A Particle Filter-based Lane Marker Tracking Approach using a Cubic Spline Model

Rodrigo Berriel*, Edilson de Aguiar, Vanderlei Vieira de Souza Filho, Thiago Oliveira-Santos†
Federal University of Espírito Santo, Brazil
*rodrigo.berriel@gmail.com, †todsantos@inf.ufes.br

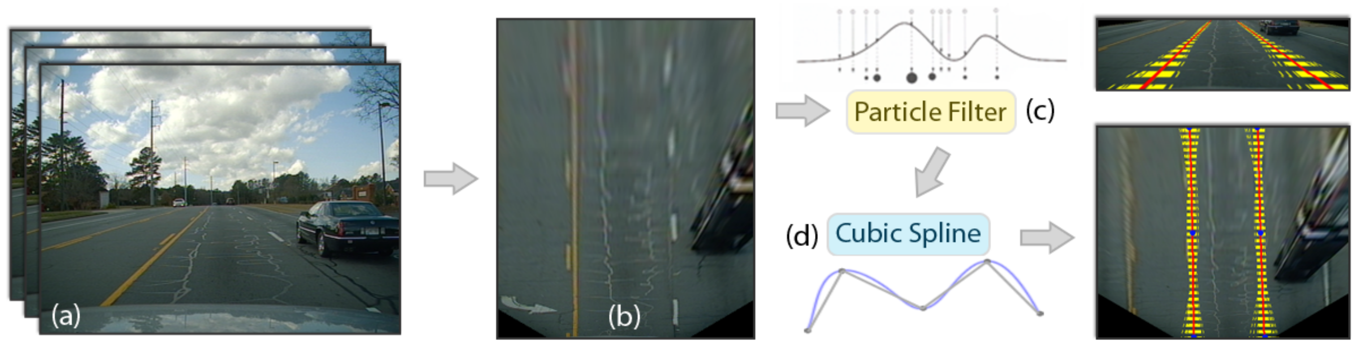


Fig. 1. Overview of our lane marker tracking system: from a sequence of frames (a), the Inverse Perspective Mapping is applied (b) and the lane marker is detected and tracked with the use of a particle filter (c). The output (d) are tracked lane markers modeled by cubic splines for each side. Yellow splines represent all the particles and the red one represents the output, a virtual best particle. The system reports an error of 0.0143 meters with 98.13% of precision.

***Abstract*—In this paper we present a particle filter-based lane marker tracking approach using a cubic spline model. The system can detect the two main lane markers (i.e. lane strips) of marked roads from a monocular camera mounted on the top of a vehicle. Traditional lane marker detection and tracking systems have limitations to properly detect curved roads and to use temporal information to better estimate and track lane markers. The proposed system works on a temporal sequence of images. For each image, one at time, it applies a sequence of steps comprising an inverse perspective mapping to correct for perspective distortions, and a particle filter to smoothly track the lane markers along time. The output of the system is a lane marker generated by a cubic spline interpolation scheme to fit a wider range of lanes. Our system can run in real applications and it was validated with various road and traffic conditions. As a result, it achieves a high precision (98.13%) and a small error (0.0143 meters).**

***Keywords*-lane marker tracking, particle filter, cubic splines**

I. Introduction

Lane marking detection and tracking is an important task that provides useful information for a range of other driving-related tasks, such as: lane departure warning, lane keeping, lane centering and others. These lane-related tasks can provide information for other type of tasks (e.g. driver drowsiness detection). All these modules are directly correlated with the capability to minimize accidents and improve safety while driving, which are the main goals of the so-called self-driving cars. The importance of such algorithms is reinforced by the fact that 93% of all crashes are caused totally or partially by a human factor [1].

Lane marking detection and tracking is a challenging task. Lane markers are constantly occluded by cars, motocycles, bicycles and pedestrians. Also, many of these lane markers are fading after some years of traffic and most of them are not repainted for many years.

In the self-driving car context, all these issues must be correctly addressed and many attempts have been made in order to overcome them. Techniques vary according to the type of sensor: monocular vision camera [2] [3], stereo vision camera [4] [5] and LiDAR [6]. Camera-based approaches have the advantage of being cheaper than laser-based ones and to natively capture the far-field of vision, but they are very sensitive to light variation. Laser-based approaches have the advantage of dealing better with shadows, direct light, sudden illumination variation (tunnels), dark environments, but they have a range limit and the data is usually sparse. Extracting marker lane information using just only one of these inputs, laser or visual data, can be hard. Therefore, most recently, sensor fusion methods [7] have also been explored and its results are promising. The drawback of this technique is usually the performance related to the enormous amount of data to process, and the high cost of integration of these sensors.

Roughly, we can classify the approaches used to model the lane markers in three categories: linear [8], parabolic [9] [10] and spline-based [11]. Each model has its advantages, but a pure linear model has serious limitations and its benefits can be achieved by the others. Some of these approaches use the Inverse Perspective Mapping (IPM) [12], also called bird's-eye view, to reduce the perspective distortion and to enable a constant lane width along the image. The lane constancy helps not only during the detection and tracking phase, but also during the validation [13] of the results.

The methods above still have limitations in fitting their models to curve roads, especially with various traffic conditions. Most of them only report qualitative results. Given these limitations and the need for improved systems nowadays, there is still need for further investigation of robust and real time methods for lane marker detection and tracking.

In this context, this paper presents a method to estimate the lane markers from monocular vision (i.e. a single camera mounted on top of a vehicle) and to track them along a sequence of images. The proposed system receives a sequence of images as input and uses a combination of Inverse Perspective Mapping [12], particle filter [14] and cubic spline model [15] to estimate the two lane markers of each input image. This system has the advantage of modeling a wider range of lanes because of the cubic spline model used to represent them. Moreover, the system is able to improve the tracking using temporal information. To validate the system, 14045 frames from 8 different scenes containing all kinds of situations (light and heavy traffic, lane changes, crosswalks and writings on the road, etc.) were used and the system reported a mean error of 0.0143 meters with 98.13% of precision.

II. Lane Detection System

The system works on a sequence of images, one at time, coming from a monocular forward-looking camera mounted on the top of a vehicle. For each image, it outputs information describing the lane marker (i.e. two splines describing the left and right lane markers). The general steps of the lane marker tracking system are described in Fig. 2. The system basically processes one image at a time and each lane marker individually. Each image is first pre-processed to remove unnecessary parts and to correct perspective distortions. Subsequently, evidences are collected for each lane marker. Finally, a particle filter is used to average the contributions of the currently found evidences and the lane markers detected in the previous frames, i.e. to allow for a smooth track along time. The filter looks for the cubic spline that better represents each lane marker, one for left and one for right. The result is represented by the points of each spline, one point per image row.

A. Pre-processing

The input of the system is a sequence of frames captured by a monocular forward-looking camera mounted on the top of a vehicle. Each frame is firstly converted from color to grayscale in order to reduce the number of channels and to improve performance. Subsequently, a region of interest is set in order to remove irrelevant parts of the image (e.g. sky, horizon, etc).

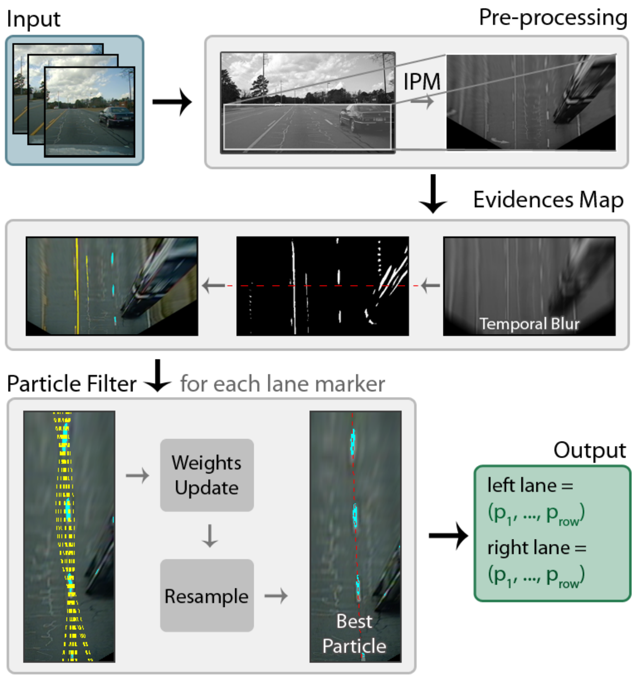


Fig. 2. The Lane Marker Tracking System: it comprises of three core modules: pre-processing, generation of the evidence maps and particle filter. The output of the system is up to two cubic splines that best represents the left and right lane markers.

Finally, an Inverse Perspective Mapping (IPM) [16] is applied to correct the problems of the original perspective view (the width of the lane markers varies according to the distance from the camera). The IPM is a geometrical transformation that remaps from a camera perspective view to a view from the top (also called birds-eye view), where the width of the lane markers is constant. To use this technique, a 3x3 homography matrix (H in Eq. 1) is required to transform the points from the original perspective to the birds-eye view.

$$(x', y', w') = \mathrm{H} \cdot \begin{bmatrix} x & y & 1 \end{bmatrix} \quad (1)$$

The ground plane is assumed to be constant along the input frames. Therefore to estimate the matrix H, four fixed points were manually selected within the lane markers (two in the left and two in right) of a chosen image and mapped to a proportional rectangle representing the IPM image. In this birds-eye view, lane markers tend to have a constant width, except for the cases where there are road inclination and casual bumps from the car since the fixed ground plane assumption fails. The cropped grayscaled IPM image is the result of the pre-processing.

B. Evidence Map

With the bird's-eye view image received from the previous module, the system can search for lane markers candidates. The goal of this module is to collect evidences of the presence of lane markers in the image.

Firstly, a temporal blur is applied to help the following modules to deal with dashed lane markers. The collection of evidences works with thresholds-based techniques. In this paper, two techniques are tested: threshold applied on a Difference of Gaussians [17] and adaptive threshold. In both cases, we divide (from top to bottom) the input image of this module into n regions (Fig. 3) and apply the threshold-based techniques on each of them. The metaparameters (i.e. the standard deviation for the difference of gaussians approach and the constant subtracted from the mean in the case of adaptive threshold, and the thresholds themselves) are set to better capture the candidates in each region. This is done to reduce the influence of the increasing blur caused by the IPM, where near field is less blurred than far field (Fig. 3). The output of this module comprises two binary images (one image for the left and one for the right lane markers), where white dots (i.e. pixels = 255) represent lane marker candidates. The region of each lane marker was empirically defined to fit most of the lane models (see Fig. 3).

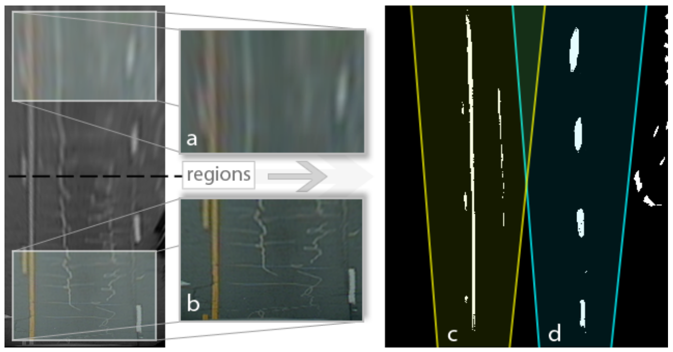


Fig. 3. Evidence map: the IPM transformation adds a blur to the image that increases in the vertical axis as it can be seen in the zoomed figures (a) and (b) of the IPM image. The output of this module is two binary images that represent the left lane and right lane evidences for the lane markers. The yellow (c) and the blue (d) regions represent the regions considered for the left and right lane marker candidates.

C. Particle Filter

Particle filters [14], also known as Sequential Monte Carlo (SMC) methods, are stochastic sampling approaches used to estimate the posterior density of the state-space. The particles are samples from a given distribution and each one of them has a weight that represents its probability of being sampled from a density function.

Basically, the filter consists of four steps: initialization, prediction, weights update and resampling. The initialization of the filter generates random initial particles from a given distribution (all with the same weight). Subsequently, a prediction step is applied to estimate the next position of the particles. Each particle is evaluated considering their new predicted position, and their weights are updated according to an error function. The error functions usually gives a measure of distance from real observations. The weight of a particle represents the probability of this particle to be sampled for the next iteration of the filter. Therefore, particles with higher weights tend to live, whereas particles with lower weights tend to die after the resampling. With the resampling, the output of the filter tends to approximate the real observations over time.

In this paper, two independent particle filters were used, one for each lane marker. The particles were used to model cubic splines with different configurations (i.e. number of control points). Therefore, each particle was defined as a N dimensional point, where N represents the number of control points c of the spline, where $c = (x, y)$ and x and y are the coordinates of the control point in the image. The particle is N dimensional because only the x value of the point is stored in the particle model, the y value is kept fixed. A particle can be defined as in Eq. 2:

$$p = \{\, x_1,\, x_2,\, \ldots,\, x_N \,\}, \quad 0 \leq x_i \leq cols \tag{2}$$

The y values are equally positioned from the top to bottom. To generate the splines from a given particle, the y_i value of each control point c_i is inferred from following Eq. 3:

$$y_i = i \left(\frac{rows}{N-1} \right),\ 0 \leq i \leq N-1 \tag{3}$$

The five steps of our particle filter are described in details below.

1) Initialization: The initialization of the particle filter aims to estimate initial lane marker candidates. The only condition to initialize the filter is the presence of at least 30 evidences, i.e. pixels chosen as lane markers candidates. This was empirically defined based on the number of points needed in the evidence map so that a lane marker could be estimated. The number of evidences can be calculated from the previously generated evidence map. If the number of evidences is above the defined threshold, the particle filter is initialized with a set of random particles (Fig. 4.b). As lane markers are expected to be within a region (assuming car is most likely to be facing forward to the road), each set of random particles (left and right lane) is initialized within a normal distribution having the mean as the center of that region and the standard deviation large enough to cover the whole region. We used $\sigma = 80/3$ because 80 pixels are 80% of the lane width in the IPM image.

2) Prediction: On this step, the filter tries to predict where the generated particles will be in the current frame considering their position in the previous frame. In the proposed prediction model (Fig. 4.c), a particle can randomly move it components (i.e. x coordinate of each of its control points) independently. The prediction is based on a normal distribution with the current particle value as mean and a small standard deviation ($\sigma = 10/3$ was empirically chosen to consider two times the width of the lane marker).

3) Weights Update: At this stage (Fig. 4.d), a set of particles (i.e. lane markers possibilities for a specific lane marker) is already estimated by the particle filter and they all have the same weight. To calculate the weight of a particle, an error function needs to be defined. The error function should be able to measure how well a spline defined by a particle fits to the evidences. The function proposed in this paper tries to cope with the presence of outliers in the evidence map by

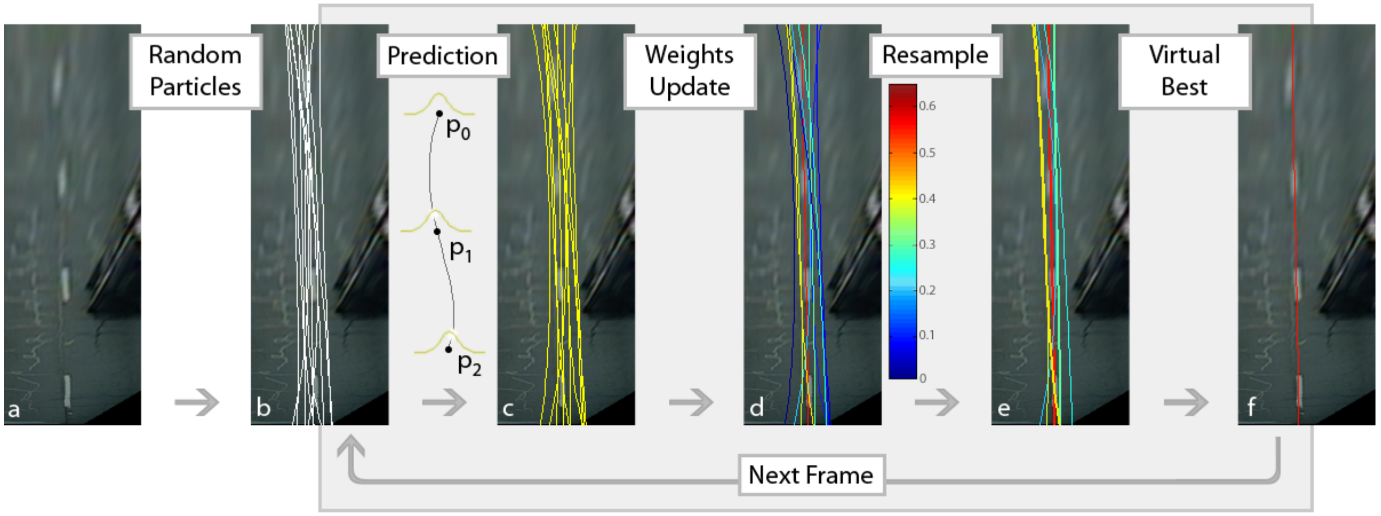


Fig. 4. Particle filter for lane marker detection and tracking. In the initialization of the particle filter, a set of random particles (b) is generated for the input evidence map (a). The particles are subsequently moved considering a normal distribution in the prediction step (c), and have their weights calculated (d). In the resampling (e), particles with higher weight have higher probability to be sampled to the next set of particles. The best particle (f) is represented by a virtual particle that is generated considering the weighted average contribution of the current set of particles of the filter.

only considering the evidences that best fits to the considered particle (i.e. lane marker spline). In addition, it tries to ensure that evidences from the bottom and from the top part of the evidence map are considered. Therefore, the evidence map is divided into N regions that are evaluated independently.

Basically, in order to calculate the error (Fig. 5), the first step is to find the best evidence for each row in the image. The best evidence is defined as the closest evidence to the spline generated from the particle being evaluated, when considering a respective row of the evidence map. The final error $E(p)$ for a particle is the mean of the best evidences for a map, when considering only the 50% best evidences for each of the N regions. Outliers are likely to have larger distances and, therefore, will be disregarded in the error calculation.

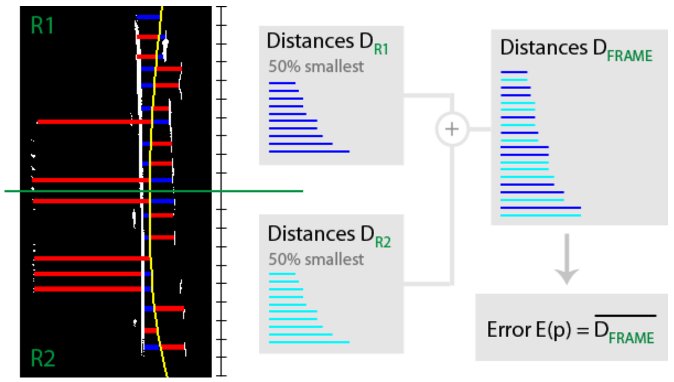


Fig. 5. Illustration of the error calculation function for two regions: in the black image, blue lines represent the smallest distance between the spline and the evidences of a given row, and the red lines represent all other higher distances. For each region, R_1 and R_2, two sets of the 50% smallest distances, D_{R1} and D_{R2} respectively, are joint in a final set D_{FRAME}. The error$E_{(p)}$ is the average of all distances in latter set, $\overline{D_{FRAME}}$.

Because the weights assigned to the particles need to represent a probability, they are defined to be inversely proportional to the error of their corresponding spline, as showed in Eq. 4.

$$W_i' = \frac{1}{1 + e^{E(p)}} \tag{4}$$

They are also normalized as showed in Eq. 5.

$$W_i = \frac{W_i'}{\sum_{j=1}^{m} W_j'} \tag{5}$$

After this normalization, $\sum W = 1$ is ensured. With this set of weighted particles, the best particle of the filter can be estimated as a virtual particle calculated by Eq. 6, where m is the number of particles.

$$p_{best} = \left\{ \sum_{i=1}^{m} W_i \; x_1^i, \; \sum_{i=1}^{m} W_i \; x_2^i, \; \ldots, \sum_{i=1}^{m} W_i \; x_N^i, \right\} \tag{6}$$

4) Resample: The convergence capability of the filter (i.e. its ability to approximate to the real observations) highly depends on the resampling method (Fig. 4.e) that will sample the particles that will live in the next filter iteration. The resampling is performed considering the weights of the particles (calculated in the previous step) as a sampling probability for each particle. This paper uses the Low Variance Sampling described by Thrun in [18].

5) Restart: In order to deal with situations in which there are temporarily no lane markers on the road, the filter was given two states (Fig. 6): enabled or disabled. The filter is enabled when there are enough evidences in the evidence map and the error of the best particle is low, i.e. there are likely lane markers on the road. The filter is disabled when the number of evidences is low or the error of the best particle is high, i.e. there are likely no markers in the road or there are only noisy detection of evidences. An error $E(p) > 5$ was empirically defined as high, based our error calculation. When the filter is being disabled, the current set of particles is stored, and from that point on it outputs no lanes. If the states goes from disabled to enabled, the filter restarts [19]. Restarting the filter means proceeding to the initialization but using the last stored particles instead of a set of random particles. In case the filter is initialized and the error stays high for more than 5 frames, the filter is reinitialized with a set of random particles as in the first frame.

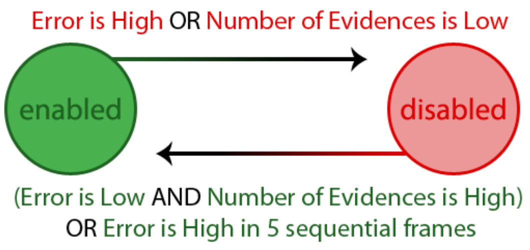

Error is High OR Number of Evidences is Low
enabled
disabled
(Error is Low AND Number of Evidences is High)
OR Error is High in 5 sequential frames

Fig. 6. Finite-state machine. The particle filter can be disabled if the error is high or if there is a not enough number of evidences, but to restart the filter the number of evidences must be enough and the error of the particles should be low. Another case where restarting happens is when the error is high for more than 5 sequential frame.

D. Custom Initialization of the Particle Filter

When the filter is initialized for a frame, a set of particles, randomly generated or representing the last visible lane markers, are considered and they might not represent the current lane marker accurately. In order to speed up the convergence of the filter in a specific frame, the filter iterates up to 10 times (Fig. 7) on the same frame. Although the filter has to run 10 times, it can still be performed quickly.

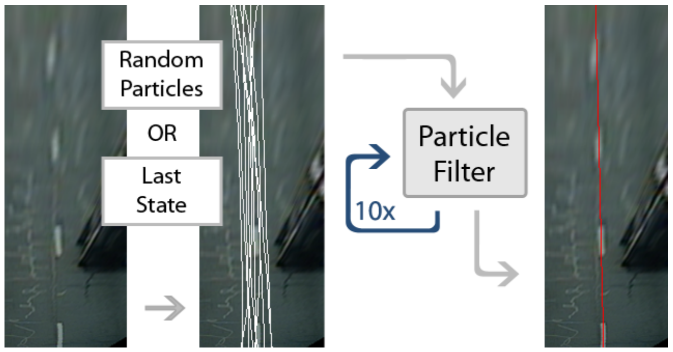


Fig. 7. Illustration of the custom initialization of the particle filter. The custom initialization of the particle filter expects as input an evidence map and a set of particles. This set of particles can be a random (in case of the first initialization) or a specific set of particles (e.g. from the last state of the filter). In order to optimize the first set of particles, the particle filter runs 10 times in the same evidence map given as input.

III. Experiments

In order to validate our system, we ran a set of experiments using datasets recorded under various road conditions and traffic. The experiments were divided into tuning and testing. Tuning was used to adjust the system metaparameters, and test was used to evaluate the performance of the system. Qualitative and quantitative results are reported. The results are compared to a hough-based method described in [13] (its implementation is publicly available[1]).

A. Setup and Dataset

We used 8 scenes with a total of 14045 frames (Table I) with 640x480 pixel resolution from the publicly available dataset [20]. The scenes consist of highway and urban roads with various traffic conditions, lane variations, lane changes and all types of lane markers.

TABLE I
Dataset overview

Scenes	Frames	Scenes	Frames	Scenes	Frames
S1C1	1376	S1C2	1300	S2C1	1424
S2C1	1811	S2C4	2373	S2C5	2492
S3C1	1001	S3C4	2268		

The datasets were annotated by their authors in the original perspective view and then mapped by us to the birds-eye view for evaluation (Fig. 8). The dataset was divided in a validation set (3 scenes) used to tune the metaparameters and a testing set (5 scenes) to evaluate the proposed method.

In addition, we also annotated one of the datasets (S1C1 with 1376 frames) directly in the IPM image (purple dots in the Fig. 8) in order to evaluate the efficiency of our system in the far-field of vision. We will reference this annotated dataset as S1C1-IPM. This dataset is publicly available[2].

The tests were performed in a laptop with Core i5-3230M (4x2.6GHz) and 4GB RAM 1600MHz. The algorithm was implemented in C++ using the open source library OpenCV.

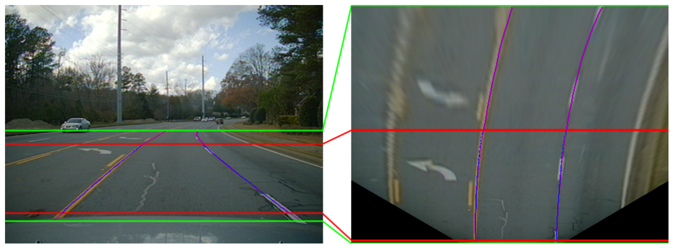


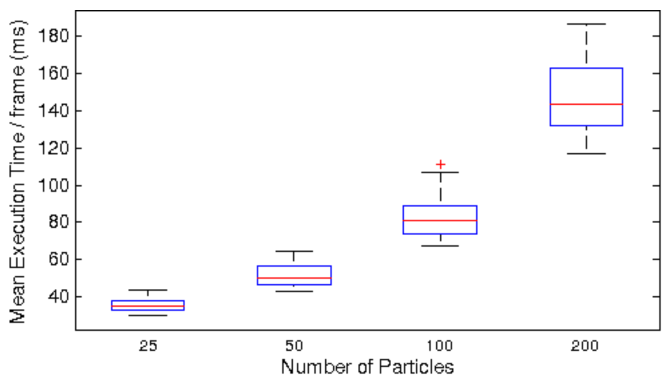
Fig. 8. Regions of interest: (a) red lines are the annotation boundaries and green lines are the boundaries of our region of interest. (b) The full IPM image is our region of interest. The blue dots represent the annotation that came with the dataset, and the purple dots represent our in-house annotation.

B. Tuning

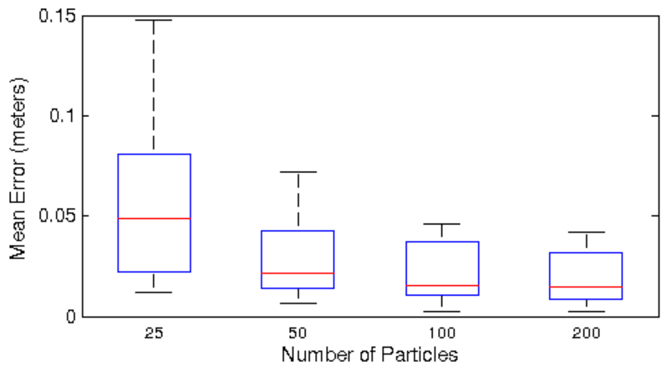
The system was tuned using two approaches to generate the evidence map: adaptive threshold and difference of gaussians. Regardless the used approach, a set of metaparameters need to be tuned in order to get the best of the proposed system:

- *Number of particles*: has a direct influence in the performance of the algorithm as well as balance the running time with the accuracy;

[1]http://github.com/mdqyy/DriveAssist
[2]S1C1-IPM dataset: http://bit.ly/S1C1-IPM or send me an email



(a) Execution Time



(b) Error

Fig. 9. The number of particles is crucial to the performance of the system. After 100 particles there is no substancial decrease in the error (b), but the performance (a) starts to be slow.

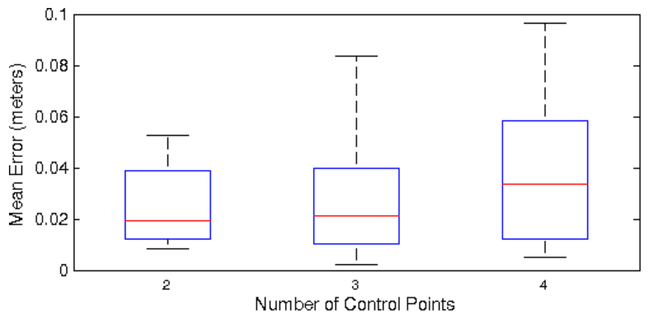


Fig. 10. The number of control points defines the flexibility of our output. We can see that with 4 control points the spline curve starts to adapt too much to the data, losing its generalization. We decided to use 3 control points. The number of control points has almost no impact in runtime performance.

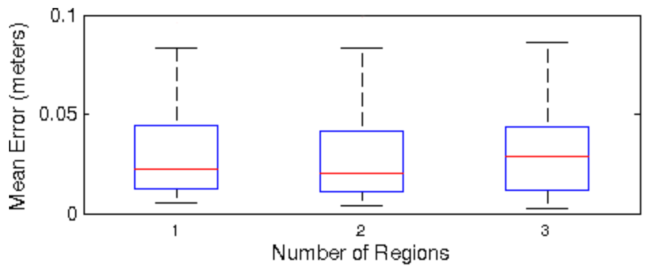


Fig. 11. The number of regions has small effect on the error and performance of the system, but it is important to have at least two regions to ensure that data from both regions of the image are in the error calculation.

- *Number of control points in each particle*: defines how flexible the lane output can be;
- *Number of regions*: allows flexibility on the threshold parameter for different parts of the IPM image;
- *With or without temporal blur*: includes another temporal element to the detection phase and it is supposed to improve the detection of evidences for dashed lane markers (Fig. 12).

The tuning experiments were executed in each dataset from the validation set and the best set of parameters were used to evaluate our system and compare with the hough-based method.

1) Number of Particles: We tested 4 possible numbers of particles: 25, 50, 100 and 200. Although 200 particles presented best results, it is not a good choice because its running time was too high (Fig. 9) for our needs (almost 2x more than 100 particles). Besides that, the difference between the mean error of 200 and 100 particles was not enough to compensate for its performance issue. The best number of particles, in terms of performance and mean error (Fig. 9), was 100 particles.

2) Number of Control Points: The control points give flexibility to the spline curve that models our lane marker output. We tested 3 possibilities (Fig. 10): 2, 3 and 4 control points. With 2 control points, our model is expected to behave such as a line like in the hough-based methods. With 3 and 4 points, we increase our model flexibility to adapt to a wider range of lane markers and achieve better results than a straight line. The result of this tuning experiment showed that 3 control points give the spline model a single inflection point and performs better than giving two inflection points. With 4 control points the model adapts too much to the evidences and sometimes tend to follow the outliers.

3) Number of Regions: The Inverse Perspective Mapping has a side effect: a blur that increases from near-field to the far-field of vision (Fig. 3). To overcome this side effect, we decided to divide the image into a small set of regions and apply different thresholds to generate the evidences in each region according to its need. We divided the image in 1, 2 and 3 regions (Fig. 11) and the results show that dividing it in 2 regions provides good results for our purpose.

4) Temporal Blur: Dashed lane markers deserve special attention in the generation of the evidence map. In order to deal with them, a temporal blur of 5 frames was applied. As the horizontal variation in a sequence of IPM images is too small, the vertical changes are magnified and dashed lane markers look longer than they really are (Fig. 12). As this technique can be applied without losing much computational performance, experiments were run with it.

IV. Results

From the above-mentioned tuning, the best metaparameters were chosen and experiments were run in the testing set, with

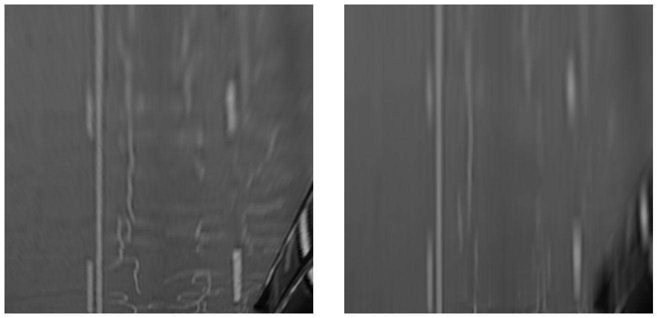

(a) Original IPM Image (b) Temporal Blur

Fig. 12. Effect of a Temporal Blur.

the following configuration:

- Number of particles: 100
- Number of control points: 3
- Number of regions: 2
- Temporal Blur: yes

In this paper, three methods are compared: hough-based [21], and the proposed system with two variations in the generation of the evidence map: adaptive threshold and difference of gaussians.

A. *Quantitatively*

To evaluate the methods quantitatively, 5 measures were chosen: error (meters), execution time (milliseconds), precision, recall and accuracy.

a) Error: The error was calculated using Eq. 7 and 8, as proposed by [13]:

$$\lambda_{(i,f)} = \max(|Gt_{(i,f)} - X_{(i,f)}| - \frac{W}{2}, 0) \tag{7}$$

$$Er(f) = \sum_{i=1}^{R} \frac{\lambda_{(i,f)}}{R} \tag{8}$$

In Eq. 7, $Gt_{(i,f)}$ is the ground truth location of the lane marker, and $X_{(i,f)}$ is its estimate on row i of frame f. W is the width of the interval around the ground truth location, and λ is the measured distance. R is the total number of rows and $Er(f)$ is defined in meters. The proposed system is more consistent across the test set, reporting lower variance than hough-based method, as we can see in Fig. 13.

Besides these measures, we used the S1C1-IPM dataset to evaluate the methods. This dataset was annotated directly in the IPM images (i.e. it includes the far-field of vision). The results shown in Table II confirms, with a greater difference, that our system outperforms the hough-based method.

TABLE II
EVALUATION USING S1C1-IPM DATASET

Method	Error (meters)
Hough-based	0.054672
Our system + Adaptive Threshold	0.025297
Our system + Difference of Gaussians	0.015137

b) Precision, Recall and Accuracy: With the proposed system, we could classify the output as a True Positive (estimated a lane marker when there was one), True Negative (estimated no lane marker when there was none), False Positive (estimated a lane marker when there was none) and False Negative (estimated no lane marker when there was one). Using these values, we calculated the next three measures: precision, recall and accuracy (Fig. 14).

c) Execution Time: Performance is another good measure, because real applications that benefit from the output of a lane marker detector usually need to run in real time or very close to it. In our understanding, an execution time up to 100 milliseconds per frame is enough for real applications, especially if using better hardware or GPU. All methods reported good computational performance. However, the Hough-based method reported slightly better average results (Fig. 15) because of its simplicity. Althought, it varies more around the mean value.

B. *Qualitatively*

As observed in Fig. 16, the proposed system can adapt well to curved roads. That was one of the motivations of choosing a spline model. Although hough-based methods can perform well on straight lanes, they are outperformed in case of curved roads.

C. *Limitations*

The application of the IPM is essential to the proposed method. However, we used a static homography matrix which might cause an undesired variation in case of bumps or variations in the slope of the road. This side effect might cause occasional variations in the IPM.

In addition, further investigation should be performed in scenes with heavy traffic, cross-walk or pedestrians. Lane markers are likely to be partially occluded in case of intense traffic. Cars in the IPM image assume an undesired shape that can be confused with lane markers. Crosswalks and writings on the road bring also additional challenge because they are true road markers and must be avoided in the evidence map. We plan to improve the robustness of our evidences map generation in the future.

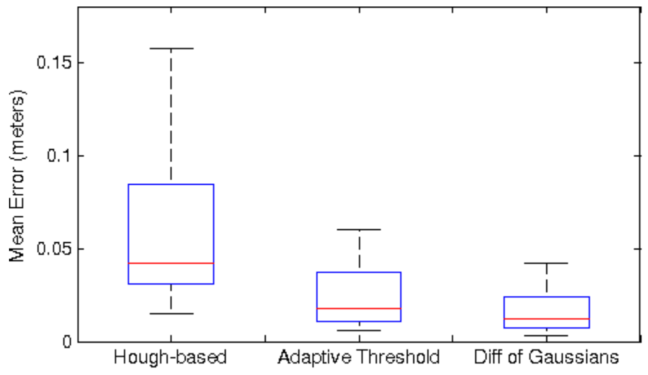


Fig. 13. The proposed system outperformed the hough-based method

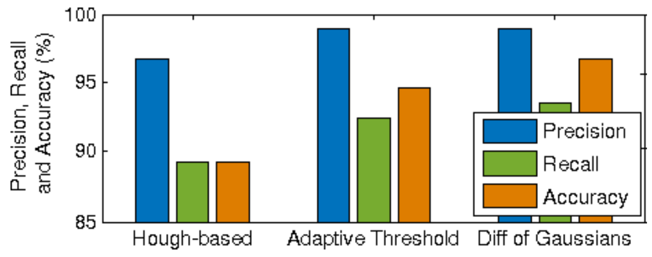


Fig. 14. Precision, recall and accuracy (%).

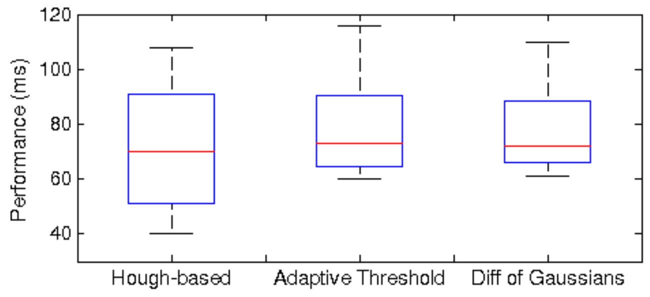


Fig. 15. The hough-based method runs faster than our method variations, but the proposed system also performs as fast as needed for real applications

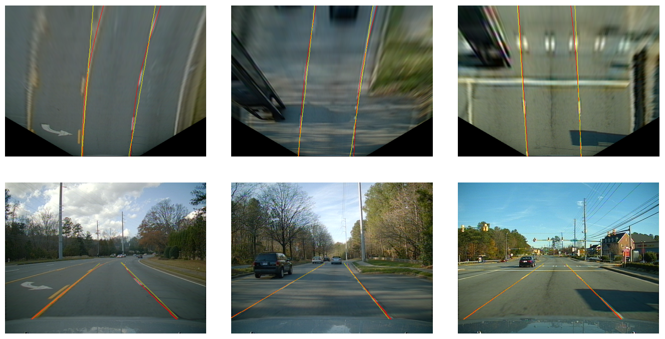


Fig. 16. Qualitative comparison: proposed system (red) and houhg-based (yellow). From left to right, frames: 532 (S1C1), 144 (S2C2), and 565 (S3C1).

V. Conclusion

This paper presents an approach to estimate lane markers from a single image. The proposed method can be used in several other systems, such as lane departure warning, lane keeping assist and lane centering. Our experiments evaluated the proposed method under real life environments. They showed that our system is able to model curved roads, and to achieve good precision, recall, and mean error of 98.13%, 92.97%, and 0.0143 meters respectively. In other words, our method is able to provide reliable estimates with a high level of confidence. In addition, the results showed that the proposed approach outperforms hough-based methods.

Acknowledgment

We would like to thank Universidade Federal do Espírito Santos - UFES (projects COCADOIC - 5384/2014, 20148447DPQ), Fundação de Amparo Pesquisa do Espírito Santo - FAPES (grants 53631242/11 and 60902841/13, and scholarship 66610354/2014), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES (grant 11012/13-7) and Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (298187:12 BRL)

References

[1] H. Lum and J. A. Reagan, "Interactive highway safety design model: accident predictive module," *Public Roads*, vol. 58, no. 3, 1995.

[2] D. Pomerleau, "Ralph: rapidly adapting lateral position handler," in *Intelligent Vehicles '95 Symposium., Proceedings of the*, Sep 1995, pp. 506–511.

[3] J. C. McCall and M. M. Trivedi, "An integrated, robust approach to lane marking detection and lane tracking," in *Intelligent Vehicles Symposium, 2004 IEEE*. IEEE, 2004, pp. 533–537.

[4] M. Bertozzi and A. Broggi, "Gold: a parallel real-time stereo vision system for generic obstacle and lane detection," *Image Processing, IEEE Transactions on*, vol. 7, no. 1, pp. 62–81, Jan 1998.

[5] S. Nedevschi, F. Oniga, R. Danescu, T. Graf, and R. Schmidt, "Increased accuracy stereo approach for 3d lane detection," in *Intelligent Vehicles Symposium, 2006 IEEE*. IEEE, 2006, pp. 42–49.

[6] P. Lindner, E. Richter, G. Wanielik, K. Takagi, and A. Isogai, "Multi-channel lidar processing for lane detection and estimation," in *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, Oct 2009, pp. 1–6.

[7] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nuchter, "A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios," *Vehicular Technology, IEEE Transactions on*, vol. 63, no. 2, pp. 540–555, 2014.

[8] J. G. Kuk, J. H. An, H. Ki, and N. I. Cho, "Fast lane detection amp; tracking based on hough transform with reduced memory requirement," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, Sept 2010, pp. 1344–1349.

[9] K. H. Lim, K. P. Seng, L.-M. Ang, and S. W. Chin, "Lane detection and kalman-based linear-parabolic lane tracking," in *Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC'09. International Conference on*, vol. 2. IEEE, 2009, pp. 351–354.

[10] C. R. Jung and C. R. Kelber, "A lane departure warning system based on a linear-parabolic lane model," in *Intelligent Vehicles Symposium, 2004 IEEE*. IEEE, 2004, pp. 891–895.

[11] Y. Wang, D. Shen, and E. K. Teoh, "Lane detection using spline model," *Pattern Recognition Letters*, vol. 21, no. 8, pp. 677–689, 2000.

[12] D. Seo and K. Jo, "Inverse perspective mapping based road curvature estimation," in *System Integration (SII), 2014 IEEE/SICE International Symposium on*, Dec 2014, pp. 480–483.

[13] A. Borkar, M. Hayes, and M. T. Smith, "An efficient method to generate ground truth for evaluating lane detection systems," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1090–1093.

[14] A. Doucet, N. De Freitas, and N. Gordon, *An introduction to sequential Monte Carlo methods*. Springer, 2001.

[15] C. H. Reinsch, "Smoothing by spline functions," *Numerische mathematik*, vol. 10, no. 3, pp. 177–183, 1967.

[16] H. A. Mallot, H. H. Bülthoff, J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," *Biological cybernetics*, vol. 64, no. 3, pp. 177–185, 1991.

[17] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[19] B. Turgut and R. Martin, "Restarting particle filters: An approach to improve the performance of dynamic indoor localization," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, Nov 2009, pp. 1–7.

[20] A. Borkar, M. Hayes, and M. Smith, "A novel lane detection system with efficient ground truth generation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 1, pp. 365–374, March 2012.

[21] X. Li, E. Seignez, and P. Loonis, "Reliability-based driver drowsiness detection using dempster-shafer theory," in *Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on*. IEEE, 2012, pp. 1059–1064.