2010 IEEE Intelligent Vehicles Symposium
University of California, San Diego, CA, USA
June 21-24, 2010

Robust monocular lane detection in urban environments

Mirko Felisa and Paolo Zani
VisLab – Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma, ITALY
http://www.vislab.it
{felisa,zani}@vislab.it

***Abstract*— An effective lane detection algorithm is a basic, yet fundamental component of both autonomous navigation and advanced road safety systems; this paper presents an approach that produces reliable results exploiting a robust polyline matching technique. The proposed solution has been designed from the ground up so that only very limited hardware resources are required: just one camera is used, and the processing is fast enough to be compatible with mainstream DSP units.**

I. INTRODUCTION

Basic environment perception tasks, such as the detection of painted road markings, are an essential component of both intelligent vehicles and advanced road safety systems, being the primary source of information on a rapidly-changing the environment. Besides the inherent difficulties of identifying the lane boundaries, whose appearance greatly varies depending on illumination, weather and road conditions, some applications pose additional constraints on the number and type of sensors and processing resources dedicated to the detection task. For example, the 2007 DARPA Urban Challenge demonstrated the feasibility of autonomous urban navigation, but the lane-detection task was often accomplished by exploiting the presence of expensive LIDAR sensors and high-end IMUs [1], [2], while typical Lane Departure Warning systems available on the automotive market today [3], [4] are based on monocular vision setups, possibly combined with in-vehicle odometry; on the other hand, these systems can be reliably used only in much more controlled situations, like highway traffic. Developing driving assistance systems being at the same time robust, low-cost and easy to integrate represents a challenge, nonetheless road safety laws are going to push these technologies onto the market: the European Parliament recently adopted a regulation [5] making Lane Departure Warning systems mandatory on heavy-goods vehicles within just a few years, and the NHTSA will likely follow.

The algorithm presented in this paper has been designed to meet all these requirements, with the aim of handling complex urban scenarios even when running on low-end hardware. Moreover, additional information (like odometry, inertial measurements and the presence of obstacles) can be exploited when available for increased robustness and precision.

II. ALGORITHM

The overall system architecture is presented in Fig. 1: first an Inverse Perspective Mapping (IPM) transformation [6], [7] is applied to the frame grabbed by the camera, then a low-level filtering is performed to highlight the Dark-Light-Dark [8] (DLD) patterns of the image; the resulting points are grouped together, and clusters are finally approximated by continuous piecewise-linear functions. Once the low-level processing is over, the resulting segment lists are compared to existing lane markings in a tracking stage (Fig. 1), which produces a set of candidate lane markings; moreover, non-tracked segments are also analyzed to extract additional candidates. An expansion step is then performed, in order to join in any pertinent non-connected component still present. Finally each candidate is assigned a score which is tested against an acceptance threshold to produce the end result. The lane markings detection is carried out for solid and dashed markings, using slightly different algorithms, and the whole procedure is performed two times, one for single and one for double lines.

A. Low level processing

The low-level processing stage derives from the one used during the 2007 DARPA Urban Challenge, described in [9]. First DLD and DLDLD transitions (with the former corresponding to single lane marking, and the latter to double ones) are extracted from the IPM and stored in separate buffers; this operation is fast since the filtering kernel is of constant size (5 and 11 pixels respectively). As an example, values of $DLD(x,y)$ are computed as

$$\begin{aligned} DLD(x,y) = \\ \max(\min(&\mathbf{Src}[x,y]-\mathbf{Src}[x-2,y], \\ &\mathbf{Src}[x,y]-\mathbf{Src}[x+2,y]),0) \end{aligned} \quad (1)$$

After the pattern extraction a binarization step is performed (Fig. 2-c,d): this process employs a variable threshold proportional to the average luminance of the region, over a window of size 32×1 pixels (Fig. 2-e,f); this helps to reduce the effects of shadows cast by vehicles and roadside elements, like buildings, trees and guard-rails. The resulting pixels are then grouped together by a clustering algorithm which uses the expansion strategy illustrated in Fig. 3, with the processing taking place starting from the

978-1-4244-7868-2/10/$26.00 ©2010 IEEE

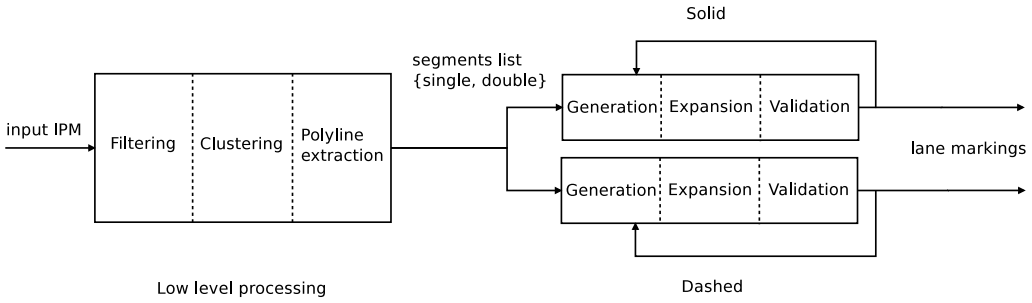


Fig. 1. Lane detection system overview.

bottom of the image; results are presented if Fig. 2-g,h. The various groups of points are then approximated using piecewise-linear functions, so that each node on the polyline corresponds to an element of the cluster, and the maximum distance between any pixel within the label and the closest segment is below a given threshold, as can be seen in Fig. 2-i,j. While different solutions to this problem exist (like the one described in [10]), the one adopted in this paper has proven to be good enough to deal with the data produced by the preprocessing stage.

B. Lane Marking Extraction

Once the set of polylines has been extracted from the image the actual lane markings extraction can take place. This process consists of three fundamental stages:

- tracking of markings detected in previous frames
- generation of new candidates
- expansion of the whole set of candidates (new and tracked)

that are carried out in that order both for solid and dashed markings, and are detailed in the following.

1) Tracking: Candidate polylines are matched against existing lane markings, and only those resulting close enough can be considered a valid correspondence. The fundamental issue to solve when performing this task is to define a distance function: a number of well-known approaches to this problem exist, like the Hausdorff [11], Frechét [12] and minimum Euclidean distances [13], but while they exhibit some interesting mathematical properties, sometimes they can produce counter-intuitive results; instead, area-based algorithms [14] seem to be more appropriate in this kind of applications: building on this idea, the criterion illustrated in the following has been devised.

Before describing the actual algorithm, the preliminary definitions of *point-segment distance*, *point-polyline distance* and *polyline-polyline overlapping area* must be introduced.

Let A be a point in the plane, $\overline{P_aP_b}$ a segment, r its support line, and $A \perp r$ the perpendicular projection of A on said line; then the distance between A and $\overline{P_aP_b}$ can be defined as

$$d(A,\overline{P_aP_b}) \stackrel{\text{def}}{=} \begin{cases} d(A,A \perp r) \text{ if } A \perp r \in \overline{P_aP_b} \\ \min(d(A,P_a),d(A,P_b)) \text{ otherwise} \end{cases} \quad (2)$$

and $A \perp \overline{P_aP_b}$ as the point on $\overline{P_aP_b}$ originating that value.

Exploiting Eq. 2, and given a point A and polyline $p = \{P_0 \dots P_n\}$, it is now possible to define the distance between them $d(A,p)$ as the length of the segment connecting A to the closest point on p (referred to as $A \perp p$ in the following). Let

$$k = \underset{i \in \{0 \dots n\}}{\arg\min}\ d(A,P_i) \quad (3)$$

it is straightforward to prove that $A \perp p \in]P_{k-1},P_{k+1}[$, with its actual value being[1]

$$A \perp p \stackrel{\text{def}}{=} \begin{cases} A \perp \overline{P_{k-1}P_k} \text{ if } d(A,\overline{P_{k-1}P_k}) < d(A,\overline{P_kP_{k+1}}) \\ A \perp \overline{P_kP_{k+1}} \text{ otherwise} \end{cases} \quad (4)$$

Given two polylines a and b the overlapping area $area(a,b)$ between them (marked in light blue in Fig. 4) is the area of the polygon constructed by adding the segments $\overline{V_0V_3}$ and $\overline{V_1V_2}$ as extra boundaries; these segments are defined as

$$\overline{V_0V_3} \stackrel{\text{def}}{=} \underset{x \in \{\overline{A_0B_f},\overline{B_0A_f}\}}{\arg\min} \|x\|_2 \quad (5)$$

with $B_f = A_0 \perp b, A_f = B_0 \perp a$ and

$$\overline{V_1V_2} \stackrel{\text{def}}{=} \underset{x \in \{\overline{A_mB_b},\overline{B_nA_b}\}}{\arg\min} \|x\|_2 \quad (6)$$

with $B_b = A_m \perp b, A_b = B_n \perp a$.

The minimum distance between the two polylines a and b can be computed as:

$$d_{min}(a,b) \stackrel{\text{def}}{=} \min(\min_{i \in \{0..m\}} d(A_i,b), \min_{j \in \{0..n\}} d(B_j,a)) \quad (7)$$

Having introduced the above definitions, the distance between two polylines a and b becomes

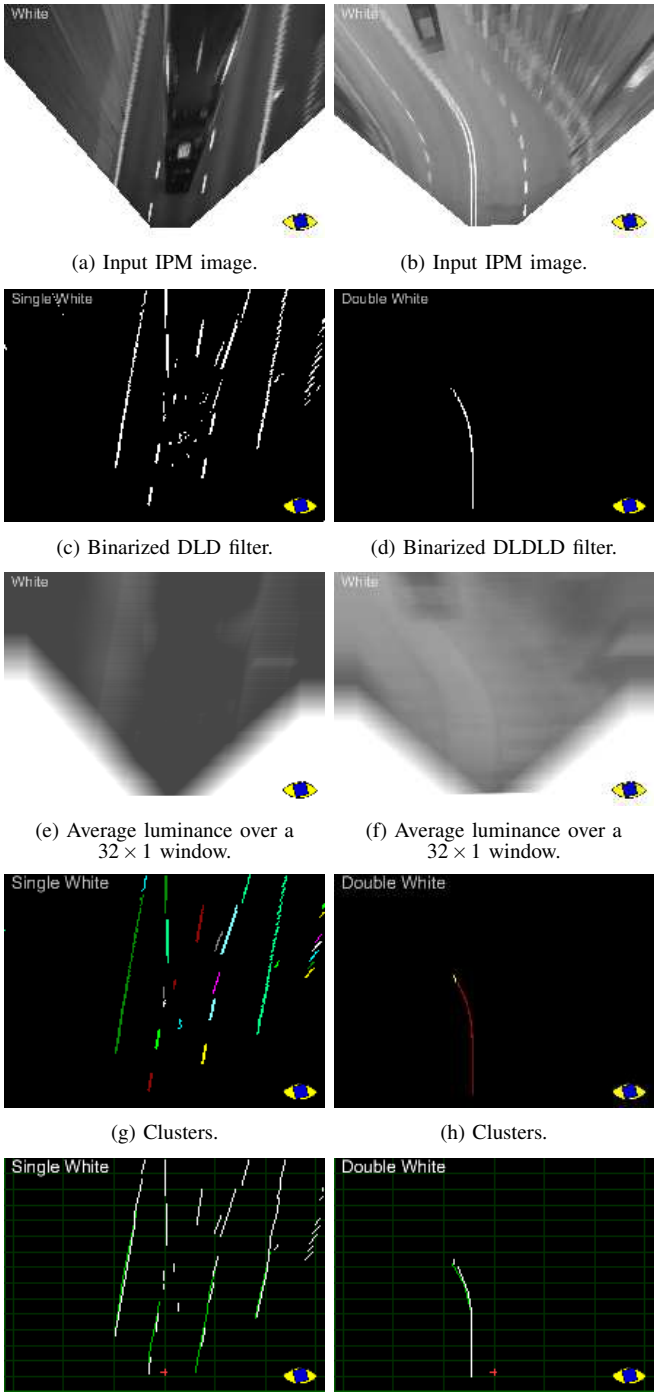
$$d(a,b) \stackrel{\text{def}}{=} \max(\frac{2 \times area(a,b)}{length(a)_{[V_0,V_1]} + length(b)_{[V_3,V_2]}}, d_{min}(a,b)) \quad (8)$$

where $length(p)_{[V_m,V_n]}$ denotes the length of the polyline portion delimited by the points V_m and V_n.

When performing tracking, each solid lane marking $ts_i \in \{ts_0 \dots ts_k\}$ identified at time $T-1$ is used to compute the score

$$score(ts_i,cs_j) \stackrel{\text{def}}{=} \frac{d(ts_i,cs_j)}{length(cs_j)}, j \in \{0 \dots h-1\} \quad (9)$$

[1] note that in case $k = 0$ ($k = n$) $A \perp p$ will simply evaluate to $A \perp \overline{P_0P_1}$ ($A \perp \overline{P_{n-1}P_n}$)



(a) Input IPM image. (b) Input IPM image.

(c) Binarized DLD filter. (d) Binarized DLDLD filter.

(e) Average luminance over a 32×1 window. (f) Average luminance over a 32×1 window.

(g) Clusters. (h) Clusters.

(i) In white, fitted polylines; in green, polylines from tracking. (j) In white, fitted polylines; in green, polylines from tracking.

Fig. 2. Two samples of the low-level processing steps: on the left, to detect single lines; on the right, to detect double ones. The IPM resolution is of 220×160 pixels.

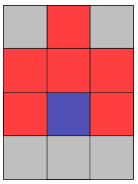


Fig. 3. Clustering algorithm expansion mask: in blue, the reference pixel; in red, the candidates for expansion. The topmost candidate helps to reduce the likelihood of cluster fragmentation due to non-continuous groups of points (e.g. because of dirty or faded lane markings).

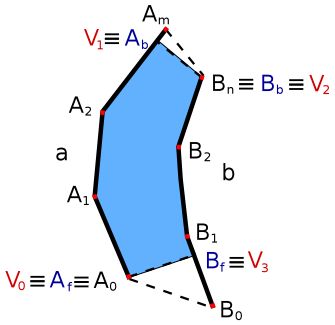


Fig. 4. Distance between polylines. In light blue, the overlapping area; in red, boundary vertexes; in blue, the projections of one polyline ends onto the other.

matching it against the candidates $\{cs_0 \dots cs_{h-1}\}$ generated by the low-level processing stage at time T; the candidate obtaining the lowest value is selected, and used in the following expansion stage.

Dashed lane marking tracking uses a different comparison criterion, since the length of candidate dashes $\{cd_0 \dots cd_{k-1}\}$ is usually too small to be reliable:

$$score(td_i, cd_j) \stackrel{\text{def}}{=} d(td_i, cd_j)^2 + x_j^2, j \in \{0 \dots k-1\} \quad (10)$$

with x_j being the x component of the first vertex of cd_j. This heuristic allows to obtain as a winning candidate a dash that is near to the lane marking to track while also being as close as possible to the camera, that is, in the IPM region that is more likely to produce accurate results.

After the first dash has been determined, it is used as a starting point to join in any other dash close to the originating lane marking (that is, td_i), iteratively building a new polyline cp. The criterion adopted to perform this task aims at isolating candidates being close both to cp and to td_i, sorting dashes according to the following comparison rule:

$$\min(cd_h, cd_k) \stackrel{\text{def}}{=} \begin{cases} cd_h \text{ if } d(cd_h, td_i) < th \text{ and } d(cd_k, td_i) > th \\ cd_k \text{ if } d(cd_h, td_i) > th \text{ and } d(cd_k, td_i) < th \\ \underset{cd \in \{cd_h, cd_k\}}{\arg\min} \; (d(cd, cp) + d(cd, td_i)) \text{ otherwise} \end{cases} \quad (11)$$

Each time a dash is added to cp the remaining ones are sorted again using Eq. 11, until no close dashes are left. To further improve the robustness of this step, dashes are joined only if they satisfy the constraints illustrated in Fig. 5, and further explained in II-B.2.

2) Expansion and validation: Tracked and non-tracked polylines are iteratively analyzed to determine whether any other compatible candidate exists; if it is found, its points are merged in, and the search continues using the resulting polyline as the new reference. For both solid and dashed lane markings, a common condition for inclusion is that the first vertex of the candidate polyline must be close to the last vertex of the reference one, as it is illustrated in Fig. 5; moreover, the orientation of the end segments must be similar (that is, the angle φ in Fig. 5 must be small), and the connection angle (θ in Fig. 5) must also be small. Dashed markings bear the additional constraint that dash lengths and pauses between dashes must be similar.

When the search is over, a score is assigned to the resulting polyline p, to determine whether it should be accepted as a valid lane marking or not. The value is computed as

$$score(p) \stackrel{\text{def}}{=} \frac{length(p)^2}{d((0,0),p)^2} \tag{12}$$

and if p has been successfully tracked, the old score is added to the current. Using this approach means that lines starting far away from the vehicle are considered valid only if they are consistently detected over a high number of frames, while close, long lines are more easily accepted.

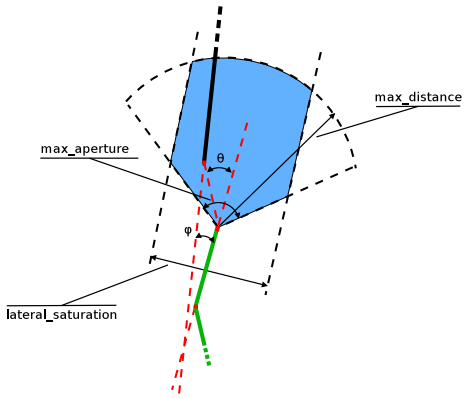


Fig. 5. Compatibility test. In green, reference marking, in black, candidate polyline to join. The candidate is considered for inclusion only if its first vertex falls inside the blue area (determined by the parameters *max_distance*, *max_aperture* and *lateral_saturation*), and the angles φ and θ are small enough.

Tracked lane markings are not discarded immediately in case of a missed detection; instead, they are kept as "ghosts" for up to 5 frames.[2]

III. Results

The algorithm proposed in this paper has been tested on image streams captured in both highway and urban scenarios, in a variety of conditions. Fig. 6 contains some samples from different image sequences, along with lane detections results.

In ideal working conditions outputs are correct (Fig. 6-a), even in case of heavy traffic (Fig. 6-b), and tracking is very stable, with lines being continuously identified for up to 600 frames. Since no predefined model of the road is enforced atypical geometries can be handled as well, like in Fig. 6-c, where lane markings leading to an intersection at the end of a downhill road are detected correctly. The adopted score criterion shows its effectiveness in Fig. 6-d: no false detection is introduced, despite the number of objects present in the scene, including a vehicle right in front of the camera, the striped barriers on the left and the guard-rail on the right. Fig. 6-e and Fig. 6-f contain the results obtained in two challengig situations, namely on a motorway with a lot of shadows, and under heavy rain at night: in both cases the algorithm correctly detects the road markings; the same happens for the scenarios depicted in Fig. 6-g,h,i. In Fig. 6-j,k,l are presented some situations which are not handled correctly: a guard-rail is incorrectly labeled as a lane boundary, a dash is identified on the side of a car and finally a strong shadow prevents the detection of a line marking.

Benchmarks have been carried out first on a system equipped with 4 GB DDR2-800 RAM and an Intel® Core™2 Quad Q6600 CPU running at 2.40 GHz, and featuring 2x32 KB L1 cache, 2x4 MB L2 cache and a 1066 MHz FSB. The compiler used is GCC version 4.4.1, with optimization flags `-O3 -fforce-addr -march=native -ftracer -floop-interchange -floop-strip-mine -floop-block -ftree-loop-distribution`. Resulting processing times range from 5 to 11 ms, with an average of 6 ms.

The same code has been run on an embedded system featuring a Texas Instruments TMS320DM642-600 600 MHz DSP with 16 KB L1 cache and up to 64 MB SDRAM, capable of 4800 MIPS. The compiler bundled with Code Composer Studio 3.3 has been used, with optimization flags `-O1`[3]. Processing times on this platform range from 45 to 70 ms, with an average of 60 ms.

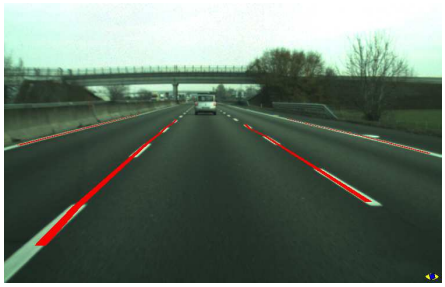
IV. Conclusions and Future Work

This paper has presented a robust lane detection algorithm capable of running at soft real-time rates on mainstream hardware. Detection performed using a single camera, exploiting an IPM transformation: this approach allows different setups, targeting different application fields; the camera can thus be placed behind a car rearview mirror, as it is commonly done, or on top of a truck cabin using lenses with much shorter focal length, as shown in Fig. 7. Lane boundaries can be successfully recognized up to 60 m even in presence of miscalibrations caused by vehicle pitching, as it has been shown in Sec. III; these results have been achieved thanks to the polyline matching and tracking criterion illustrated in Sec. II-B.

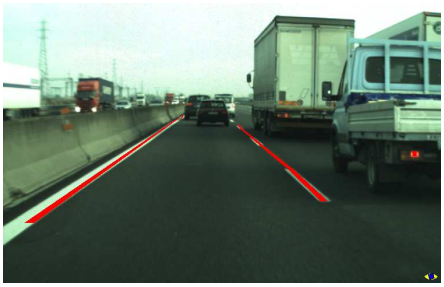
Tests have also allowed to identify some critical scenarios: obstacles and roadside structures can be sometimes incorrectly classified as line markings, and strong shadows reduce the detection rate. However, in some applications more sensors are available that can be exploited to carry

[2]the value has been determined empirically

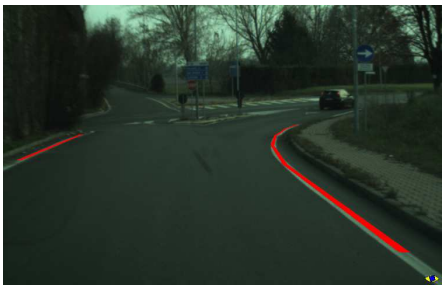
[3]higher optimization levels would produce incorrect code



(a) Highway scenario.

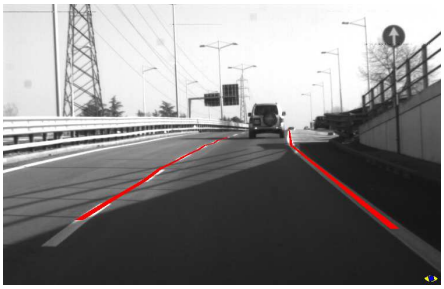


(b) Heavy traffic.

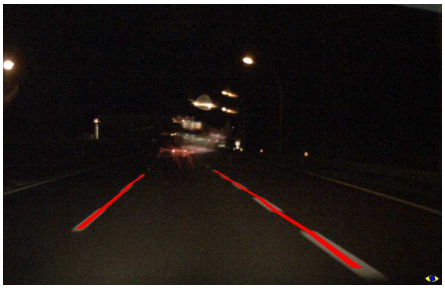


(c) A downhill intersection.

(d) Construction area, with no false detections.

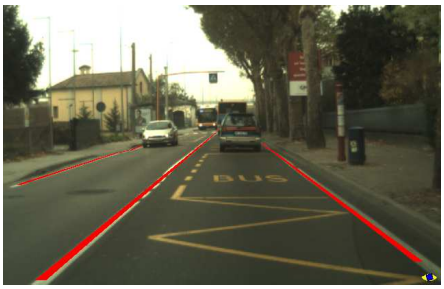


(e) Uphill motorway with shadows.



(f) Nighttime highway under heavy rain.

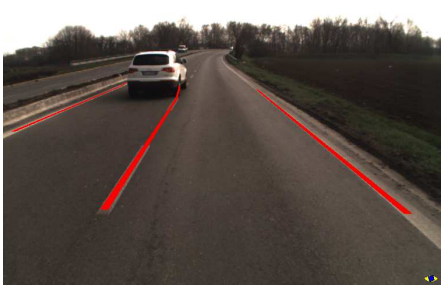
(g) Queued vehicles.



(h) Urban environment.

(i) Entering an highway tunnel at night.

(j) False detection due to a guard-rail.



(k) The side of a car interpreted as a dash.

(l) Country road with strong shadows.

Fig. 6. Some sample outputs in different situations. Image h has a resolution of 640×480 pixels and has been captured using a 6 mm lens; image j has a resolution of 752×480 pixels and has been captured using a 4 mm lens; all others have a resolution of 752×480 pixels and have been captured using a 6 mm lens.

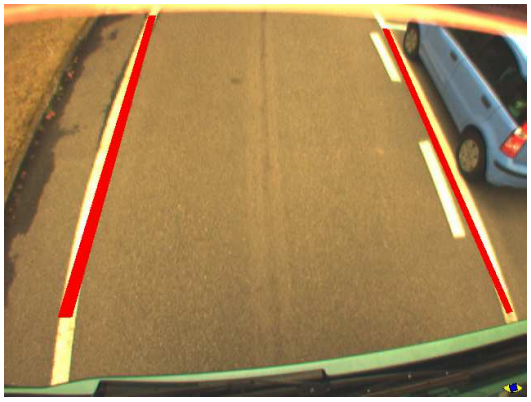


Fig. 7. Algorithm output using a 640×480 pixels camera with a 4 mm lens mounted on top of a truck cabin. Perspective and distortion are both removed using a single LUT [15], yielding to a bird's-eye view of the scene.

out additional preprocessing steps, in order to improve the precision and the detection performance:

- vehicle pitch can be corrected at runtime by means of an IMU, or using a pitch detector algorithm, such as the one presented in [16];
- the image areas corresponding to obstacles can be masked, thus reducing the chance of incurring in false detections. The obstacle detection step can exploit various techniques, such as stereovision or LIDAR-based solutions, as in [17] and [1].

Detection of colored lane markings has not been finalized yet, but the algorithm can be used as-is to handle it through two mechanisms, targeted at different scenarios:

- input images can be analyzed to extract the desired colour component (e.g. yellow or blue). This can achieved by converting the image to an appropriate color space (such as HSV), and then isolating the desired component, or performing an approximate color selection (as it is done in [9]). The resulting images (one for each color to handle) can then be separately processed, combining resulting lane markings in a single map, along with information about their color. In case of multiple detections duplicate markings have to be suppressed: for example a yellow line can be often clearly seen both in the yellow and luminance images, but only the former has to be kept.
- Alternatively, the original frame can be preprocessed in order to enhance the target color: in the resulting grayscale image higher values will correspond not only to white areas, but also to regions of the color of interest. The usual algorithm can then be applied, thus detecting white and non-white lane markings at the same time.

The first strategy clearly provides more information, that is useful when color classification needs to be exploited to take navigation decisions (as, for example, yellow lane markings delimit opposing driving directions in some countries), but requires to run the same algorithm multiple times. The second approach can conversely be useful in embedded lane departure warning systems, where the focus is on establishing lane boundaries, and analyzing multiple images is not a viable option, given the limited processing resources available on these platforms.

V. Acknowledgements

The authors gratefully thank Alberto Broggi for his support and review of this project.

References

[1] A. Huang, D. Moore, M. Antone, E. Olson, and S. Teller, "Multi-sensor lane finding in urban road networks," in *Proceedings of Robotics: Science and Systems (RSS)*, Zurich, Switzerland, June 2008.

[2] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The stanford entry in the urban challenge," *Journal of Field Robotics*, 2008.

[3] Mobileye http://www.mobileye.com.

[4] Valeo http://www.valeo.com.

[5] E. Commission, "Regulation (ec) no 661/2009 of the european parliament and of the council of 13 july 2009 concerning type-approval requirements for the general safety of motor vehicles, their trailers and systems, components and separate technical units intended therefor," available at http://eur-lex.europa.eu/smartapi/cgi/sga_doc?smartapi!celexplus!prod!CELEXnumdoc&lg=EN&numdoc=32009R0661.

[6] H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," *Biological Cybernetics*, vol. 64, pp. 177–185, 1991.

[7] M. Bertozzi, A. Broggi, and A. Fascioli, "Stereo Inverse Perspective Mapping: Theory and Applications," *Image and Vision Computing Journal*, vol. 8, no. 16, pp. 585–590, 1998.

[8] S. Nedevschi, F. Oniga, R. Danescu, T. Graf, and R. Schmidt, "Increased Accuracy Stereo Approach for 3D Lane Detection," in *IEEE Intelligent Vehicles Symposium*, Tokyo, Japan, June 2006, pp. 42–49.

[9] A. Broggi, A. Cappalunga, C. Caraffi, S. Cattani, S. Ghidoni, P. Grisleri, P. P. Porta, M. Posterli, and P. Zani, "TerraMax Vision at the Urban Challenge 2007," *IEEE Trans. on Intelligent Transportation Systems*, 2008, submitted.

[10] S. L. Hakimi and E. F. Schmeichel, "Fitting polygonal functions to a set of points in the plane," *CVGIP: Graph. Models Image Process.*, vol. 53, no. 2, pp. 132–136, 1991.

[11] J. F. Hangouet, "Computation of the Hausdorff distance between plane vector polylines," in *Procs. Twelfth International Symposium on Computer-Assisted Cartography*, vol. 4, Charlotte, Noth Carolina, USA, 1995, pp. 1–10.

[12] M. M. Frchet, "Sur quelques points du calcul fonctionnel," *Rendiconti del Circolo Matematico di Palermo (1884 - 1940)*, vol. 22, no. 1, pp. 132–136, 1906.

[13] D. J. Peuquet, "An algorithm for calculating minimum euclidean distance between two geographic features," *Computers & Geosciences*, vol. 18, no. 8, pp. 989–1001, 1992.

[14] R. B. McMaster, "A statistical analysis of mathematical measures for linear simplification," *Cartography and Geographic Information Science*, vol. 13, no. 2, pp. 103–116, Apr. 1986.

[15] A. Broggi, P. Medici, and P. P. Porta, "StereoBox: a Robust and Efficient Solution for Automotive Short Range Obstacle Detection," *EURASIP Journal on Embedded Systems – Special Issue on Embedded Systems for Intelligent Vehicles*, June 2007, iSSN 1687-3955.

[16] M. Bertozzi, L. Bombini, P. Cerri, P. Medici, P. C. Antonello, and M. Miglietta, "Obstacle Detection and Classification fusing Radar and Vision," in *Procs. IEEE Intelligent Vehicles Symposium 2008*, Eindhoven, Netherlands, June 2008, pp. 608–613.

[17] Y.-L. Chen, V. Sundareswaran, C. Anderson, A. Broggi, P. Grisleri, P. P. Porta, P. Zani, and J. Beck, "TerraMax: Team Oshkosh Urban Robot," *Journal of Field Robotics*, vol. 25, no. 10, pp. 841–860, Oct. 2008.