser range scanner and a ccd camera. Through experimentations, we have shown that a clustered particle filter can be used to efficiently extract lane markings.**

I. INTRODUCTION

Today we observe an increasing demand for traffic safety systems to minimize the risk of accidents. There are a large number of vision based systems for lateral and longitudinal vehicle control, collision avoidance and lane departure warning, which have been developed during the last decade around the world (some examples are [1], [2], [3] and [4]). Recently announced DARPA Urban Grand Challenge is yet another proof of enthusiasm in autonomous urban driving.

The development of advanced driver assistance systems and ultimately autonomous driving requires the ability to analyse the road scene. One prerequisite for this is the extraction of lanes and lane markings. This information is essential in order to obey traffic rules and to detect possible hazards.

The presented work aims on a novel approach to robustly extract lane markings in urban traffic scenarios. Former approaches concentrated on highway like scenarios, where the traffic situations are less complex and markings are generally very visible [5], [6], [7]. However, many of the proposed methods are not suitable for urban traffic.

A number of different constraints are commonly used to be able to detect and track lane markings, such as lanes being straight [8] or only slightly curved [6]. Such an assumption holds for freeways but is certain to fail in urban areas. Furthermore, geometric models were applied [6], [5], which describe the shape of a lane. These models rely on the visibility of markings. However, in urban areas we often have to deal with occlusions and bad or missing markings over extended periods of time.

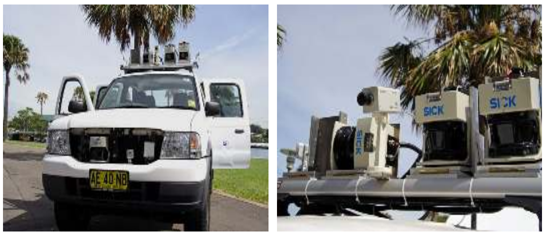

(a) (b)

Fig. 1. a) The testbed vehicle and b) the sensors mounted on the roof.

As outlined, strong assumptions about the lane geometry must be expected to be violated. Therefore, we concluded that the use of weaker models is necessary. In this work we use a particle filter, which particles move from the bottom to the top of the 2D image plane of a inverse perspective mapped image. Each sample represents the possible position of a piece of lane marking and the according weights indicate the probability of this.

Although we are using this filter in a software to actually track lanes, here we concentrate on the difficult task of detection. Tracking is then within the scope of a following publication.

Data Collection and testing is done on our research vehicle shown in Fig. 1. For this work we use a vertically mounted laser scanner and a colour ccd camera, both located on the roof of the car. Runs are undertaken in Sydney's urban area.

The remainder of this publication is organised as follows. In section II we briefly explain the basic particle filter to then present our implementation of a clustered particle filter as we use it for lane marking tracking. In section III we present our novel approach to track lane markings in an image stream in real time. Furthermore, in IV we present experimental results with real world data. Finally, future work and the conclusions are discussed in sections V and VI.

II. PARTICLE FILTER

Monte Carlo Methods or Particle Filters allow to approximate arbitrary multimodal probability distributions recursively. To estimate the state of a system a set of samples X at time t is used. This set $X_t = \langle x_t^i \mid i = 1...N\rangle$ and it's associated weights ω_t^i represent the belief at time t. The weights are computed according to a sensor model, which contains the information of how likely it is that a

---

This work is supported by the ARC Centre of Excellence program, funded by the Australian Research Council (ARC) and the New South Wales State Government.

The authors gratefully acknowledge the partial funding of this work by the German Academic Exchange Service (DAAD) with a PhD student scholarship.

Stephan Sehestedt is with the ARC Centre of Excellence for Autonomous Systems, Sydney, Australia (e-mail: s.sehestedt@cas.edu.au)

Sarath Kodagoda is with the ARC Centre of Excellence for Autonomous Systems, Sydney, Australia (e-mail:s.kodagoda@ cas.edu.au)

Alen Alempijevic is with the ARC Centre of Excellence for Autonomous Systems, Sydney, Australia (e-mail:a.alempijevic@cas.edu.au)

Gamini Dissanayake is with the ARC Centre of Excellence for Autonomous Systems, Sydney, Australia (e-mail:g.dissanayake@cas.edu.au)

TABLE I

THE THREE STEPS OF THE PARTICLE FILTER ALGORITHM

1) Prediction: Draw $x_t^i \sim p(x_t^i \mid x_{t-1}^i, u_{t-1})$.
2) Update: Compute the importance factors $\omega_t^i = \eta p(y_t \mid x_t^i)$, with η being a normalization factor to ensure that the weights sum up to 1. Here, y_t is a sensor reading at time t.
3) Resample.

sample x represents the true state. The computation of the posterior is then done in three steps: 1. Prediction. 2. Update 3. Resampling.

The additional resampling step ensures that the resources, i.e. the particles, are concentrated in areas of high probability. Thus, the samples are used in the areas of interest. However, due to this step the particle filter also tends to converge to one state, which means that in the basic implementation this filter would not be suitable to track multiple hypotheses over extended periods of time. Clearly in our application we need to able to track multiple lane markings, also without prior knowledge of how many.

For a more wholesome summary of Monte Carlo methods refer to [12], [11].

A. Clustered Particle Filters

In order to be able to track multiple lane markings we apply a clustered particle filter, similar to what is presented in [13]. There the sample set is divided into clusters that each represent one hypothesis. A cluster is divided into two clusters when a subset of samples has a certain distance to the others and if this subset has a high average weight. If the clusters get near to each other, they get fused to one single set.

This approach uses a modified proposal distribution, where we take account of the existance of multiple hypotheses. According to [12], the weights are calculated as

$$\omega_t^i = \eta \frac{p(y_t \mid x_t^i) p(x_t^i \mid x_{t-1}^i)}{q(x_t^i \mid x_{t-1}^i, y_t)} \tag{1}$$

η is a normalizing constant, which ensures that the weights sum up to 1. In that way, ω_t^i is only dependent on x_t^i, y_t, and y_{t-1}. To take account of the existence of clusters, the normalizing constant η is now dependent on the individual clusters and becomes η_i and consequently the weights are now computed as

$$\omega_t^i = \eta(x_t^i) p(y_t \mid x_t^i) \tag{2}$$

For C_t^j being the sum of weights of the jth cluster

$$\eta(x_t^i) = 1/C_t^j \tag{3}$$

This method is known in statistics literature as two stage sampling. The application to mobile robot localization is described in [13].

B. Adaptive Particle Filter

Furthermore, we apply KLD-Sampling, as introduced by Dieter Fox [14]. This is a technique for adjusting the sample set to a size that ensures a certain minimum quality of the approximation of the posterior. This improves the the overall performance significantly by adding robustness and reducing the computational burden.

III. LANE MARKING TRACKING

In this section our approach to lane marking tracking is presented. Furthermore, we outline all the background necessary to implement the proposed method.

A. Principle

Detection of lane markings in image data is not a trivial task. That is mainly due to the non existence of unique models, poor quality of lane markings due to wear, occlusions due to the presence of traffic and complex road geometry.

This leads us to the conclusion that any method using a too strong model of the road (lane), will fail eventually. Thus, weak models must be applied. As a result of this, particle filters are a good choice for the task of lane marking tracking due to their ability to handle poor process models.

The idea of the proposed method is to use an inverse perspective mapped image (IPM image) to run a particle set from the bottom to the top and observe the presence of lane markings in each line. Furthermore, we make sure that the filter is able to track multiple lines and to store each estimated line as a trail. In this way we produce a correct data association, e.g. we associate every detected piece of lane marking to one trail, which then represents the marking of one lane.

Various issues need to be solved to apply this method. A clustering routine needs to be implemented which allows to detect a number of lane markings at any time. We need to be able to split clusters into subclusters to correctly detect markings in special situations like the appearance of an additional lane, where a line splits up into two lines. Above this, we want to be able to handle high degrees of curvature also in the presence of broken markings with large gaps.

The basic principle is illustrated in Fig. 2. 2(a) shows the original image from which the IPM image (2(b)) is computed. In Fig. 2(b) we do not have any prior knowledge about the lane markings and therefore the filter is initialized with a uniform distribution. As the particles move up in the image (according a process model), the filter eventually converges to the position of the marking. In Fig. 2(c) and 2(d) the filter tracks the lane marking correctly.

In the following we will present the single components of our approach, which are the process model, the observation model and clustering. Above this, we present the usage of an uncertainty measure for the estimate of a marking.

B. Inverse Perspective Mapping

Lane detection is generally based on the localization of a specific pattern (the lane markings) in the acquired image and can be performed with the analysis of a single still image.

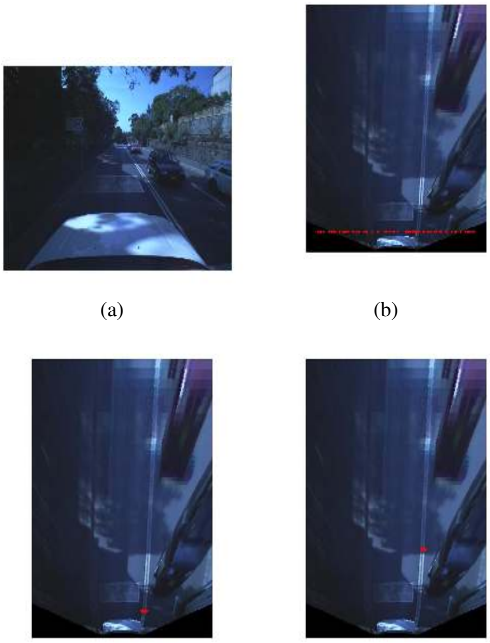
The method for lane marking detection employed in our paper is based on the Generic Obstacle and Lane Detection (GOLD) implementation.

In order to fit the lane model to the acquired road images geometrical image warping is performed with Inverse Perspective Mapping (IPM) The IPM technique projects each pixel of a 2D perspective view of a 3D object and re-maps it to a new position constructing a new image on an inverse 2D plane. Conversely, this will result in a bird's eye view of the road, removing the perspective effect. Each pixel represents the same portion of the road, allowing a homogeneous distribution of the information among all image pixels. To remove the perspective effect, it is necessary to know a priori the specific acquisition conditions (camera position, orientation, optics) and the scene represented in the image (the road, which is now assumed to be flat) . Mathematically, IPM can be described as a projection from a 3D Euclidean space $W = (x, y, z)$ onto a planar 2D space $I = (u, v)$.

$$u(x, y, 0) = \frac{arctan(\frac{y}{x}) - \theta - \alpha}{2\alpha(n-1)^{-1}} \tag{4}$$

$$v(x, y, 0) = \frac{arctan(\frac{h}{\sqrt{x^2+y^2}})}{2\alpha(n-1)^{-1}} \tag{5}$$

Where θ is the angle between the projection of the optical axis on the flat plane, 2θ is the camera angular aperture, n being the camera resolution and h the viewpoint of the camera C above the ground as can be seen in Fig. 3.


(a) (b)

(c) (d)

Fig. 2. a) The original image. b) Initialization phase, the particles (in red) are uniformily distributed. c) and d) The filter converges and then follows the marking

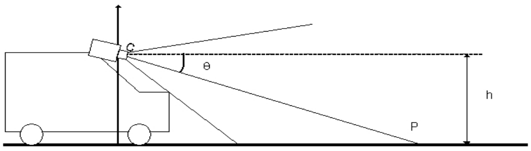


Fig. 3. The IPM Parameters

Furthermore, x and y are points in world co-ordinate system and u and v are points in the image co-ordinate system.

The mapping (4 and 5) is derived using triangulation and trigonometry of the IPM model [15].

The remapping process defined by (4 and 5) is implemented by scanning the array of pixels of coordinates which form the remapped image in order to associate to each of them the corresponding value assumed by the point of coordinates. The resolution of the remapped image has been chosen as a trade-off between information loss and processing time. At the moment we are using the blue channel of an rgb image only. C1C2C3 space gives better results but at the cost of computational effort.

As mentioned above, the mapping assumes that the road is flat. However, in due to non flat road profile and vehicle pitch this assumption is in most cases not true. Therefore, we use a vertically mounted laser to correct for the error. The result can be seen in figure, where the left image shows the uncorrected ipm image and the right one the corrected version.

Lane marking detection is performed on the remapped image applying a lane model which stipulates that a road marking is represented by a predominantly vertical bright line (lane marking) of constant width surrounded by a darker region (the road). Thus, the pixels belonging to a road marking have a brightness value higher than their left and right neighbours at a given horizontal distance. A vertical edge in an image conforms similarly to the same principle; the intensity difference between neighbouring pixels must be over a threshold to be validated. Therefore, an exhaustive search across each row of the image will produce potential lane marking candidates where the match probability can be measured with the edge quality (difference of intensity).

This result is exploited as a sensor model in this work, because this method produces high quality observations also

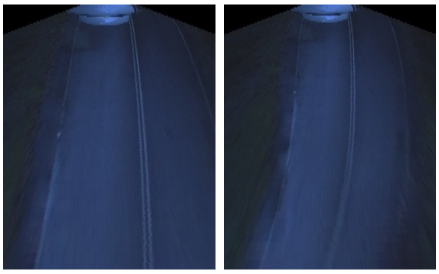

(a) (b)

Fig. 4. a) The uncorrected IPM image. b) The corrected IPM image using a vertically mounted laser.

in areas of shadow and changing light conditions. For further details refer to the section about sensor models below. Another reason to use the 2d image plane of an IPM image is the easier implementation of a process model as described below.

C. The Process Model

In the process model we define how the particles move in the image. For every incoming observation, the filter starts from the bottom of the image. In every time step the particles are then moved to the next line of the image according to a defined Markov model, which incorporates general knowledge about road design.

In this model we define that a straight lane is more likely than a lane of any degree of curvature. Furthermore, a low degree of curvature is more likely than a high degree of curvature. This property is derived from the observation that most road segments are straight or only slightly curved and larger degrees of curvature are usually only present for relatively short times. Finally, we also take into account that there is a certain maximum degree of curvature, which will by definition not be exceeded.

A simplified Markov model for this is illustrated in Fig. 5 as an example of how it works in principle. This simple version lacks the distinguishment between different degrees of curvature and should be regarded for illustrational purposes only. Qualitatively, we define that if the particle was moving straight it is then more likely to move straight again than moving to the left or the right. Moreover, if a sample moved left or right, then it is equally likely to move into this direction again or to just move straight.

Alternatively, a (constrained) random walk process may be applied. Constrained in the meaning that it is more likely that a particle just moves straight up into the next line, rather than moving up and to the left or right.

D. Observation Models

It is reasonable to apply a number of observation models to gain additional robustness for the estimate. Currently we are using an edge image, which also encodes the strength of the edges. The model assumes that the stronger an edge is the more likely it is to be part of a lane marking. This assumption is reasonable, because markings are generally features on the road which are designed to be outstanding. Thus, these features should always be very distinguishable from the surroundings.

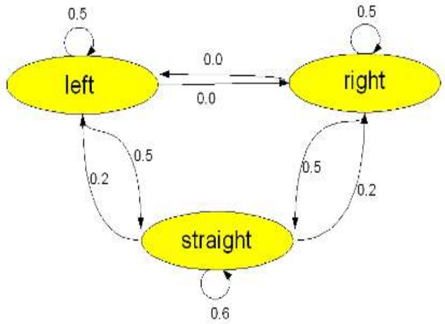


Fig. 5. The constrained random walk model. Where $p1 = p3$ and $p2 > p1$.

Additionally, we use prior knowledge about the colour of the lane markings, i.g. we know whether these are white or yellow and can therefore use the distance to this colour as a quality measure. Again, this is doable because the lane markings colour does not coincide with the surroundings colour in general.

E. Multiple Markings

It is essential to be able to detect multiple lines for we want to obtain knowledge about as many lanes as possible. For this, the filter needs to be able to detect multiple hypotheses. New tracks may be initiated at any time.

For this task we implemented a modified version of a clustered particle filter as described above. The main difference is that we have to remember clusters, so that the association of a particle with a cluster remains in following time steps and therefore the data association is maintained.

Clusters may only be dropped when the average uncertainty was high for a long time. This time is defined by the rules for lane markings given by law, which define the maximum gap between dashes and for other special cases.

Fig. 6 illustrates such a situation, where a single line splits up into two lines. Apparently, both lines are needed to be able to extract both lanes. In the images it visible how the sample set splits up to then track both lines.

F. Uncertainty

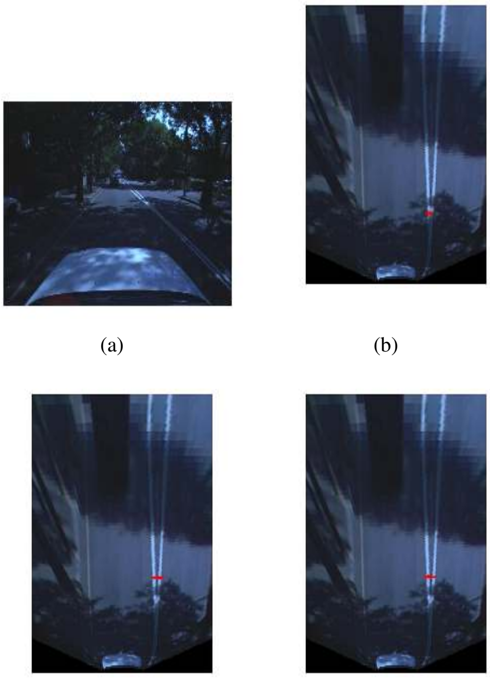
There are certain reasons why we want to keep track of the uncertainty in our estimates. Whenever we do not have enough data for a good estimate we want to able to not regard in further considerations. That means, if data is bad or incomplete, which gives rise to high uncertainty, then it is necessary to have a measure for the uncertainty.

This measure is derived from the variance in the sample set. Obviously, when there is no or bad data, the samples will be distributed over a larger area. This indicates that the produced estimate is less accurate.

Adding the uncertainty measure we can now see the full functionality of the filter, which is illustrated in Fig. 7. Here the particles are shown as red dots, the estimates are shown in green and the uncertainty in yellow.

In Fig. 7(b) the filter is initialized with a high uncertainty. Fig. 7(c) shows when the filter converges the uncertainty decreases. As long as there is a meaningful observation, the uncertainty remains low. However, in a gap the uncertainty grows as in Fig. 7(d). After reconverging at the end of the gap the uncertainty is low again. Hence, this measure allows us to extract dashed and non dashed lines and also enables the filter to produce meaningful outputs with bad data.

Hence, this measure allows us to extract dashed and non dashed lines and also enables the filter to produce meaningful outputs with bad data.



(a) (b)

(c) (d)

Fig. 6. a) The original image. b) Initialization phase, the particles (in red) are uniformily distributed. c) and d) The filter converges and then follows the marking

IV. EXPERIMENTAL RESULTS

The algorithm was tested with several gigabytes of data sets captured by our testbed vehicle CRUISE (see Fig. 1). The testbed is instrumented with several laser scanners, cameras and radar. The data are logged in three different computers synchronized via Ethernet. All our C++ software implementations are based on the Orca2 component based software framework [16].

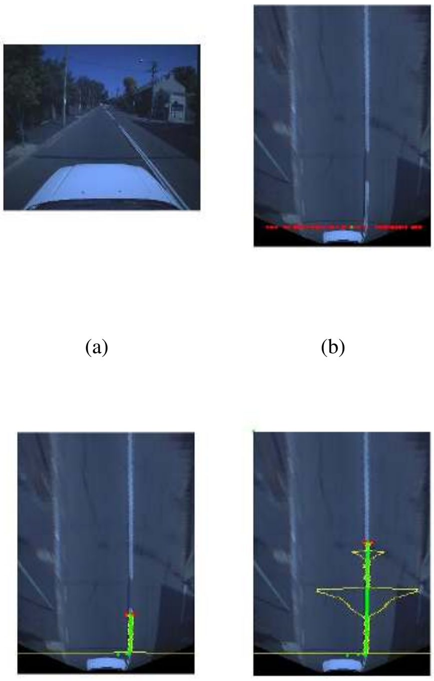
For it is virtually impossible to define a ground truth we had to rely on visual control to check the results. Therefore, only qualitative expression about the detection performance can be given and are supported by the images. It can be seen that in situations of high curvature and gaps and in difficult light conditions the proposed algorithm is still able to produce good results (see Fig. 6, 7, 9, 10).

Performance measures in an early Matlab implementation give an average processing time for one image of approximately 0.2 seconds. In a not yet finished and therefore barely optimized Orca2/C++ implementation we experience processing times of below 0.05 seconds per image.

V. FUTURE WORK

The presented approach is part of ongoing research and several things could not be discussed in this paper. Some of these are briefly discussed in this section.

Currently we are working on the implementation of new sensor models. The goal is to avoid any process that would work on the whole image (as the one presented in section III-B. Instead, we only want to process parts of the image where we have samples. The main benefit is lower computational effort.



(a) (b)

(c) (d)

Fig. 7. a) The original image. b) Initialization phase, the particles (in red) are uniformily distributed and uncertainty is high. c) The filter converged, so the uncertainty is low d) Due to a gap in the marking the uncertainty grows and decreases as soon as the filter starts converging again.

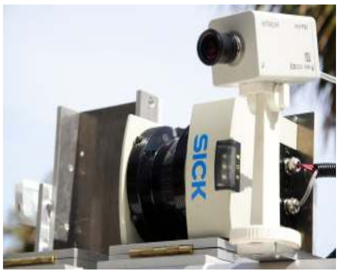
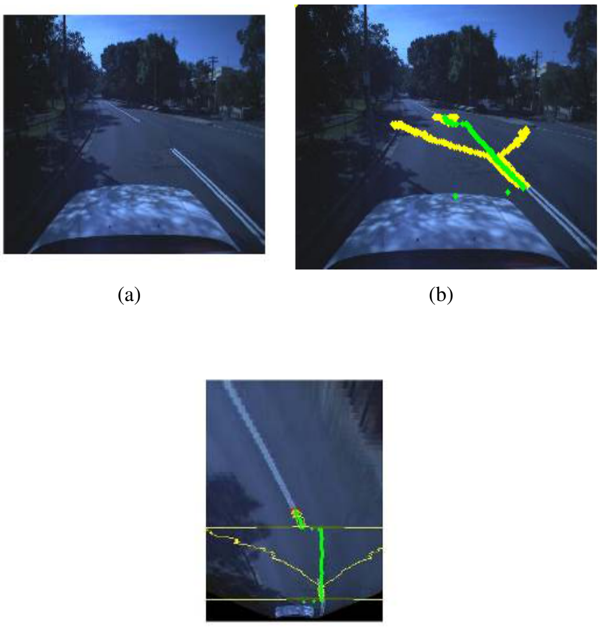


Fig. 8. The camera and vertically mounted laser.

Above this we want to use the original image directly for sensor models rather than using IPM images alone, because the IPM process drops some information of the original image. In the original image colour models and template matching might be used. Template matching is especially interesting to track specific kinds of markings like double lines.

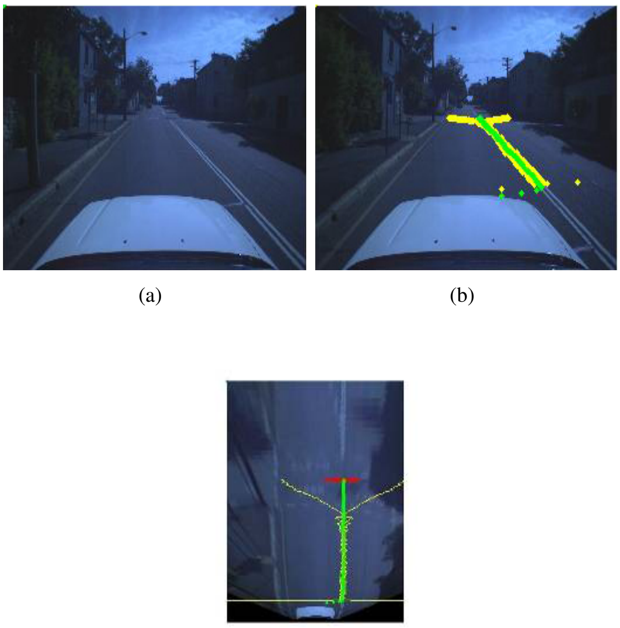
We also intend to address the issue of lane marking tracking. One feasible solution is to utilize the a priori knowledge. It can be done in two ways. The first is to



(a) (b)

(c)

Fig. 9. a) The original image. b) and c) The estimated lane marking (green) and the according uncertainty (yellow).



(a) (b)

(c)

Fig. 10. a) The original image. b) and c) The estimated lane marking (green) and the according uncertainty (yellow).

track individual lines, i.e. the filter is initialized with an estimate using vehicle egomotion. The other possibility is to incorporate the previous information in the filter itself.

Furthermore, the robustness of the algorithm can be improved by incorporating geometry of the lanes, such as the lane width whenever possible.

VI. CONCLUSIONS

In this work we have presented a novel and robust particle filter based algorithm for lane marking detection. The algorithm runs in real-time and is able to pick up any number of lanes. The main advantage of the detection algorithm lies in the use of weak models, which is arguably more general. This is also the main distinguishment to many of the previously presented algorithms. Furthermore, the basic algorithm is relatively easy to implement and offers good results.

We have only addressed the most difficult lane marking ”detection” problem and now we are working on the lane marking ”detection and tracking” problem. The work on this topic will be proceeded and most points outlined in section V are currently under development.

REFERENCES

[1] D. Pomerleau and T. Jochem, Rapidly adapting machine vision for autonomated vehicle steering, *IEEE Expert*, Volume 11(2), pp. 19 - 27, 1996.

[2] E. D. Dickmanns and B. D. Mysliwetz, Recursive 3D road and relative ego-state recognition, *IEEE Trans. of Pattern Analysis and Machine Intelligence*, Volume 14(2), pp. 199-213, 1992.

[3] M. Bertozzi and A. Broggi, GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection, IEEE Trans. on Image Processing, Volume 7(1), pp. 62-81, 1998.

[4] U.Franke, I.Kutzbach: ”Fast Stereo based Object Detection for Stop and Go Traffic”, *Intelligent Vehicles’96*, Tokyo, pp. 339-344, 1996.

[5] C. Kreucher, S. Lakshmanan, A frequency domain approach to lane detection in roadway images, *in Proceedings of the International Conference on Image Processing 1999*, Volume 2, pp. 31-35, 1999.

[6] K. Kluge, Extracting Road Curvature and Orientation From Image Edge Points Without Perceptual Grouping Into Features, *in Proceedings of the Intelligent Vehicles Symposium*, pp 109-114, 1994.

[7] R. Labayrade, J. Douret, J. Laneurit, R. Chapius, A Reliable and Robust Lane Detection System Based on the Parallel Use of Three Algorithms for Driving Safety Assistance, *in IEICE Transactions on Information and Systems 2006, E89-D(7):2092-2100, 2006.*

[8] R. Wang, Y. Xu, Libin, Y. Zhao, A Vision-Based Road Edge Detection Algorithm, in Proceedings of IEEE Intelligent Vehicle Symposium 2002, Vol. 1, pp. 141-147, 2002.

[9] D. Pomerleau, RALPH: Rapidly Adapting Lateral Position Handler, in Proceedings of the Intelligent Vehicles Symposium 1995, pp. 506-511, 1995.

[10] N. Apostoloff, A. Zelinsky, Robust Vision based Lane Tracking using Multiple Cues and Particle Filtering, {it in Proceedings of Intelligent Vehicles Symposium 2003, pp. 558-563, 2003.

[11] A. Doucet, S.J. Godsill, C. Andrieu, On sequential simulation-based methods for Bayesian filtering, *Statistics and Computing*, Volume 10, pp. 197-208, 2000.

[12] S. Arulampalam, S. Makell, N. Gordon, T. Clapp, A Tutorial on Particle Filters for Online Non-linear/Non-Gaussian Bayesian Tracking, *in IEEE Transactions on Signal Processing*, Volume 50, pp. 174-188, 2002.

[13] S. Sehestedt, F. Schneider, A. Kraeussling, Monte Carlo Localization in Highly Symmetric Environments, *In Proceedings of The 3rd International Conference on Informatics in Control, Automation and Robotics*, pp. 249-254, 2006.

[14] D. Fox, Adapting the sample size in particle filters through kld-sampling, in *International Journal of Robotics Research (IJRR)*, pp. 22:9851003, 2003.

[15] Muad, A.M. Hussain, A. Samad, S.A. Mustaffa, M.M. Majlis, B.Y., Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system, *TENCON 2004*, Volume 1, pp. 207-210, 2004.

[16] The Orca2 robotics software framework:
http://orca-robotics.sourceforge.net/