Fast multi-lane detection and modeling for embedded platforms

Marcos Nieto*, Lorena García*, Orti Senderos* and Oihana Otaegui*
*Vicomtech
Mikeletegi 57, P. Tecnológico, 20009, San Sebastián, Spain
Email: see http://www.vicomtech.org

***Abstract*—Most Advanced Driver Assistance Systems (ADAS) or Autonomous Driving (AD) functions require the ability to perceive the road and its elements around the ego-vehicle. The precise localization of other road participants (e.g. vehicles, pedestrians, traffic signs) is demanded at lane level, to enable higher semantic analysis of the scene. This requires a lane detection and modeling stage able to provide the number of existing lanes, and their precise local geometry. The current trend in computer vision is to use the full power of GPU technology with deep learning-based detection methods, which requires costly high-end platforms, and difficult the co-existence with other heavy-processing functions (e.g. vehicle detection), specially critical when considering a single platform processing multiple cameras and Laser scanners. In this paper we propose an efficient lane detection and modeling pipeline, composed of optimized steps for segmentation, transformation, modeling, control and tracking. The method is able to detect multiple lanes and their curvature, in continuous function, with minimal processing power requirements, thus enabling its implementation into low-cost embedded platforms. Experimental results support our claims, and demonstrate that the proposed method outperforms other methods in the literature in computational cost, while keeping good accuracy results for a variety of road types.**

I. Introduction

It's been at least two decades since the first effective lane detection techniques appeared [1], starting a research line that represented one of the first successful examples of computer vision techniques applied to the automotive sector. However, during the following decade (2000-2010) this research found only partial success in its deployment into commercial systems, with the consolidation of Lane Departure Warning Systems (LDWS). These LDWS, due to the poor HW capabilities of the time, were limited to extremely simple methods and constrained to basic detection of bright pixels at given positions of the image. More advanced modeling techniques [2][3][4], which required more computational resources, never broke the research-to-industry gap [11].

The focus turned into detection of objects, and for the last decade (2010-2018), the industry and scientific community have produced enormous advances in HW/SW for object detection, tracking and recognition, using single camera, stereo set-ups and range sensors (e.g. Laser scanners). Specially relevant is the revolution caused by the re-birth of Neural Networks (i.e. Deep Learning) when applied on massive parallel computing devices such as GPUs or FPGAs, and in general, due to the rapid development of machine learning and computer vision frameworks, boosted by investments from IT companies (e.g. Google, NVIDIA, Intel, etc.).

As a result, in recent years (since 2015) more advanced ADAS are being devised, which participate in the race towards autonomous driving, for instance in the form of function-specific mechanisms, such as Automatic Breaking, Automatic Parking, Lane Keeping Systems, Lane-Level Navigation, etc.

Most, if not all, of these ADAS require now the review of the old lane detection and modeling techniques, provided a rich description of the road is needed to locate all those detected vehicles and vulnerable road users (i.e. pedestrians, bikes) in the semantic coordinate systems of a road/scene description. Also navigation applications using digital maps get benefit from camera-based lane modeling in order to compute image-to-map alignment and generate accurate localization, improving GNSS accuracy (which nowadays, using low-cost receivers, still remain above meter-accuracy).

As a consequence, recent years have shown progress on lane modeling, particularly using deep learning resources (e.g. LaneNet from NVIDIA's DriveWorks[1]), multi-sensor set-ups [9], and exploiting advances in image segmentation with transfer learning [10]. Results show extremely accurate and robust detections.

Still, we argue that using deep learning for lane detection and modeling represent a waste of resources which does not really solve the problem, but transfer it into a number of other domains: (i) to find a cost-effective inference GPU or FPGA-enabled platform; (ii) to create or purchase a sufficiently large annotated dataset; (iii) to jointly optimize the allocation of computational resources for co-existing modules (such as semantic segmentation, Lidar processing, GNSS or calibration processes). None of the previous problems is trivial, and CPU-only, light-weight lane detection seems to be a reasonable proposal.

Considering the gathered experience on lane detection and modeling by years of (previous-to-deep learning) techniques, we have created a method that feeds from good practices [6], successful architectures [5], and is composed of a number of extremely efficient and effective steps required to provide a robust lane detection, flexible multi-lane model, lane change logic, auto-calibration, parameter learning and auto-assessment with curvature and lane marking type information.

[1] https://developer.nvidia.com/driveworks

ISBN 978-90-827970-1-5

Our approach can be deployed in almost any embedded platform, as its computational cost is significantly lower than other existing approaches, producing a range of 1-10 ms/frame processing time for different tested platforms, while keeping state-of-the-art quality and robustness.

II. Pipeline overview

The proposed pipeline is illustrated in Figure 1. Its design has followed a number of principles: apply a single image processing step, split processing in simple steps, remove parameterization by online learning, utilise calibration to operate on a bird's-eye view domain where to apply geometric constraints, and keep computational load as constant as possible.

One of the key aspects of the approach is the computation of a homography transform between the original image and a warped domain corresponding to a bird's-eye view of the road. Next sections discuss our approach to obtain such transformation, keep its computation efficient, and its application to simplify the geometric models of the lane.

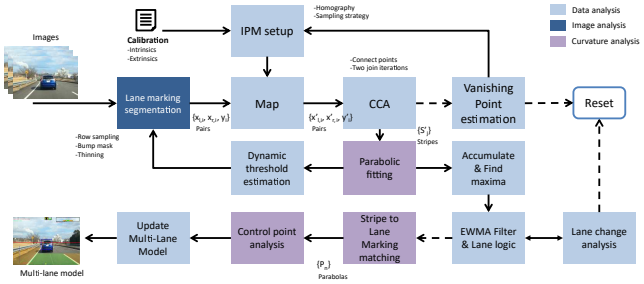

Fig. 1. Lane detection pipeline, highlighting image processing steps, data processing and curvature analysis.

Next sections will detail each of the steps. In summary, the input image is filtered to find pixels likely belonging to lane markings, then, their positions are transformed into the bird's-eye view domain, where a stripe analysis is carried out to group these pixels according to their proximity. Lateral tracking is applied using parabolic fitting to stripes and voting into an online accumulator. Curvature and lane marking type analysis is applied at the last step, by finding the stripes which fit to the lateral tracking results. The rest of the modules provide the required robustness: auto-calibration, auto-assessment and reset functions.

III. Lane marking segmentation

We propose a row-level bump detection technique, which is able to detect lane markings of any width. We define it as the Dynamic Step Row Filter (DSRF), in comparison with the Step Row Filter (SRF) proposed in [7].

The SRF is a variation of the classic top-hat filter [12], favoring lane markings with similar positive-negative gradients. The filter is applied at row level such that for each y-th row of an image $I_{x,y}$ with resolution $W \times H$ the SRF produces filtered values $\{F_x\}_{x=1}^{W}$:

$$F_x = 2I_x - I_{x-\tau} + I_{x+\tau} - |I_{x-\tau} - I_{x+\tau}| \qquad (1)$$

where τ is the step in the x-dimension used to compute the differences. In SRF, τ is precomputed according to an expected lane marking width, and its perspective variation with respect to the horizon (τ linearly decreases to zero from the bottom row to the horizon line).

In the proposed DSRF, instead of using a pre-defined τ value, pairs of up and down slopes in intensity are found (i.e. intensity variations above an intensity difference threshold T), with x-coordinates x_l and x_r, such that the values F_x for the pixels in the range x_l to x_r are computed as:

$$F_x = 2I_x - I_{x_l-1} + I_{x_r+1} - |I_{x_l-1} - I_{x_r+1}| \qquad (2)$$

Each found pair is then defined as $p = (x_l, x_r, y, F_{x_m})$, where x_m is the middle position between x_l and x_r. Since only F_{x_m} is required to characterize each pair, all the other DSRF values for the range x_l and x_r can be ignored and not computed. One advantage of DSRF over SRF is that all lane markings are detected, and no expected width prior parameter is required (see Fig. 2). Additionally, to further decrease the computational load, not all image rows are filtered. The rationale is that the perspective effect makes the lower part of the image most abundant in redundant information, while far distance is represented with few very valuable pixels.

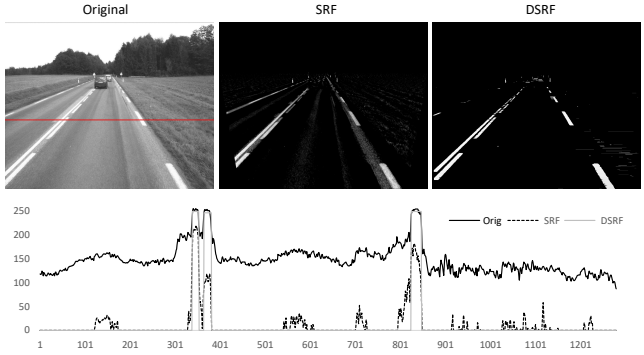

Fig. 2. Sample comparison of SRF and DSRF for a given row (y-axis is 8-bit intensity level): SRF produces noisier results, while DSRF is able to produce a cleaner output, closer to a binarized result.

In an attempt to suppress the impact of the perspective effect, many researchers have adopted the construction of a bird's-eye image (also known as Inverse Perspective Mapping [1]), a warped image built using an image-to-road homography, and then apply the filter on the transformed domain. However, the construction of such image is itself computational costly , and introduces its own distortion to the available information, i.e. interpolation at this domain implies far distance is represented with as many pixels as the close distance, but with much less information.

So, to avoid this additional cost, it is better to apply all image processing at the original domain, and then transform the resulting points or geometric entities into the IPM domain for further modeling. Row sampling at the original image can be used to target a fixed number of rows to process, N, and then keep the computational cost known and steady.

ISBN 978-90-827970-1-5 © EURASIP 2018

Regularly sampling to select only some rows on either the original domain or the IPM domain does not solve the perspective problem either. Regular sampling in any domain produces information which overlaps in the other domain, e.g. in the extreme case all rows are filtered in the original domain, a lot of resources are wasted analyzing bottom rows, since many of them correspond to a single row at the IPM domain; analogously, using all rows at the IPM domain produces useless information at the far distance, which is interpolated in this domain and correspond to only some rows close to the horizon at the original domain.

We propose a hybrid approach, which consists on using a non-regular sampling on the original domain, in a way a single row is sampled from a group of adjacent rows which correspond to the same row on the IPM domain. The result is that the sampling step decreases from bottom to top, respecting all or most of the rows corresponding to the far distance, and sampling the closer regions in a way no overlapping is produced at the IPM domain. The result is a perspective-guided sampling approach, which ensures no resources are wasted, and can be used to keep a fixed number of rows to be computed. The rows to be analyzed can be obtained by first defining a regular step on the original domain $\Delta y = \frac{H-h_y}{N}$, where h_y is the y-coordinate of the horizon. Iterating over candidate rows $y_j = \lfloor h_y + j\Delta y \rfloor$, $j = 0...N-1$, and verifying $y_j \neq y_{j-1}$ and $y'_j \neq y'_{j-1}$ where y' is the corresponding y-coordinate at the IPM domain.

IV. Stripes computation

Pairs p are transformed into the IPM domain, applying the image-to-plane homography to each of its points. Then, a connected-component analysis (CCA) is applied to group pairs into stripes, $\mathcal{S} = \{S_j\}_{j=1}^{S}$, applying bottom-to-top processing. Small gaps (i.e. missing pairs) are allowed, in order to reconstruct stripes not correctly detected by the DSRF.

Additional filters are applied at stripe level: valid lane markings should have homogeneous width, should not present variations in second derivative, and should span a sufficiently large longitudinal distance. Note that complex lane markings, such as urban signs or text are correctly detected by the DSRF, but filtered at this step to simplify the model fitting process.

V. Lane model

The proposed approach divides the model into two parts: lateral tracking, and curvature analysis. It follows the principles expressed in [5], where Rao-Blackwellization splits an inference problem into a linear and a non-linear part, to simplify both steps. In our case, the division consists on finding first the lateral position of lane markings, according to robust, reliable information, and then use it as a basis in a second step to estimate curvature (which is always less reliable due to the very limited amount of information the images contains about the far distance).

A. Lateral tracking

For each consolidated stripe, a parabolic fitting process is applied to each stripe. A parabola can be described by $x = k_1y^2 + k_2y + k_3$. In order to prevent fitting solutions which do not correspond to longitudinal lanes, we can include the restriction that the parabola is tangent to the vertical direction at the bottom row of the image, i.e. the first derivate is zero at $y = H$. Therefore, the model is defined as $x = k_1(y^2 - 2Hy) + k_3$, which is a two-parameters model, where we have applied the restriction as $k_2 = -2k_1H$. Although this model is simpler than a fully three-parameters parabola, it provides suitable results specially robust for short stripes. A least-squares solution is applied using the points of the stripe.

The computed parabola is used to determine the x-coordinate at the bottom row in which the stripe hits. A 1D-accumulator h_i with equal number of bins as the width of the IPM domain, is created where each stripe votes according to its number of pairs at the hit position (plus a smoothing gaussian kernel). A first step of temporal smoothness is applied, in the form of a Exponential Weighted Mean Average (EWMA). Each position of the accumulator is filtered with the following EWMA expression:

$$h^*_{i,t} = \alpha h^*_{i,t-1} + (1-\alpha)h_{i,t} \tag{3}$$

where $h_{i,t}$ is the measured value of the accumulator at the i-th position, in instant t, while the symbol * denotes EWMA filtered response. The value of alpha can be set to meet some expected lateral motion dynamics of the vehicle, but to simplify, we use $\alpha = 0.9$ when the vehicle is driving straightforward, and $\alpha = 0.5$ when moving laterally.

Local peaks of $h^*_{i,t}$ are used to update the estimated position of lane markings, $\mathcal{L} = \{l_k\}_{k=1}^{L}$. In our implementation, we use a fixed number of possible lane markings $L = 8$, which corresponds to one central ego-lane and 3 lateral lanes at each side (which is usually far more than necessary for usual optics on forwards-looking cameras). Those lane markings not associated to any peak enter into a lane-level sanity check, to infer their position according to measured lane markings.

B. Curvature analysis

Curvature analysis is applied to stripes and computed lane marking lateral positions to create a grid of control points at the IPM domain (similar in concept to the approach described in [8]). The grid is defined by a number R of equidistant heights in which the IPM is divided, and L points corresponding to each lane marking. Let then the grid be $\mathcal{C} = \{C_{r,l}\}_{r=1,l=1}^{R,L}$.

The lateral positions (at the bottom) of the lane markings $\mathcal{L}$ are then used as starting points to analyze the stripes and find an association between stripes $\mathcal{S}$ and lane markings. First, an $S \times L$ matrix is built with the likelihood of each stripe to correspond to a given lane marking according to the distance $d_{i,k}$ between the lane marking position and the lateral hit point ot the stripe's parabola at the bottom row:

ISBN 978-90-827970-1-5 © EURASIP 2018

$$p(s_i, l_k) = \exp(-\lambda d_{i,k}) \quad (4)$$

where s_i is the i-th stripe,and l_k the k-th lane marking. The parameter λ can be defined to ensure a certain restriction on the likelihood: e.g. the likelihood decays to 0.1 for a distance of one-third of the domain width, thus $\lambda = \frac{\log(0.1)}{d_{i,k}}$.

A 1-to-1 association is obtained first, to find long stripes (those spanning all R control point heights) which can be reliably associated to specific lane markings. If found, a parabola fit is used to define the position of the control points at different heights in the IPM domain.

Short stripes (corresponding to discontinuous lane markings) are less reliable: their parabolic fit is more sensitive to detection noise and outliers. A second $S' \times S'$ matrix is built (being S' the number of short stripes) to represent the likelihood of two stripes to belong to the same curve. For that purpose, pairs of stripes are joined and their joint parabola is computed. The fit error is used to produce a likelihood value. Note that this matrix is triangular superior, and pairs of stripes are only evaluated if one stripe is strictly above the other. That limits the number of parabolic fits needed in the procedure.

Control points which have been defined from a stripe, are declared as measured. The others are left as not-measured. At this point inference steps are given, in order to define as inferred those control points whose position can be interpolated from measured ones (e.g. immediate superior and inferior). This can be done applying lane-level logic, to ensure reasonable shapes are produced (smooth curve, semi-constant lane width, etc.).

Each lane marking has R control points either measured, inferred, or not-measured. This information can be used to determine its type. In this work we stay with two simple states: continuous (lane markings with all control points measured), and discontinuous (lane markings with some control points measured). Lane markings with all control points remaining as not-measured are set to be of unknown/unobserved type.

VI. Implementation

The proposed method has been implemented as a C++ library, using efficient C++11 principles. Calibration can be injected to the application in a number of ways, for instance providing a file of intrinsic and extrinsic parameters of the projection matrix. Also, the rotation and translation of the camera with respect to a given coordinate framework.

A. Lane marking segmentation

One of the key steps of the proposed approach is the DSRF for lane marking segmentation. To evaluate its performance, we have used the ROMA dataset [12], which is composed of 116 annotated images of road scenes which include variable lightning conditions, varied types of context (highways, rural areas, urban scenarios), and different levels of lane marking complexity and visibility. The dataset is subdivided in three categories: adverse light, high curvature and normal.

We have run SRF and DSRF against all the images in the dataset, aggregating the results and spanning the binarization threshold T from 1 to 255, in order to obtain the F-measure curve for each algorithm. The results are shown in Fig. 3.

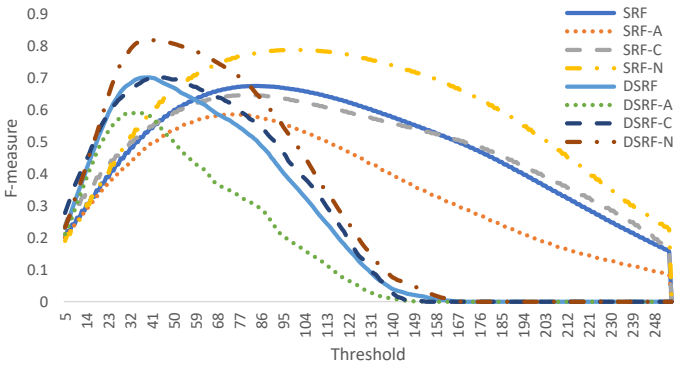

Fig. 3. F-measure computed for the ROMA dataset, for algorithms SRF and the proposed DSRF, spanning threshold T values from 0 to 255. Curves show the results aggregated for the entire dataset, the adverse sub-set (labeled as -A), curvature sub-set (-C), and normal (-N).

As we can see, the DSRF reaches better results than SRF in all categories, quite good for the normal sub-dataset, reaching F-measures of 0.8, while staying at 0.6 for the adverse light subset, which is composed by images where lane markings are imaged with reflections and strong shadows.

One significant advantage of DSRF with respect to SRF is its speed. As explained in section III, DSRF is designed to quickly advance through pixels seeking for up-down slopes, computing the SRF response only for found pairs. As a result, we have observed DSRF is $20\% - 30\%$ faster than SRF.

B. Subjective results

On the top of the image segmentation provided by the SRF, the method gets robust by applying the successive steps of Stripe creation and filtering at the IPM domain, lateral position tracking, and curvature analysis. In our experiments, the method is able to determine accurately the presence of the ego-lane, and in most cases, also lateral lanes.

The stripe analysis helps identifying which pairs actually correspond to lane markings, and which are outliers or other sort of road paintings (see Fig. 4). The accumulation on the histogram h_i^* is used to manage lane-level logic, such as lane changes (upon identification, the information is shifted in the model to continue modeling the ego-lane and up to 3 lateral lanes at both sides), lane merges and lane splits.

Curvature is built on the top of the other steps, and naturally is the one subject to more uncertainty. Curvature is modeled from small bits of information (from pixels in the distance). The proposed method is able to weight the contribution of these pixels to the model fit in order to accurately determine the curvature of each lane marking. The usage of control points helps to determine the view distance, i.e. the maximum distance at which the lane marking can be extrapolated (see Fig. 4 bottom-right).

Additional examples of behaviour of the proposed method, in different situations can be found in our website[2].

[2]https://vicomtech.box.com/v/laneDetectionSamples

ISBN 978-90-827970-1-5 © EURASIP 2018

TABLE I
AVERAGE PROCESSING TIME (MS/FRAME) FOR THE TEST PLATFORMS.

	Total	DSRF	Stripes	Lateral	Curvature
Std. PC	2.05	0.71	0.49	0.29	0.53
Emb. PC	3.75	0.60	0.86	0.52	1.74

C. Performance

We have tested the processing time of the method in two platforms: (1) Standard PC, Intel Core i5-6500 @3.2 GHz, 8 GB RAM; and (2) embedded PC with Jetson TX2, Quad ARM A57, 8 GB RAM. Table I shows the processing time of the method and the main steps of the algorithm.

As we can see, the proposed method is a light-weight algorithm, which consumes only about 13% of CPU resources of the standard PC, and 50% of the embedded platform, which leaves plenty of room to run other processes in parallel. Additionally, the processing time is well below (2.05 ms/frame and 3.78 ms/frame for the standard and embedded PC) the real-time limit of 40 ms/frame for 25 fps cameras.

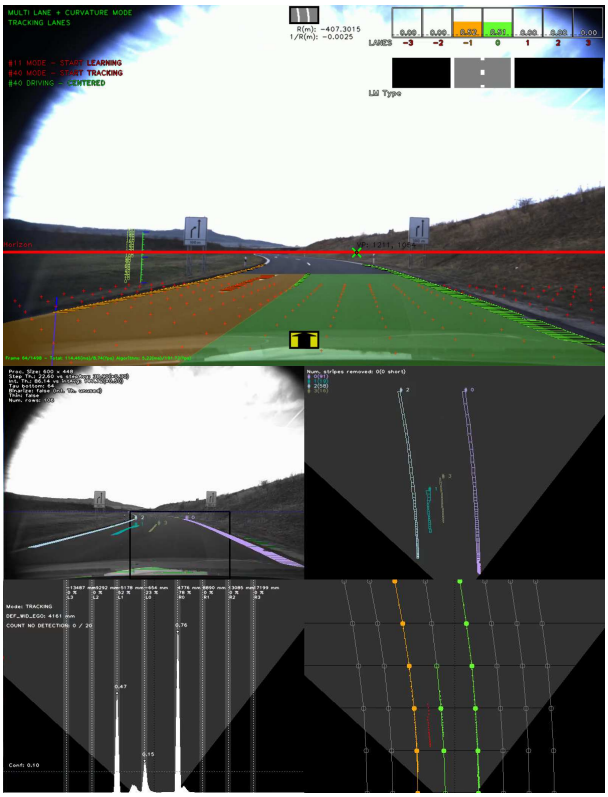

Fig. 4. Example visualization of the proposed lane detection and modeling method (upper image). Icons show the corresponding confidence of each detected lane, the type of lane markings and radius of curvature. The bottom group illustrate the algorithm steps: top-left shows the detected pairs using the DSRF, top-right is the bird's-eye view of the created Stripes; bottom-left show the histogram used for lateral tracking; and bottom-right depicts the control points and the estimated curvature for each lane marking.

VII. CONCLUSIONS

In this paper we have presented a lane detection and modeling method, which has been devised to work efficiently for in-vehicle platforms, to keep its computational load low enough to leave room for heavier algorithms such as deep learning object detection in the same processor.

The main contributions are: novel lane marking segmentation approach (DSRF), with a dynamic step filter which can detect lane markings of any width, plus an efficient row sampling technique which significantly reduces the computational load. The lane model uses constrained parabolic fitting based on stripe building and association. Lane logic is used to associate stripes to lane markings and these to lanes, allowing to detect maneouvres such as lane change, lane merge, lane split, and also define the level of curvature of lanes (individually) plus the type of lane marking (continuous and discontinuous).

Future works will include a full analysis of the computational cost of this approach while running in parallel with other methods, specially deep learning-based, in order to evaluate how different in-vehicle platforms distribute the load between CPU and GPU processors.

ACKNOWLEDGMENT

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 6880099, project Cloud-LSVA).

REFERENCES

[1] M. Bertozzi and A. Broggi, *GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection*, IEEE Transactions on Image Processing, 1998, vol. 7, no. 1, pp. 62-81.
[2] B. Southall and C. J. Taylor, *Stochastic road shape estimation*, in Proc. International Conference on Computer Vision, 2001, vol. 1, pp. 205-212.
[3] C. Corridori and M. Zanin, *High curvature two-clothoid road model estimation*, in Proc. IEEE Intelligent Transportation System Conference, 2004, pp. 630-636.
[4] Z. Kim, *Robust lane detection and tracking in challenging scenarios*, IEEE Transactions on Intelligent Transport Systems, 2008, vol. 9, no. 1, pp. 16-26.
[5] M. Nieto, A. Cortés, O. Otaegui, J. Arróspide and L. Salgado, *Real-time lane tracking using Rao-Blackwellized particle filter*, Journal of Real-Time Image Processing, 2016, vol. 11, no. 1, pp. 179-191.
[6] M. Nieto, G. Vélez, O. Otaegui, S. Gaines and G. Van Custem, *Optimising computer vision based ADAS: vehicle detection case study*, IET Intelligent Transport Systems, 2016, vol. 10, no. 3, pp. 157-164.
[7] M. Nieto, J. Arróspide and L. Salgado, *Road environment modeling using robust perspective analysis and recursive Bayesian segmentation*, Machine Vision and Applications, 2011, vol. 22, pp. 927-945.
[8] M. Nieto, L. Salgado and F. Jaureguizar, *Robust Road Modeling based on a Hierarchical Bipartite Graph*, in Proc. IEEE Intelligent Vehicles Symposium, 2008, pp. 61-66.
[9] K. Behrendt and J. Witt, *Deep learning lane marker segmentation from automatically generated labels*, in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) 2017.
[10] J. Kim and C. Park, *End-To-End Ego Lane Estimation based on Sequential Transfer Learning for Self-Driving Cars*, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017.
[11] A. B. Hillel, R. Lerner, D. Levi and G. Raz, *Recent progress in road and lane detection: a survey*, Machine Vision and Applications, 2014, vol. 25, pp. 727-745.
[12] T. Veit, J.-P. Tarel, P. Nicolle and P. Charbonnier, *Evaluation of Road Marking Feature Extraction*, in Proc. 11th IEEE Conf. on Intelligent Transportation Systems (ITSC), 2008.
[13] 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.