robotics

MDPI

Article

Design of an Embedded Multi-Camera Vision System—A Case Study in Mobile Robotics

Valter Costa [1,2,†,*], Peter Cebola [1], Armando Sousa [2,3] and Ana Reis [1,2]

1 INEGI—Institute of Science and Innovation in Mechanical and Industrial Engineering, 4200-465 Porto, Portugal; pcebola@inegi.up.pt (P.C.); areis@inegi.up.pt (A.R.)
2 FEUP— Faculty of Engineering of the University of Porto, 4200-465 Porto, Portugal; asousa@fe.up.pt
3 INESC TEC—INESC Technology and Science (formerly INESC Porto), 4200-465 Porto, Portugal
* Correspondence: ee09115@gmail.com
† Current address: Campus da FEUP, Rua Dr. Roberto Frias, 400 Porto, Portugal.

Received: 2 January 2018 ; Accepted: 22 February 2018; Published: 26 February 2018

Abstract: The purpose of this work is to explore the design principles for a Real-Time Robotic Multi Camera Vision System, in a case study involving a real world competition of autonomous driving. Design practices from vision and real-time research areas are applied into a Real-Time Robotic Vision application, thus exemplifying good algorithm design practices, the advantages of employing the "zero copy one pass" methodology and associated trade-offs leading to the selection of a controller platform. The vision tasks under study are: (i) recognition of a "flat" signal; and (ii) track following, requiring 3D reconstruction. This research firstly improves the used algorithms for the mentioned tasks and finally selects the controller hardware. Optimization for the shown algorithms yielded from 1.5 times to 190 times improvements, always with acceptable quality for the target application, with algorithm optimization being more important on lower computing power platforms. Results also include a 3-cm and five-degree accuracy for lane tracking and 100% accuracy for signalling panel recognition, which are better than most results found in the literature for this application. Clear results comparing different PC platforms for the mentioned Robotic Vision tasks are also shown, demonstrating trade-offs between accuracy and computing power, leading to the proper choice of control platform. The presented design principles are portable to other applications, where Real-Time constraints exist.

Keywords: camera as a sensor; real-time vision; robotic vision; lane tracking; signalling panel recognition; zero copy one pass

1. Introduction

Using cameras as rich sensors, with data coming from vision systems, is a particularly important part of many real world robotic applications, be it by assisting people in their daily tasks [1] or in vision perception related to mobile platforms [2]. One very relevant area of application is autonomous vehicles, representing a major innovation for the automotive industry that, in turn, results in a great economic impact world-wide. In such a dynamic context, work and research in this area has grown greatly over the last years.

This topic has even raised interest within the robotics community and was the target focus of many robotic competitions around the world [3], such as the Audi Autonomous Driving Cup [4] or the Portuguese Robotics Open [5].

An autonomous intelligent vehicle has to perform a number of tasks, sometimes in a limited amount of time [6]. This involves being able to identify and track road lanes, being able to process traffic lights and road signs, and being consistent at identifying and avoiding obstacles [7–9].

"Robot vision is not just computer vision. While robotics poses several challenges for visual processing, for example, three-dimensional (3-D) operation and performance requirements, there is an aspect in particular that makes robot vision special and distinguishes it from mainstream computer vision." [10]. Although many solutions exist, one of the most common approaches to tackle these problems is the use of one or more cameras on the robot's body.

Usually, when extracting information from an image, one holds some prior knowledge on the state of the system (that is, some notion on the state of the surrounding environment) and, therefore, more detailed measurements can be obtained. Often, however, in autonomous driving, little information is known about the environment because no previous or overall structure is present [11]. Therefore, it is not only important to get accurate results, but also to do it in an adequate timing—real-time. Real-time is often defined as the sufficient time the robot needs to process external data and actuate without causing any harm to himself or to others. The presented ideas lead to a set of processing deadlines per task, as defined in the soft real time literature [12]. Furthermore, using several cameras involves specific challenges that must be addressed to reduce the global amount of computing power needed for processing—this is especially relevant for battery operating processing platforms that are common in mobile robotics.

Robotic vision naturally inherits the characteristics from real-time vision, considered as having soft real-time constraints, and encompasses two kinds of applications: applications where 3D information is relevant and perception of flat scenes. A known real-time vision motto is "zero copy, one pass" technique as proposed by Sousa et al. [13] and used by Costa et al. [14]. Using this technique, the input image should be read only once, taking all the necessary information (which is organized into smaller data structures). This approach guarantees that the objective is reached without creating copies of the input image thus saving processing time.

The presented research uses a multi-camera vision system to tackle two classes of challenges in mobile robotics: recognition of a "flat" signal while tolerating distortions, and white lines detection and tracking that need 3D reconstruction, using the "zero copy one pass" motto [15]. The proposed algorithms optimizations and the tests performed in four hardware platforms culminate in generic design principles for robotic vision tasks where real time requirements are demanded.

The remainder of this paper is organized as follows. In Section 2, a literature review is performed, leading to the identification of the research issues, proposal, and research goals. In Section 3, all the steps and methods followed in designing the algorithm are presented. Section 4 presents the real time results obtained from the algorithm. Finally, Section 5 draws conclusions from the obtained data.

2. Literature Review

To assess the problems associated with Real-Time programming and computer vision, a literature review is performed, focusing in two important classes of tasks for autonomous robots, here exemplified as: (i) lane detection/tracking algorithms; and (ii) semaphore/sign recognition.

2.1. Lane Tracking

Road perception is a major task in an autonomous driving robot. Several strategies can be employed, resulting in different trade-offs between accuracy and computational cost. Ros et al. have proposed offline-online strategies with a well defined trade-off between accuracy and computational cost to the perception of the real world [16]. A real time approach to lane marker detection in urban streets, proposed by Aly [17] and capable of running at 50 Hz, is the generation of the top view of the road, filtering it with selective oriented Gaussian filters, and using RANSAC line fitting to give initial guesses to a new and fast RANSAC algorithm, for fitting Bezier Splines. Other authors like Sotelo et al. centred their detection algorithm is the HSI colour segmentation, with the key difference of defining a lane "skeleton": a curve representing the middle line of the road. This "skeleton" is estimated using a 2nd-order polynomial parabolic function, and subsequent estimations are then calculated with the help

of a Kalman filter [18]. A real-time lane detection invariant to illumination for lane departure warning system, based on vanishing point calculation, running at 33 ms, was proposed by Son et al. [19].

Another algorithm, proposed by Thorpe et al. [20], is SCARF: after a step of preprocessing with sub-sampling and applying a mean filter, the pixels of the image are separated into sets of similar colours using mean clustering, using on-road and off-road sample sets to compute multiple Gaussian colour models. Then, with the application of a matched filter, a road or intersection is selected from a list of candidates. Some problems with this approach include difficulties in the classification step if there aren't distinguishable colours between on-road and off-road samples and control instabilities as the robot speed increases.

Another technique useful to obtain relevant features for road estimation algorithms is the Inverse Perspective Mapping (IPM), as proposed by Muad et al. and Miguel et al. [21,22]. IPM will be further explained in this work, with more detail, in Section 3.1.1. IPM is a geometrical transformation technique that projects each pixel of the 2D camera perspective view of a 3D object and re-maps it to a new position, constructing a rectified image on a new 2D plane. Mathematically, IPM can be described as a projection from a 3D manifold, $W = \{(x, y, z)\} \in E^3$ (real life input space) onto a 2D plane, $I = \{(u, v)\} \in E^2$. The result is the desired bird's eye view of the image, thus removing the perspective effect.

This is achieved using information from the camera's position and orientation in relation to the road.

2.2. Semaphore/Traffic Light Recognition

Usually, the semaphore/traffic signs recognition algorithms comprise two tasks: (i) detection; and (ii) classification.

One well known technique is colour segmentation. A transformation to enhance red, blue, and yellow colours in the RGB colour space was presented by Ruta et al. in [23]. Additionally, they have proposed a variant of the Distance Transform (DT) called Colour Distance Transform (CDT). The main idea of CDT is the definition of a smooth distance metric to be used on the comparisons between the previously stored template and raw input image. For the traffic sign detection task, Dalal et al. in [24] have presented seven colour representations: the plain channels R, G, B, the normalized channels $r = R/S$, $g = G/S$, $b = B/S$, and the greyscale channel $S/3$, where $S = R + G + B$. For the classification stage, the authors used a Linear Discriminant Analysis (LDA), and unimodal Gaussian probability densities. For the detection phase, Maldonado-Bascon et al. in [25] have used a threshold in the HSI colour space (Hue Saturation Intensity) for the chromatic signs and, at the same time, the white signs are detected using achromatic decomposition. Then, the recognition phase uses linear Support Vector Machines (SVM) for shape recognition and SVMs with Gaussian kernels for recognition.

The main drawback of using HSI for colour segmentation is the high computational cost due to its non-linear formula, as demonstrated by Escalera et al. in [26]. To improve the time performance and accuracy, Zaklouta and Stanciulescu [27] have proposed an adaptation to the colour enhancement stage and improved the classification stage using a HOG-based SVM detector (Histogram of Oriented Gradients).

Other computational expensive solution to recognize signs is the use of the Fast Fourier Transform (FFT) as used by Gil-Jiménez et al. in [28]. The classification phase involves the usage of the FFT transform to generate signatures and the matching is performed by comparing the signatures of the template and detected blob. The use of FFT makes the matching method very robust to rotation and deformation.

2.3. Research Issues

From the literature review, it is possible to identify some issues:

- Little information on results about robotic vision systems under real time constraints is presented.

- The trade-off results relating accuracy and characterization of global execution times are not commonly found.
- There is a lack of information about the hardware/software platforms used to obtain results, mainly their specifications.

2.4. Proposal and Research Goals

The presented work aims to minimize the issues listed above, mainly the addition of information about results to the area of real time robotic vision, identification on trade-offs and comparisons between real computing platforms.

Using as motivation the PRO Autonomous Driving competition, the goal of the current research is the search of Real-Time Artificial Vision approaches that are applicable to real robotics problems, such as:

- Understanding the Robotic Real-Time Vision design principles;
- Understanding the Robotic Real-Time Vision paradigm as a trade-off—time versus accuracy;
- Identifying design strategies for the development of an algorithm capable of running on low computational power, battery operated hardware to tackle the autonomous driving tasks with adequate accuracy;
- Demonstration of the importance of the reduction of data to process (as in "zero copy one pass" principle);
- Identifying the information of the Signalling Panel as an example of perception of a flat scene;
- Finding, tracking and measuring angles and distances to track lane;
- Characterization of execution times in different processes; and
- Discussion of platform choices.

3. Materials and Methods

The methodology is based in an analysis of the PRO Autonomous Driving competition, with the test robot being used in the before-mentioned competition and in four tested hardware platforms.

The test robot, Figure 3a, uses a differential steering and no mechanical modifications on the robot platform were made throughout all tests.

The sensory system is vision-based and is made up of two PLAYSTATION Eye™("PSEye") running with a resolution of 320x240 @ 30Hz (wide lens mode), kept constant. For testing purposes, video footage was recorded to create a relevant dataset to be used in all platforms under test (using the same data allows meaningful comparisons of issues). The recording of this dataset was made to ensure that most pertinent tests are performed.

The on-board high level computing platform is changeable, communicating with motor driver via USB connection. Two vision tasks are presented as examples: (i) "flat" scene interpretation; and (ii) 3D scene perception (Figure 1).

- Task (i)—track following—3D recognition and tracking of the several white lines of the road (dashed or continuous) on black background;
- Task (ii)—signalling panel recognition—identification of one signal from a set of 6, with characteristic colours and shapes, occupying a large portion of the image. The region of interest occupies about about 30% of the total image area.

Firstly, the algorithms in their original form for both vision tasks are briefly presented and their improvement is discussed.

Results include comparative results of execution times for both algorithms for several computing platforms.

The conclusions include a systematization of the design guidelines for algorithm adaptation and discussion of other relevant results.

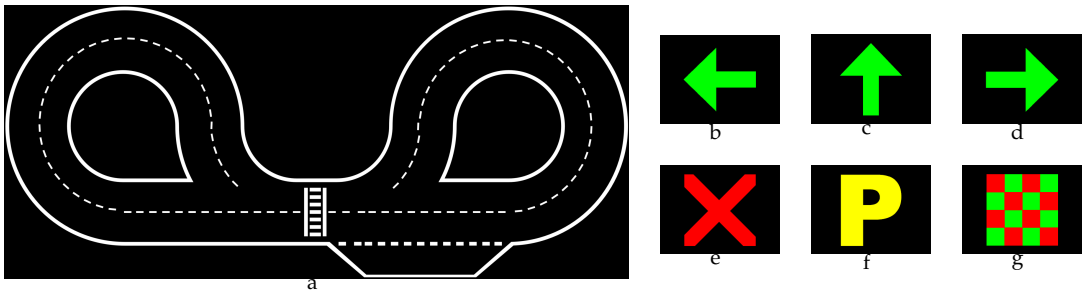


Figure 1. Track (**a**); and signalling panels used in the 2016/2017 autonomous driving competition: (**b**) green turn left; (**c**) green go forward; (**d**) green turn right; (**e**) red stop; (**f**) yellow parking; and (**g**) green and red checkers flag.

3.1. Lane Tracking Algorithm

The tracking algorithm used, represented in a flowchart diagram, is exhibited in Figure 2. The first step is to capture a RGB frame from the camera. Then, a point selection is performed in the greyscale space, extracting only the points that most likely form a line. The third step is the removal of the distortion due to the lens used (which raises the field of view of the camera). Next, for each of these points the Inverse Perspective Mapping (explained in the following subsection) is applied, and, after this step, the Probabilistic Hough Transform is employed to detect the lines. From the detected lines returned by the transform, only the "best" one is retained (the line that has the minimum distance between the actual line and the one before).

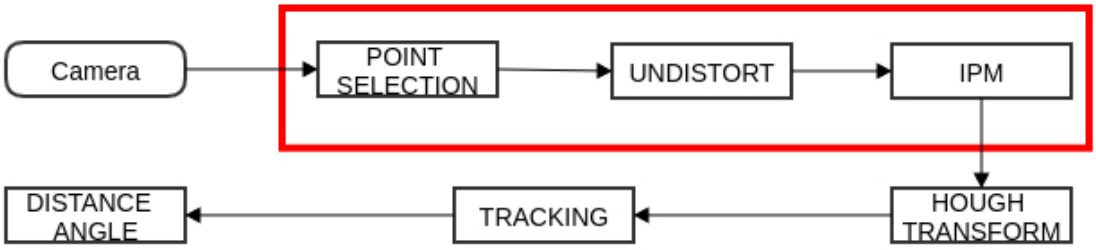
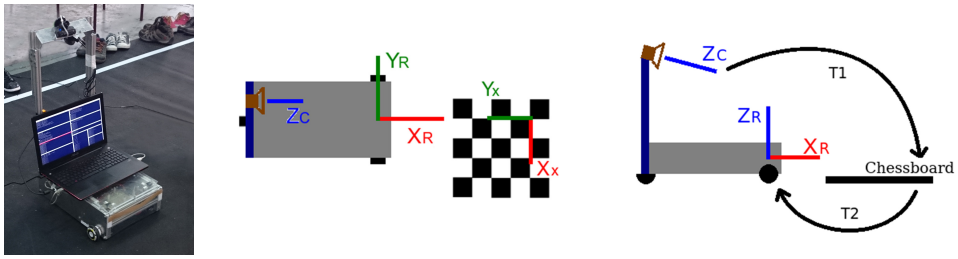


Figure 2. Block diagram for the tracking algorithm.

3.1.1. Inverse Perspective Mapping (IPM)

IPM is a geometrical transform that projects each pixel from the 2D view of 3D objects with perspective and maps them to a new position, building a new image on a new inverse 2D plane (this perspective is normally called bird's eye view). In this case, the strategy used was the calculation of the transform from the camera referential to the chessboard referential (T1), then calculating the transform from the chessboard to the robot referential (T2), as seen in Figure 3c. Figure 3b shows the setup used to calculate T1 and T2.



a b c

Figure 3. Real robot (**a**) used on the 2016 and 2017 Portuguese autonomous driving competition; and IPM-Transformations used: top view of the scheme used to calibrate IPM (**b**); and side view of the scheme used to calibrate IPM (**c**).

T1 was obtained using the camera extrinsic parameters and T2 was calculated measuring the distances between the chessboard/robot references and the rotation matrix between them. To apply the IPM transform, it is necessary to perform the transformation described in Equation (1). The remaining equations, Equations (2)–(4), are intermediate steps leading to IPM transform:

$$\begin{bmatrix} X \\ Y \\ Z \\ w \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & -x \\ p_{21} & p_{22} & p_{23} & -y \\ p_{31} & p_{32} & p_{33} & -1 \\ a & b & c & 0 \end{bmatrix}^{-1} \begin{bmatrix} -t_1 \\ -t_2 \\ -t_3 \\ -d \end{bmatrix} \tag{1}$$

$$P = KT2_R T1_R = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \tag{2}$$

$$t = KT_t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{3}$$

$$K = \begin{bmatrix} \alpha_x & \beta & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$aX + bY + cZ + d = 0 \tag{5}$$

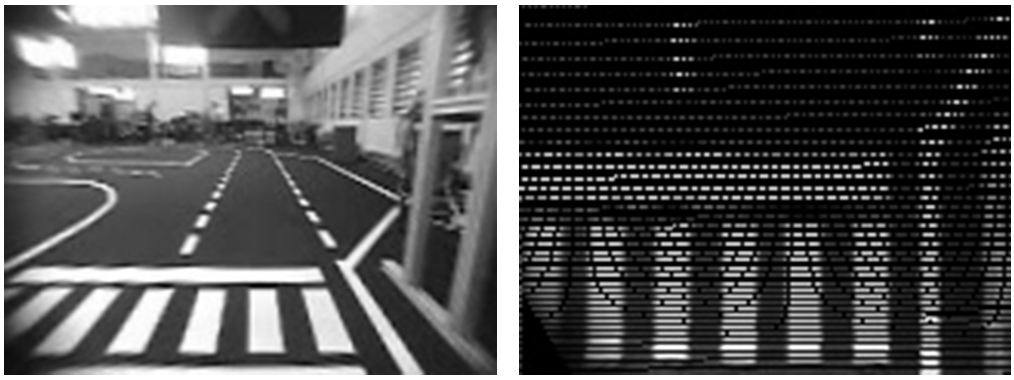
in which α_x and α_y are the focal length in pixels along x and y, respectively, x_0 and y_0 the principal point coordinates in pixels and β the skew factor in matrix K. $T1_R$ and $T2_R$ are the rotation matrices from the transformations $T1$ and $T2$. x and y are the coordinates from the input image pixel. T_t is the vector of translation resulting of the $T1$ and $T2$ multiplication. a, b, c, d define a geometric plane for the application of the IPM (Equation (5)). X, Y and Z are metrical coordinates on the robot's referential.

After we obtain X and Y (the Z coordinate is irrelevant to this case and can be discarded), to achieve a visual representation of the IPM, the last step is to map these points into an image (Equations (6) and (7)):

$$y_i = H - \frac{H \times (X - x_{min})}{x_{max} - x_{min}} \tag{6}$$

$$x_i = \frac{W}{2} - \frac{W \times Y}{y_{max} - y_{min}} \tag{7}$$

where H and W are the height and width in pixels of the window used to map the points. x_{min}, x_{max}, y_{min}, y_{max} define a box in meters on the robot's referential. An example of the application of the IPM can be seen in Figure 4.



a b

Figure 4. Example of the IPM transform: (**a**) the original image taken from the camera undistorted; and (**b**) image obtained by the IPM transform with a defined window of 2 m width and 1.5 m height.

This algorithm has the advantage of allowing, from the input image, the direct calculation of the position (x,y) in meters on the robot's referential. Another advantage is the possibility to delimit a window of interest on the robot's referential, allowing an optimized processing of the relevant information.

3.1.2. Tracking Lines Algorithm

After the IPM transform, the next step is the detection of the lines present on the IPM transformed image. The strategy employed is the application of the Hough Transform variant for line detection. This transform is available in OpenCV in two alternatives: *HoughLines()* and *HoughLinesP()*. The major difference between them is on the choice of the pixels used in the calculation of the Hough space. The first one resorts to every pixel on the image, while the second uses a set number of random pixels to calculate the transform, making it more faster than the previous one mentioned.

The Hough Transform function returns a vector of lines detected. After that, the algorithm chooses the "best" line from the received vector, which is the line nearest to the previous one detected (inside a limited range). If no line fulfils this condition, the previous one is maintained. The last step is to calculate the distance from the robot's referential to the point chosen. Because the IPM strategy was used, this step is simplified, making its computation direct.

The algorithm represented in Figure 2 (from now on denominated as "Fast version"), was designed under the real-time image motto: "zero copy one pass"—meaning that, in a single pass, all relevant information is extracted, with no need for creating kernels or auxiliary images. However, a previous attempt ("Slow version") was tested, and is as follows: after receiving a RGB frame and converting it to a greyscale representation, a Gaussian filter is applied to remove noise. With the image preprocessed, the lens distortion is corrected with IPM. Finally, the Hough line transform is used, enabling the retrieval of the distance and angle needed for the tracking. A full time-performance comparison between these two approaches is performed in Section 4.

3.2. Signalling Panel Recognition

The detection of the panel is achieved via another camera installed on the robot (pointed up). The algorithm was also designed under the "zero copy one pass" principle (the tasks inside the red bounding box in Figure 5 are computed in a single pass), and this version of the program is also

labelled "Fast version". The competition ruling [29] defines a specific dataset of panels used, shown in Figure 1.

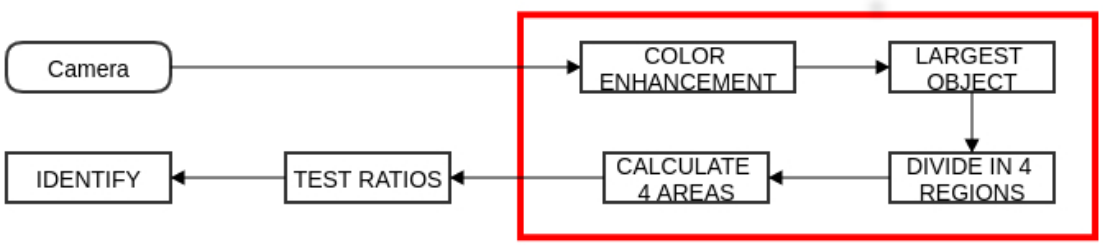


Figure 5. Block diagram for the signalling panel recognition algorithm.

The signalling panels are characterized by three main different shapes (arrow, cross and the letter "P") and colours (red, yellow and green). A block diagram representing the main stages of the semaphore recognition algorithm can be seen in Figure 5. Based on that information, for each RGB pixel x, the algorithm uses a colour segmentation strategy to enhance red (Equation (8)), green (Equation (9)), and yellow (Equation (10)) colours where $S = x_R + x_G + x_B$. Then, the object that possess the maximum area is selected. Next, the object is divided in four quarters (Q1, Q2, Q3 and Q4; see Figure 6), and the area for each one is calculated. Finally, a decision tree is used to recognize each panel by testing ratios between the quarters. In the same way as in the lane tracking algorithm, the first version of the signalling panel recognition algorithm ("Slow version") was tested and all of the tasks were applied to the full size images with no reduction of information between tasks. The "Fast version" takes advantage of design principle "zero copy one pass" and a full comparison between the two versions is presented in Section 4.

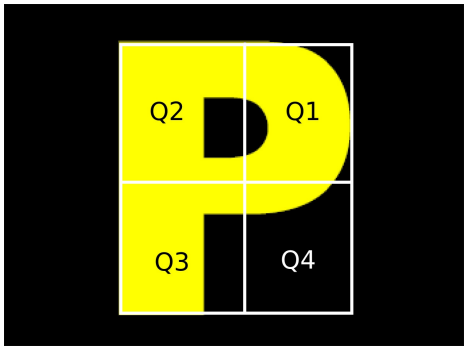


Figure 6. Example of a signalling panel divided into quarters.

$$f_R(x) = max(0, min(x_R - x_G, x_R - x_B)/S) \tag{8}$$

$$f_G(x) = max(0, min(x_G - x_R, x_G - x_B)/S) \tag{9}$$

$$f_Y(x) = max(0, min(x_R - x_B, x_G - x_B)/S) \tag{10}$$

The decision tree used to recognize each signalling panel is described in Algorithm 1.

Algorithm 1 Signalling Panel Recognition Algorithm

procedure RECOGNIZEPANEL
//Input: RGBImage
//Output: PanelString
$blob \leftarrow$ search for blob in $imageRaw$ using acceptable RGBRegion
$globalArea \leftarrow$ calculate area in $blob$
if $globalArea < minimunArea$ **then**
return "No Semaphore"
else
$q1area \leftarrow$ calculate area in $blob$
$q2area \leftarrow$ calculate area in $blob$
$q3area \leftarrow$ calculate area in $blob$
$q4area \leftarrow$ calculate area in $blob$
if $0.9 < (q1area + q2area)/(q3area + q4area) < 1.1$ & $0.9 < (q2area + q3area)/(q1area + q4area) < 1.1$ & $0.9 < q2area/q4area < 1.1$ & $0.9 < q1area/q3area < 1.1$ **then**
return "Stop"
else
if $0.9 < (q1area + q4area)/(q2area + q3area) < 1.1$ & $0.9 < q2area/q1area < 1.1$ **then**
return "Go Forward"
else
if $0.9 < (q1area + q2area)/(q3area + q4area) < 1.1$ & $(q2area + q3area) > (q1area + q4area)$ **then**
return "Turn Left"
else
if $0.9 < (q1area + q2area)/(q3area + q4area) < 1.1$ & $(q2area + q3area) < (q1area + q4area)$ **then**
return "Turn Right"
else
if $(q1area + q2area + q3area)/(q1area + q2area + q3area + q4area) > 0.75$ **then**
return "Park"

return "No Panel"

4. Results and Discussion

This section presents the results obtained from the application of real time vision guidelines. Tables 1 and 2 show a comparison between the same tracking algorithm, one version using the real-time vision premise "zero copy, one pass" (Fast) and another version without (Slow). Table 3 presents the real time results for the signalling panel recognition algorithm. Additionally, two histograms that juxtapose the four platforms time results for the tracking and panel recognition algorithms are presented. The accuracy of the distance and angle measures obtained from the tracking system is displayed in Figure 8, and some image results for the tracking lines algorithm are also presented. Regarding to the signalling panel recognition algorithm, some results taken in a laboratory environment are presented in Figure 9.

Table 1. Average execution time of IPM algorithm.

PC	Mean of Slow IPM Time (ms)	Mean of Fast IPM Time (ms)	Ratio (S/F)
Asus ROG	80.74	0.4854	166.34
EeePC	587.7	3.100	189.60
RaspberryPi2	1344	7.174	187.41
RaspberryPi	2970	15.52	191.31

Table 2. Average execution time of the tracking algorithm shown in Figure 2.

PC	Mean of Slow Tracking Time (ms)	Mean of Fast Tracking Time (ms)	Ratio (S/F)
Asus ROG	84.56	1.424	59.36
EeePC	604.2	8.599	70.25
RaspberryPi2	1366	22.98	59.46
RaspberryPi	3209	45.60	70.37

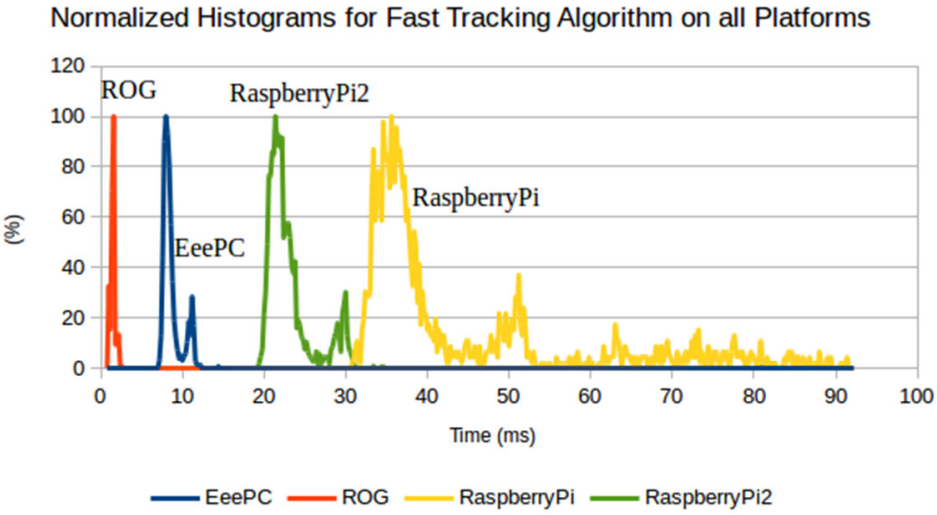
Table 3. Average execution time of the signalling panel recognition algorithm.

PC	Mean of Slow Panel Recognition Time (ms)	Mean of Fast Panel Recognition Time (ms)	Ratio (S/F)
Asus ROG	0.5830	0.3533	1.650
EeePC	11.28	4.084	2.761
RaspberryPi2	13.20	5.301	2.490
RaspberryPi	73.99	24.42	3.030

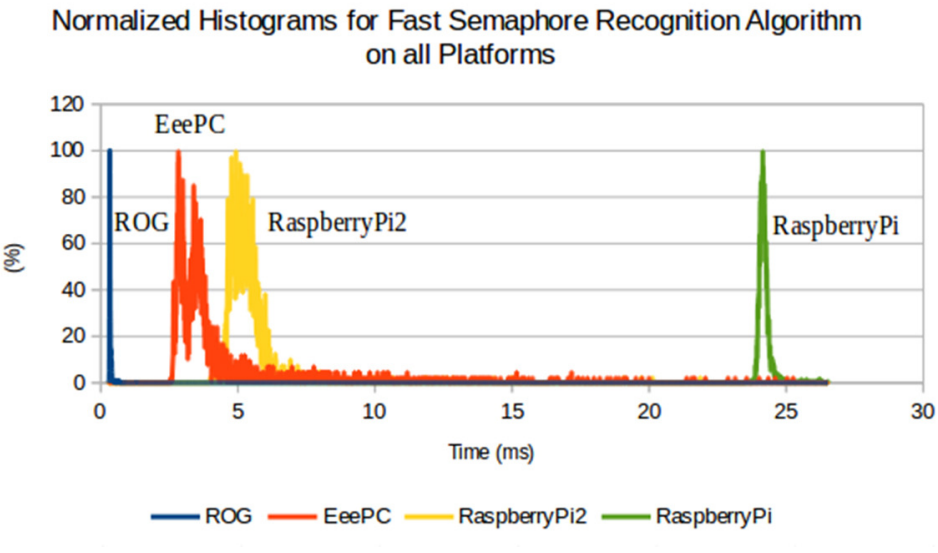
Looking at Tables 1 and 2, it is clear that there is significant time reduction between the different platforms on the IPM and tracking algorithm. In the case of the Asus ROG, the execution time became 59 times faster and on the EeePC and in the Raspberry was 70 times faster. The results in the signalling panel recognition shown in Table 3 also present a satisfactory real-time vision performance in this context. The measured time does not take into account the camera data transfer time on the USB port.

The histograms presented in Figure 7 are relative to the measured time obtained, both in the fast version, for the tracking algorithm, Figure 7a; and for the measured times for the signalling panel recognition algorithm, Figure 7b. To test the tracking and panel recognition algorithms, two representative real case scenario datasets from the 2016/2017 PRO autonomous driving competition were used, which were recorded and replayed similarly on all machines. The number of samples used to draw the histograms was 5000 in both algorithms (in all platforms), and the same implementation of the algorithms was used. Furthermore, the histograms are individually normalized.

From the histograms in Figure 7a,b, it is possible to see that the peak in each platform appears around its mean. However, in Figure 7a, another peak appears in all platforms. This can be explained by the presence of a cross-walk (zebra crossing, as in Figure 4a) on the dataset used (real case scenario regarding the 2016/2017 PRO autonomous driving competition) to test the tracking algorithm. In this region of the track more lines appear, therefore more information needs to be processed.



a



b

Figure 7. Normalized time histograms for the tracking algorithm (**a**); and signalling panel recognition algorithm (**b**).

Another important result is the accuracy of the distance and angle returned by the tracking algorithm. The accuracy results were obtained by placing the robot in different known positions on the track, measuring the real distance and angle between the robot's referential and the intersection point to the right line. Figure 8a,b show a comparison between the real-life measurements and the ones obtained from the algorithm. The blue line represents the real-life measurements, assumed to be the "ground truth" (ignoring errors in the manual measurement). By examining Figure 8a,b, it is noticeable that the obtained distance and angle are precise enough for the application at hand, showing a distance error below 0.03 m, and an angle error below five degrees.

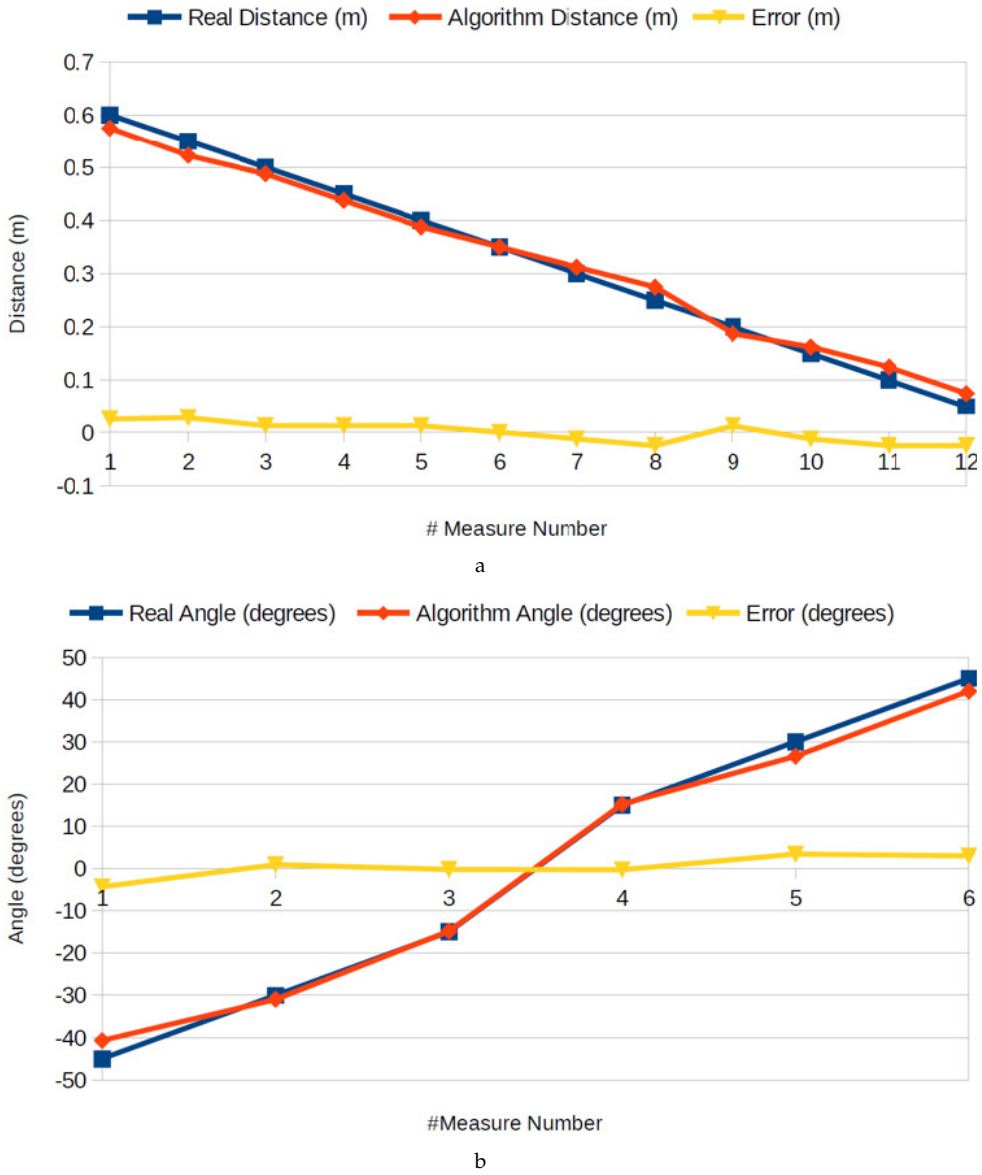


Figure 8. Distance accuracy tests results and related absolute error: (**a**) angle accuracy tests results; and (**b**) related absolute error .

In Figure 9b, detected lines returned by the Probabilistic Hough Lines Transform function drawn on top of the IPM image (Figure 9c) in the slow version, and fast version Figure 9d are shown.

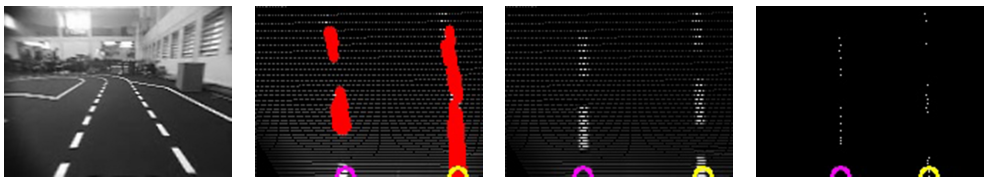
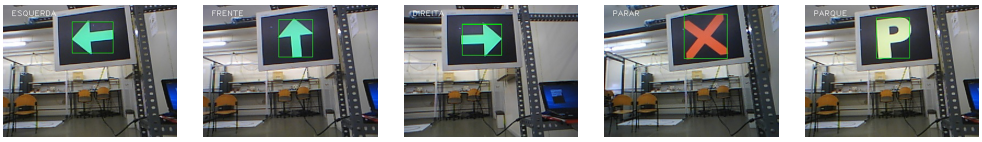


Figure 9. Undistorted image from the track (**a**); probabilistic Hough transform detected lines drawn on top of the IPM image using slow tracking (**b**); IPM transform image with slow tracking (**c**); and IPM transform image with fast tracking (**d**).

The yellow and purple circles are the result of the tracking lines system (A video showing the results of the proposed tracking system can be found in Supplementay Material). The distance and angle measures needed are calculated from the yellow/purple circle near bottom part of Figure 9d.

The signalling panel recognition algorithm was also tested into the 2016 and 2017 PRO. In the context of the competition, this technique is precise enough, ensuring zero false positives and zero false negatives in both tests made at the competition (A video showing the results of the proposed signalling panel recognition system can be found in Supplementay Material).

In Figure 10, a laboratory test is shown for each panel used in the competition.



a b c d e

Figure 10. Example of the application of the proposed signalling panel recognition algorithm: (**a**) turn left panel successfully recognized; (**b**) go forward panel successfully recognized; (**c**) turn right panel successfully recognized; (**d**) stop panel successfully recognized; and (**e**) parking panel successfully recognized.

The characteristics of the PCs used on the tests are detailed in Table 4.

Table 4. Specifications of the platforms used in the tests of the proposed algorithms.

Model	CPU	RAM	OS
Raspberry Pi B 512MB	ARM1176JZF-S 700 MHz	512 MB	Raspbian
Raspberry Pi 2 B	900MHz quad-core ARM Cortex-A7 CPU	1GB	Raspbian
Asus EeePC 1005HA	1.66GHz Intel Atom N280	2GB	Xubuntu 14.04 LTS
Asus ROG GL550JK	Intel Core i7-4700HQ 2.5GHz	16GB	Xubuntu 14.04 LTS

5. Conclusions and General Design Guidelines

This article presents and debates design principles for Robotic Vision Systems targeting real world autonomous mobile robotics running software under soft real time constraints. A common application example is autonomous driving and this article presents a scaled down car, in this case, a robotic platform for the autonomous driving competition of the Portuguese Robotics Open. In many applications, and particularly in the presented robot, the perception is solely vision based and typical robotic vision tasks include interpretation of both flat and 3D scenes coming from several cameras. Considering the presented case study, the recognition of a "flat" scene corresponds to identifying a Signalling Device and the perception of a 3D "scene" relates to lane tracking. The utilized robot has two cameras and runs perception and high level control software in a changeable platform of PC class. The tests were run using several distributions of the Linux operating system. The two mentioned vision tasks (signalling recognition and track following) are presented and then adapted/improved to follow interesting robotic real time vision guidelines.

The proposed general guidelines are:

(1) Use the structure of the problem to define the working limits of the algorithms (the algorithm does not have to work outside of the "rules" of the problem).
(2) Try to recognize Real Time limits for your problem by finding lowest safe working frequency (for both the perception and the control loops); additionally, try to minimize latency of the closed loop information travel inside the system from perception to control (including decision)—the perception, decision and control loop.

(3) Plan debug and visualization features to run at a lower rate than other tasks, preferably in a separate task; if this is complex, preferably run debug and visualization after the perception, decision and control loop.
(4) Robotic vision is information intensive: try to use as low resolution as possible in the initial image (naturally without loosing relevant data).
(5) Design algorithms so that information is reduced as quickly as possible, particularly do not visit same data structure more than once and do not copy raw data; if possible, calculate all results with a single pass on the data structure (in short, the motto of "zero copy, one pass").
(6) Presently, real world robotic vision tasks mostly deal with semi-structured environment and practitioners recognize that there is a trade-off between accuracy and low computing power.
(7) Use high level programming languages and Consumer Of The Shelf (COTS) hardware to allow choosing from many different hardware platforms that are currently available from the results of the profiling (execution times of the application).
(8) The practitioner should acutely be aware that running hardware more powerful than necessary or an algorithm doing excessive processing above what is strictly necessary will drain valuable energy from the batteries; separate and run tasks only as needed; use profiling tools to optimize relevant algorithms.
(9) Plan and run tests for the safety, security and Real-Time compliance that need to be addressed.

From the presented guidelines, the signalling panels are recognized without any perspective correction and the decision tree algorithm is run from data coming from half of the pixels of the image. All the images on the dataset with 5000 frames were recognized correctly (no missing values, no wrong recognitions). Naturally, this result arises from the definition ("easiness") of the problem and if tested elsewhere the presented algorithm will inevitably fail (more perspective, further away).

Under the same guidelines, the 3D perception used only the IPM algorithm for meaningful pixels, such as starting edges (left to right) of the lines of the track. The accuracy of the proposed method yields about 3 cm in distance tracking and five degrees in angle measurement, for the dataset of 3 min of video coming from a real driving situation, containing frequent purposeful oscillations on the track.

The improved "fast" algorithms are comparable to previous "slow" ones and prove to be better than most of the state of art for this application, such as Ribeiro et al. in [30]. Execution time improvements range from almost 200 times to 1.6 times (time_slow_algorithm/time_fast_algorithm) for the same hardware, with the same input videos. For the shown platforms, improvements seem to be even more relevant in low processing power hardware platforms than for hi-end computing platforms. Tests on the well known Raspberry Pi 1 displayed the largest improvements for the presented algorithms—this makes sense because resource economy is more relevant when they are scarcer.

The guidelines above are thought to be of general application and they demonstrate dramatic improvements that are enough to determine the viability of the project as a whole or at least help choose an adequate hardware processing platform that, in turn, will allow to optimize the overall cost of the project (example: reduce battery size). With the optimized algorithms and for the shown vision tasks, the second most inexpensive platform (Raspberry Pi 2) can be chosen, with associated energy savings, even more relevant for battery-based systems. The mentioned platform was successfully tested in autonomous driving competition of the Portuguese Robotics Open.

Regarding the case study, future work includes better profiling inside the perception, decision and control loop, even better separation of debug tools and measuring latency times. Another study of interest would be multi-core distribution of tasks on several platforms.

Supplementary Materials: This work was been tested in the 2017 Autonomous Driving Competition of the Portuguese Robotics Open. A video showing the results of the proposed tracking system can be found at: https://www.youtube.com/watch?v=6XN29PRc5Eg. A video showing the results of the proposed signalling panel recognition system can be found at: https://www.youtube.com/watch?v=KaPIKzncMd8. A representative video of the robot competing in the year of 2017 can be found at https://www.youtube.com/watch?v=dbCXKyT-d-w, and the corresponding simulator at https://github.com/ee09115/conde_simulator.

Acknowledgments: Authors gratefully acknowledge the funding of Project NORTE-01-0145-FEDER-000022—SciTech—Science and Technology for Competitive and Sustainable Industries, co-financed by Programa Operacional Regional do Norte (NORTE2020), through Fundo Europeu de Desenvolvimento Regional (FEDER).

Author Contributions: Valter Costa and Peter Cebola have designed and implemented the proposed vision system: Valter was mainly responsible on the lane tracking task, while Peter was more focused on the signalling panel recognition task. Armando Sousa and Ana Reis supervised the work by providing important insights such as the "zero copy one pass" methodology. Moreover, they have provided the means (robot, cameras, etc) by which this work was possible.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bharatharaj, J.; Huang, L.; Mohan, R.; Al-Jumaily, A.; Krägeloh, C. Robot-Assisted Therapy for Learning and Social Interaction of Children with Autism Spectrum Disorder. *Robotics* **2017**, *6*, 4.
2. Horst, M.; Möller, R. Visual Place Recognition for Autonomous Mobile Robots. *Robotics* **2017**, *6*, 9.
3. Kawatsu, C.; Li, J.; Chung, C.J. Robot Intelligence Technology and Applications 2012. In *Intelligence*; Springer: Cham, Switzerland; Heidelberg, Germany; New York, NY, USA; Dordrecht, The Netherlands; London, UK, 2012; Volume 345, pp. 623–630.
4. Audi Autonomous Driving Cup. Audi Autonomous Driving Cup. Available online: https://www.audi-autonomous-driving-cup.com/ (accessed on 15 July 2016).
5. PRO. XVI Portuguese Robotics Open. Available online: http://robotica2016.ipb.pt/indexen.html (accessed on 20 June 2016).
6. Hager, G.D.; Toyama, K. X Vision: A Portable Substrate for Real-Time Vision Applications. *Comput. Vision Image Underst.* **1998**, *69*, 23–37.
7. Kardkovács, Z.T.; Paróczi, Z.; Varga, E.; Siegler, Á.; Lucz, P. Real-time Traffic Sign Recognition System. In Proceedings of the 2nd International Conference on Cognitive Infocommunications (CogInfoCom) (CogInfoCom), Budapest, Hungary, 7–9 July 2011; pp. 1–5.
8. Wahyono; Kurnianggoro, L.; Hariyono, J.; Jo, K.H. Traffic sign recognition system for autonomous vehicle using cascade SVM classifier. In Proceedings of the 40th Annual Conference of the IEEE Industrial Electronics Society, (IECON 2014), Dallas, TX, USA, 30 October–1 November 2014; pp. 4081–4086.
9. Della Vedova, M.; Facchinetti, T.; Ferrara, A.; Martinelli, A. Visual Interaction for Real-Time Navigation of Autonomous Mobile Robots. In Proceedings of the 2009 International Conference on CyberWorlds, Bradford, UK, 7–11 September 2009; pp. 211–218.
10. Kyrki, V.; Kragic, D. Computer and Robot Vision [TC Spotlight]. *IEEE Rob. Autom Mag.* **2011**, *18*, 121–122.
11. Chen, N. A vision-guided autonomous vehicle: An alternative micromouse competition. *IEEE Trans. Educ.* **1997**, *40*, 253–258.
12. Stankovic, J.A. Misconceptions about real-time computing: A serious problem fornext-generation systems. *Computer* **1988**, *21*, 10–19.
13. Sousa, A.; Santiago, C.; Malheiros, P.; Costa, P.; Moreira, A.P. Using Barcodes for Robotic Landmarks. In Proceedings of the 14th Portuguese Conference on Artificial Intelligence, Aveiro, Portugal, 12–15 October 2009; pp. 300–311.
14. Costa, V.; Cebola, P.; Sousa, A.; Reis, A. Design Hints for Efficient Robotic Vision—Lessons Learned from a Robotic Platform. In *Lecture Notes in Computational Vision and Biomechanics*; Springer International Publishing: Cham, Switzerland, 2018; Volume 27, pp. 515–524.
15. Costa, V.; Rossetti, R.; Sousa, A. Simulator for Teaching Robotics, ROS and Autonomous Driving in a Competitive Mindset. *Int. J. Technol. Human Interact.* **2017**, *13*, 19–32.
16. Ros, G.; Ramos, S.; Granados, M.; Bakhtiary, A.; Vazquez, D.; Lopez, A.M. Vision-Based Offline-Online Perception Paradigm for Autonomous Driving. In Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 5–9 January 2015; pp. 231–238.
17. Aly, M. Real time Detection of Lane Markers in Urban Streets. In Proceedings of the 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, 4–6 Jun 2008; pp. 7–12.
18. Sotelo, M.A.; Rodriguez, F.J.; Magdalena, L.; Bergasa, L.M.; Boquete, L. A color vision-based lane tracking system for autonomous driving on unmarked roads. *Auton. Robots* **2004**, *16*, 95–116.

19. Son, J.; Yoo, H.; Kim, S.; Sohn, K. Real-time illumination invariant lane detection for lane departure warning system. *Expert Syst. Appl.* **2015**, *42*, 1816–1824.
20. Thorpe, C.E.; Crisman, J.D. SCARF: A Color Vision System that Tracks Roads and Intersections. *IEEE Trans. Rob. Autom.* **1993**, *9*, 49–58.
21. Muad, A.M.; Hussain, A.; Samad, S.A.; Mustaffa, M.M.; Majlis, B.Y. Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system. In Proceedings of the 2004 IEEE Region 10 Conference Analog and Digital Techniques in Electrical Engineering, Chiang Mai, Thailand, 21–24 November 2004; pp. 207–210.
22. Oliveira, M.; Santos, V.; Sappa, A.D. Multimodal inverse perspective mapping. *Inf. Fusion* **2015**, *24*, 108–121.
23. Ruta, A.; Li, Y.; Liu, X. Real-time traffic sign recognition from video by class-specific discriminative features. *Pattern Recognit.* **2010**, *43*, 416–430.
24. Dalal, N.; Triggs, B. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In Proceedings of the IEEE Intelligent Vehicles Symposium, Las Vegas, NV, USA, 6 June–8 June 2005; pp. 255–260.
25. Maldonado-Bascon, S.; Lafuente-Arroyo, S.; Gil-Jimenez, P.; Gomez-Moreno, H.; Lopez-Ferreras, F. Road-Sign Detection and Recognition Based on Support Vector Machines. *IEEE Trans. Intell. Transp. Syst.* **2007**, *8*, 264–278.
26. De la Escalera, A.; Moreno, L.E.; Salichs, M.A.; Armingol, J.M. Road traffic sign detection and classification. *IEEE Trans. Ind. Electron.* **1997**, *44*, 848–859.
27. Zaklouta, F.; Stanciulescu, B. Real-time traffic sign recognition using spatially weighted HOG trees. In Proceedings of the 15th International Conference on Advanced Robotics (ICAR), Tallinn, Estonia, 20–23 June 2011; pp. 61–66.
28. Gil-Jiménez, P.; Lafuente-Arroy, S.; Gómez-Moreno, H.; López-Ferreras, F.; Maldonado-Bascón, S. Traffic sign shape classification evaluation II: FFT applied to the signature of Blobs. In Proceedings of the IEEE Intelligent Vehicles Symposium, Las Vegas, NV, USA, 6–8 June 2005; pp. 607–612.
29. SPR. Festival Nacional de Robótica–Portuguese Robotics Open Rules for Autonomous Driving. Available online: http://robotica2016.ipb.pt/docs/Regras_Autonomous_Driving.pdf (accessed on 10 November 2016).
30. Ribeiro, P.; Ribeiro, F.; Lopes, G. Vision and Distance Integrated Sensor (Kinect) for an Autonomous Robot. *Rev. Robot.* **2011**, *85*, 8–14.