A Vision-based Solution for the Navigation of a Mobile Robot in a Road-like Environment

M. Oliveira, V. Santos, *Member, IEEE*

***Abstract* — This paper presents a set of algorithms and techniques based exclusively on artificial vision to provide robust and efficient real-time perception for a mobile robot to follow a road-like track, including several obstacles, in the context of a mobile robot competition concerned with Autonomous Driving. The paper shows how a set of basic image processing methods and elementary vision techniques can be arranged and combined to provide competitive results on the challenging task of autonomous navigation. The methods were applied on a robot that obtained the first place in the national competition of *Robot Autonomous Driving* in 2006.**

I. INTRODUCTION

THE Portuguese Robotics Open (*Festival Nacional de Robótica*) is a mobile robot competition introduced in 2001 where one of the challenges for the robots is to navigate autonomously on a road-like track [1][2][3]. The task complexity is increased by the presence of a zebra crossing area and a mid-road dashed line plus traffic lights to regulate and interfere in the navigation sequences. Further on, there is a tunnel which affects light conditions and, above all in matters of difficulty, the presence of unknown obstacles in the form of boxes covering about half of the road and a road maintenance area which completely reshapes the road using alternative delimiters which must be respected.

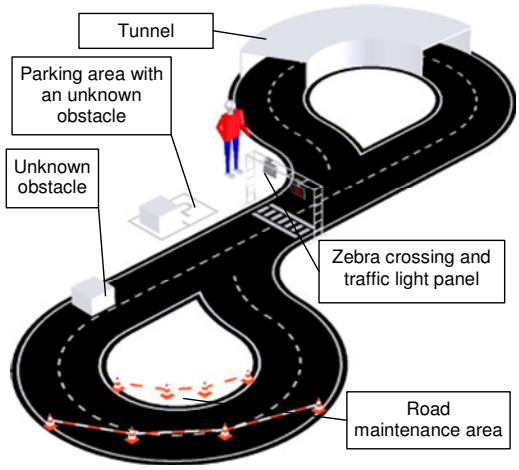

Figure 1 - Autonomous Driving competition area.

There are essentially three main paradigms to perform autonomous navigation along the road-like track: odometric, inertial or global referencing methods; line-following techniques; or, finally, following the track itself instead of lines.

Odometric and inertial methods require a model of the environment and is not easily extensible to general cases, besides the fact of cumulative unbounded errors due to proprioceptive sensors. Line-following techniques are quite common and still used nowadays [4] [5], and could do the job, but on this problem, in several parts of the track, the lines are absent or are in excessive number, forcing frequently the navigation to be done in open loop control, making the approach prone to failure and not efficient. For these reasons, the authors decided to use track-based navigation, in the idea pioneered by several authors, namely D. Pomerleau [6], but using simple on-the-fly processing tools for robustness, and very little previous knowledge of the environment.

Now that the navigation paradigm was decided, overcoming the entire set of challenges could be done with the assumption of one of two possible approaches: single frame and non-contextual processing, or a time dependent analysis including predictive filters and similar stochastic tools. This latter alternative based on statistical processing of data, based on models, either for the perception or the environment features, was at first not considered primordial because there was the expectation that a deterministic single-frame processing would be possible. This expectation was later verified and the alternative approach was then discarded.

Obviously, the algorithm has to be fast enough for real time navigation and at least the system must desirably be able to process 10 to 15 frames per second, since at 3 meters per second this would imply one frame being processed every 20 cm of traveled distance by the robot. As will be shown, this was perfectly achieved and the overall navigation algorithm, including vision processing, turned out to be quite robust practically without any fatal failures during the competitions.

II. HARDWARE AND COMPUTATIONAL SYSTEM

For the sake of easier comprehension of the overall navigation and processing system, a brief description of the entire set-up is now given.

The robot, named Atlas 4, is fairly 1 meter long per 65 cm wide, possesses traction wheels coupled with a mechanical differential gear, and uses an Ackerman steering system.

Reviewed Manuscript received April 13, 2007.

Authors are with the Department of Mechanical Engineering, University of Aveiro, Portugal. E-mails: {mriem, vsantos}@mec.ua.pt.

Two wide-angle cameras (89º field view each) plus a third one dedicated to traffic light processing fulfil the vision hardware, and are Firewire (IEEE1394) compliant. The low level control is achieved by a distributed system based on PIC microcontrollers which interface motors and other accessory peripherals, and there is a central system based on a laptop running Linux with the adequate drivers for Firewire image acquisition, and using the OpenCV open source library for image processing. Atlas 4 (Figure 2) is a revised version of Atlas 3, which has participated in the 2005 edition of the Portuguese Robotics Open.

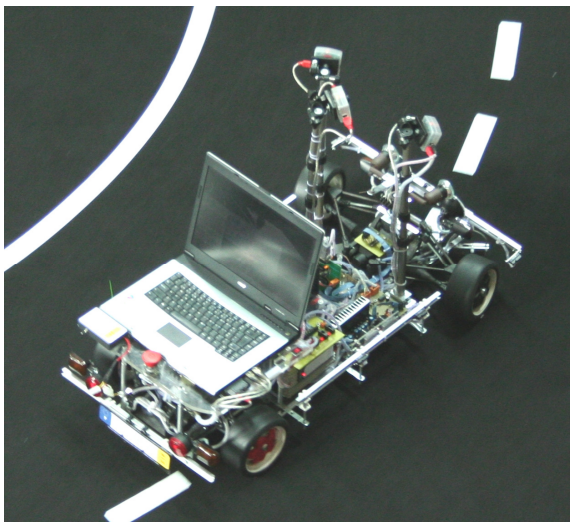

Figure 2 –The Atlas IV robot.

III. Image Acquisition and Merging

For road navigation, the robot uses two cameras with the purpose of obtaining a very wide angle image. Cameras are not tightly registered so, in order to merge the images from the cameras, image transformations have to be accounted for. Furthermore, since the cameras possess wide angle lenses a considerable amount of distortion occurs. At first glimpse, the full modeling of both the lenses parameters and the perspective transformations would be expected in order to obtain a perfect (geometrically accurate) merging of the information. However, the authors have come to the conclusion that these calibration procedures were fairly demanding and could be easily lost due to unstable camera physical fixations and other hardware issues. Based on this observation, a different approach was attempted where a rough image merging would be enough. In fact, image merging is required to be accurate only if precise geometrical features are to be extracted from the combined image. That is not the case on this road navigation problem. The rough image combination suits well if the subsequent road filters are only minimally affected by it. Figure 3 shows the original images obtained on each camera.

Figure 3 - Separate left and right images as given by the cameras.

Since only a rough combination of both images was required, a manual calibration of the distortion parameters is performed for each camera in order to combine them; this procedure is executed offline, therefore without any effect on the efficiency of the navigation algorithm. An interactive application was then developed to allow this manual calibration and its interface is shown in Figure 4.

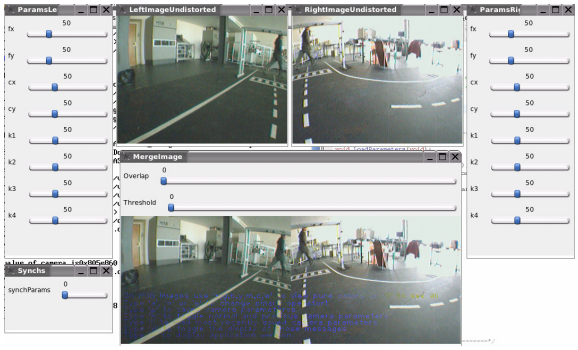

Figure 4 - Interface of the application for manual image calibration,

It should be noted that this method is entirely empiric, since for more rigorous combinations a more precise perspective transformation should also be taken into account.

IV. Filtering Road Images

A. *Basic Image Filters*

Several simple image processing techniques and filters are applied to each frame. All of them aim to obtain an image that is suited for the retrieval of navigation relevant information. Furthermore, with these filters, the navigation output is expected to be less prone to errors, hence improving the reliability of the whole process.

It is the user's initial responsibility to manually achieve the best possible match between both images, including tuning the image overlapping in the horizontal direction. The overlapping corresponds to the amount of columns that are merged (a simple mean) on both images.

Figure 5 - RGB threshold of merged images in the offline user interface.

The application used to manually configure the image

merging also allows adjusting a threshold value for the processing that will follow (Figure 5). A fixed threshold was found to be enough as opposed to dynamic threshold.

1) Image Threshold

For faster processing, and considering that no relevant information is lost, a single channel of the RGB input image is selected, and any of the three channels yields similar results for this kind of images. Alternatively, the RGB image can be converted to grayscale, but either way, the resulting image should have only one channel for the remainder processing. As mentioned, this operation's input parameter, the threshold limit, is usually tuned offline so that none of the road border lines looses connectivity, even if this implies some spurious white spots remaining present in the image.

This "poor tuning" is not critical since subsequent processes will be able to filter the spurious spots requiring only that the road border line connectivity is not lost. In Figure 7 (binary version of Figure 6) many spurious white spots, both inside and outside the road, were not eliminated but the road line connectivity remains solid.

Figure 6 - A view of the road in RGB.

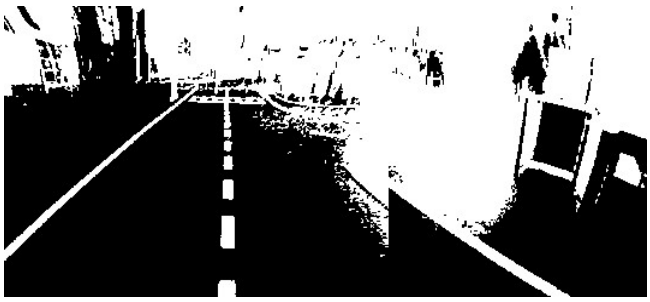

Figure 7 - Binary version of image in Figure 6

2) Filter for isolated points

A filter for the removal of isolated pixels or very small spots is also applied to the binary image. This kind of operation is quite common in image processing and a simple 3x3 kernel like the following can be used.

$$K_{ipt} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (1)$$

The result of the convolution of the image with K_{ipt} is a mask that is used to erase the pixels whose result of convolution was equal to 8.

B. Advanced Road Filter

The Advanced Road Filter is not a filter in the usual sense but rather a predefined sequence of simple image operations, most of them of Boolean nature and hence computationally efficient. This methodology is the core of the whole navigation process but, to be applied, some conditions or characteristics in the input image must be met or imposed.

1) Border line connectivity

The most important condition is that the road border lines must always be connected. If the condition is not met for one of the lines, the process will rely on the other line, although with a lower robustness. If by any chance, in one occasional frame, none of the lines is connected, the process takes no decisions and uses the previous frame's decision, but in the unlikely possibility of this to happen it will certainly be temporary.

2) Virtual horizon selection

A virtual horizontal line, henceforth simply called "horizon", is drawn in the source image at a height so that at least one of the road border lines always touches it. The horizon line index placement can be performed at a fixed height coordinate or computed dynamically. This last option was attempted in several ways although the results were no better when compared to its simpler counterpart. The tools developed to dynamically select the horizon's coordinate were mostly based on an iterative process where a line would be drawn, the fill operations executed, and the results compared to some type of evaluation criteria. However, given that the fill operations are quite computationally demanding in the whole navigation process, performing them iteratively is of no major advantage.

3) Inner seed point

A predetermined point called inner seed point is placed always inside the road in a position that corresponds to a black pixel. The first premise is actually not hard to ensure since that a high height value combined with a column value close to half of the image width usually satisfies this condition. In what regards the black pixel condition, a simple search routine around the preferred inner seed point coordinates solves the problem. The importance of this condition is due to the fact that the flood fill operations will start here and so it is important to be sure that the inner seed point connects most of the blacked area inside the road. Further tests can be done to ensure this. Two or more seed points can be used in a parallel process that unifies once the flood fill is completed. The unification is done with a mere Boolean AND between the flood fill results.

4) Sequence of operations

Once all of the above conditions are met, the actual process can now be applied.

Figure 8 - The original merged image and its binary version.

The image is made binary (Figure 8) using a threshold value obeying the criteria defined in section IV.A.1).

Then, the horizon line is artificially inserted in the binary image (Figure 9). This covers the top of the road ensuring that there is a delimited area in the image that actually corresponds to the road. All the pixels above the image horizon are erased (set to black). The erased information will have no influence on the navigation. However, erasing the pixels will help some subsequent operations.

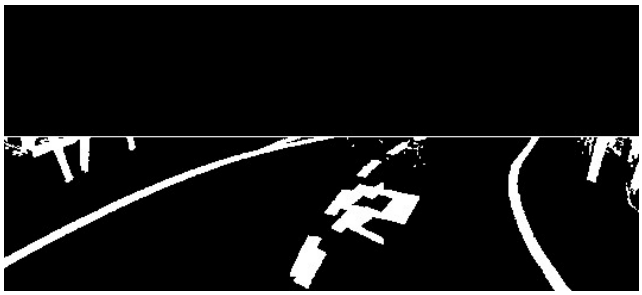

Figure 9 - Placement of Horizon.

Now, the first fill can be performed (Figure 10). The image is filled using the inner seed point as the start of the process. It is important to obtain the filled area as the next work image.

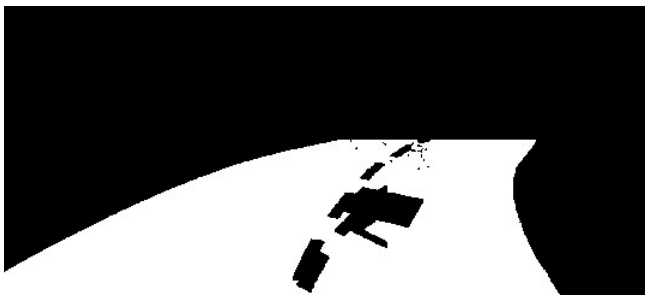

Figure 10 - Inner road fill.

A new horizon line is drawn on the new image (Figure 11). This second horizon line should be inserted in the same place where the first was. Its purpose is to cover the top of the previous image, but still allow the subsequent fill operation to propagate to the lower part of the image (under the horizon). In order to do this, the line's length is two pixels shorter than the image's width. The line is centered on the image and so two "connection channels" remain at columns with coordinate 0 and image width.

With a new horizon set, the last fill operation is executed. Any point above the horizon can be used as seed point since the area was previously erased. If one uses the negative of the filled area image, its white pixels correspond exactly to the road bellow the horizon.

Figure 11 - Outer road fill.

The advantage of the process is that with the resulting information one can think in terms of forbidden/allowed space. The resultant image holds important information for the navigation process since it describes which pixels are inside the road and which are not. Furthermore, the pixels on the edge of the road may easily report the curvature of the road. The area of the road can also be easily calculated, if needed.

V. Road Information Compression

The previously presented road filter obtains information about the road's features that are more trustworthy than the ones in the original image. There are several ways of using this information. In the present case, and considering some particular road and robot characteristics such as the maximum useful distance to perceive (arbitrarily set at about 2 meters) and the maximum road curvature (which is known before hand and never very hard to comply to), it was decided to compress the information from one image with N pixels to just 4 point coordinates. N can be given by.

$$N = \underbrace{\left[Limg_{width} + Rimg_{width} - OV \right]}_{Merge\,\mathrm{Im}age_{Width}} \times \underbrace{\left[Limg_{height} - H \right]}_{Merge\,\mathrm{Im}age_{Height}} \quad (2)$$

Where *Limg* represents the left side image, *Rimg* the right side, *OV* is the preset overlap between original images and *H* the defined horizon column value. These points, two on each side, approximate the road to a trapezium that we call "box".

A. Road Area Search – Box Analysis

To build the box, four points have to be found. The points are called top right (*TR*), top left (*TL*), low right (*LR*) and

low left (*LL*). The search algorithm is very simple. Starting from preset coordinates for each point, a white pixel is searched from the outside to the inside of the road and to the bottom or to the top of the image for top and low points respectively. When found, its line and column coordinates are stored in the corresponding points and the search is finished.

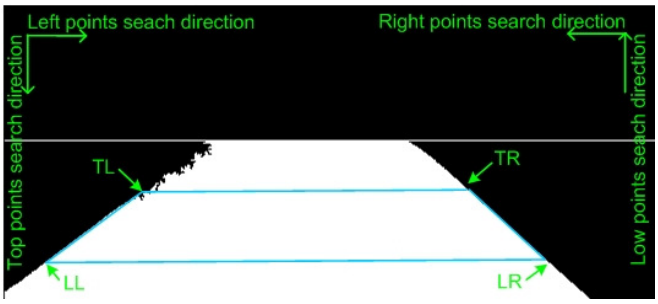

Figure 12 - Box Points Search.

Notice that both $\overline{LL,TL}$ and $\overline{LR,TR}$ slopes define very well the road limits, without relevant loss of information. Sometimes, however, some of these points cannot be found and in that case the point is invalidated.

B. Calculating the GoTo Points - GPT

All navigation decisions are based or deduced from the box analysis information. The next step is now to calculate points to where the robot should head. We call these points GoTo points (GTP). Several GTPs are created based on several different criteria. Also, different GTPs need the validity of distinct box points in order to be calculated. An overview of the procedure to calculate GTPs is shown in Figure 13.

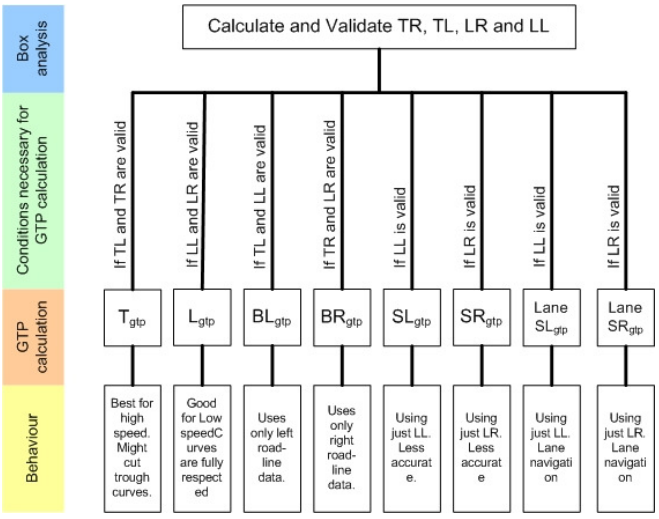

Figure 13 - GoTo points calculation procedure.

The first GTP is the Top GoTo point (T_{gtp}). To be calculated, it requires the validity of both top points in the box analysis. It is obtained by finding the coordinates in the middle of the line that unites *TL* and *TR*.

$$\overrightarrow{T_{gtp}} = \frac{\overrightarrow{TR} + \overrightarrow{TL}}{2} \tag{3}$$

The T_{gtp} point, when defined, is the preferred one. It ensures smooth navigation by aiming the steer towards the end of the vision field. By using *LR* and *LL* box points, a similar low GTP (L_{gtp}) can be calculated.

Sometimes, however, these pairs of box points are not valid. Due to this, some other GTPs are calculated. These are less accurate than the ones mentioned before, but on the other hand they don't need as much information. It is the case of both left GoTo points BL_{gtp} in which only *TL* and *LL* are used.

$$\overrightarrow{BL_{gtp}} = \begin{bmatrix} LL_x + \alpha \times \dfrac{RW(y)}{2} \\ LL_y \end{bmatrix} \tag{4}$$

where *RW(y)* is the typical road width as a function of the selected line. It is found by using a lookup table that was built during calibration and α is the difference between the typical left road line angle obtained during calibration and the angle formed by $\overline{TL, LL}$ and the horizontal line.

$$\alpha = \beta - \tan\left(\frac{TL_y - LL_y}{TL_x - LL_x}\right) \tag{5}$$

where β is the typical left road line angle, also obtained during calibration procedures. Of course, the same calculation can be made to find BR_{gtp} on the other side of the road.

In the worst case scenario, only *LL* box point is valid (or *LR*, but the calculation is the same). The only information available is the point's coordinates. Therefore, a similar calculation is made to find the single left GoTo point SL_{gtp}, but no angle can be used in the process.

$$\overrightarrow{SL_{gtp}} = \begin{bmatrix} LL_x + \dfrac{RW(y)}{2} \\ LL_y \end{bmatrix} \tag{6}$$

In all of the previously mentioned points, the goal is, despite having different approaches, to try to head for the center of the road. Sometimes this might not be the desired behavior. It is interesting to try to drive in the middle of the right or left lane. Following this reasoning, a new set of points can be created where the image road width is not divided by 2 but by 4. With this approach, a new set of points is defined in order to attempt to drive on each one of the lanes. In the application at hand, following one of the lanes is a behavior rarely used. Thus, only the simplest GTPs are calculated for this mode, the lane single left and right GoTo points.

$$\overrightarrow{LaneSL_{gtp}} = \begin{bmatrix} LL_x + \dfrac{RW(y)}{4} \\ LL_y \end{bmatrix} \tag{7}$$

All these points are mere possibilities of where the direction should steer to. They are all calculated whenever

possible. Hence, there is a parallelization of the process of navigation at this point that will be united further ahead.

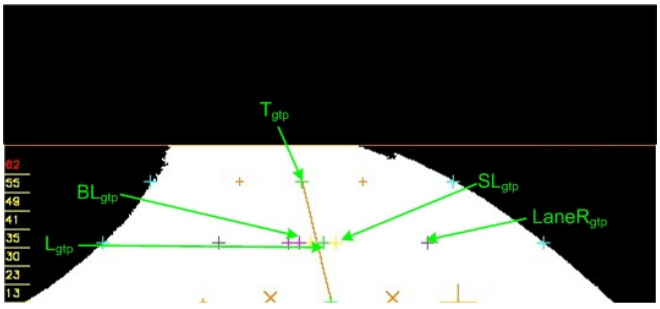

Figure 14 - GoTo points calculation.

C. GTP validation tests

The calculation of the GTPs as exposed in section V.B, especially regarding the T_{gtp} and L_{gtp}, might sometimes be inadequate. For this reason some validation tests are performed to check the validity of the points.

1) Line stepping test

Because the T_{gtp}'s height value is low (it's close to the top of the image), this implies that heading towards this point might sometimes cut trough curves or obstacles that are closer (because the steer planning aims very far). Figure 15 shows a view of the road with an obstacle very close to the robot. The obstacle is white and so, is handled by the advanced road filter as a mere increase in the thickness of the line, therefore being considered as an area outside of the road (Figure 16).

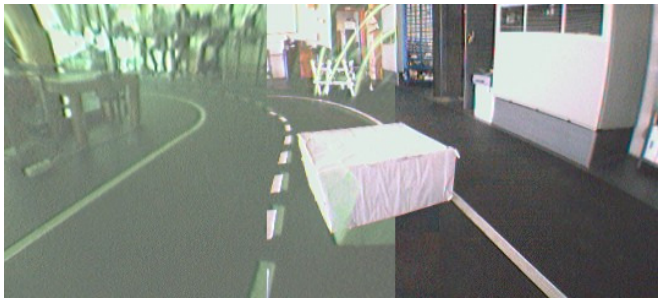

Figure 15 - Obstacle on the road.

Since the obstacle is very close to the robot, heading towards the end of the road might disregard the object's presence and possibly cause the robot to bump into the obstacle. This situation must be detected and the T_{gtp} invalidated. Once again, a less accurate geometrical criterion suits adequately. Since both cameras are at the same distance to the robot's longitudinal axis, it can be assumed that its position in the image is at the bottom (line) and middle column of the image. This point is called present point (*PP*). Two points are obtained by shifting the column coordinate of *PP* left and right. The amount to shift can be obtained as a relation between the calibrated road width *RW*, and the robot width to road width ratio, this last is also measured experimentally:

$$Shift = PP_x \times RW\left(PP_y\right) \times r \tag{8}$$

Where r represents the robot width to road width ratio. These points are referred to as low line stepping points *LLST*. These roughly represent the robot's width as if it was seeing himself positioned at height equal to *PP*'s height. The same is done using the T_{gtp} as shift origin, obtaining the top line stepping points *TLST*.

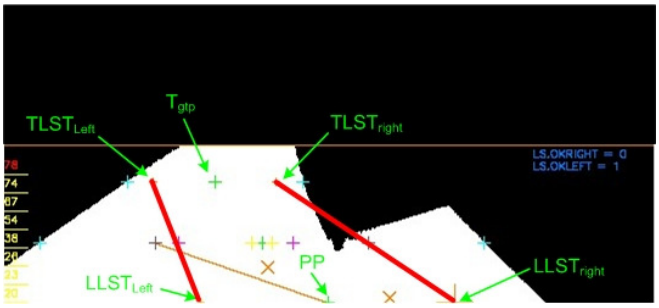

Figure 16 - Line stepping test.

The criterion used to validate T_{gtp} is simple. Lines $\overline{LLST_{left}TLTS_{left}}$ and $\overline{LLST_{right}TLTS_{right}}$ (thick lines in Figure 16) must not cross through any black region, since that, because of the advanced road filter, the whole area inside the road is painted in white. If this condition is not met, T_{gtp} is invalidated. Figure 16 is generated automatically from the navigation program; on the right side, just below the horizon, one can see the result of the application of this test to the left and the right line. The test on the left is 1, which means that no line stepping occurs on the left. The result for the right side, on the other hand, indicates that line stepping will exist and therefore T_{gpt} was discarded.

2) Anomalous road width

Another test checks the variation of the road width at line coordinates T_{gtpx} and L_{gtpx}, and compares it to the typical road width for those coordinates.

$$arw = \frac{TR_x - TL_x}{RW(T_{gtpx})} \tag{9}$$

The value of *arw* is equal to 1 if the widths are equal and is less than 1 when the road width is smaller than typical, and vice versa. The criterion is that *arw* must lay between some preset variation of the typical road width.

Figure 17 - View of a junction.

Figure 17 shows a view of a junction. In this case, the road width considerably exceeds the typical value. Hence, it is not

reliable. Accordingly, points that are based on the image road width (and not on the typical road width), i.e. that are based on the difference from the left and right box points column coordinates, must be discarded.

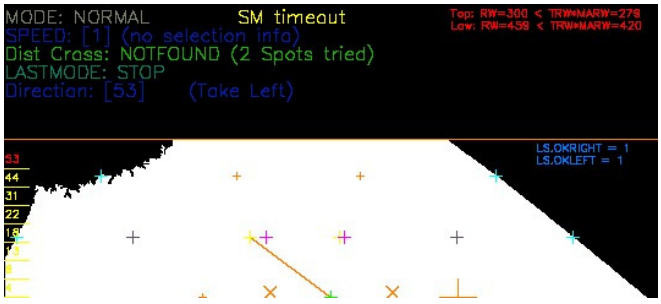

Figure 18 - Junction handling.

When the same happens for a confined road width, those points are also invalidated and the navigation algorithm detects it and also communicates it to the user for debug purposes (Figure 18).

D. Calculation of the Drive Angle for each GTP

For each calculated and validated GTP, an angle of heading is found. This angle's calculation is actually very simple since it is merely translated by the angle between line that unites *PP* and the GTP in question and the vertical axis.

$$DA_{GTPi} = \frac{\pi}{2} - \tan\left(\frac{GTPi_y - PP_y}{GTPi_x - PP_x}\right) \qquad (10)$$

E. Decision making process

As was mentioned in section V.B, the navigation process finds several GTPs and all of them are plausible possibilities depending on the context or even the desired navigation behavior. "Context" means the multitude of state flags that are computed for each road image. For example, in normal conditions, SL_{gtp} has priority over $LaneSL_{gtp}$. This can be easily understood by the assertion that a behavior that tries to center the robot in the road is preferred to the one that attempts a middle lane placement. However, when an obstacle is present on the road, and a confined road width flag is raised, as in Figure 15 and Figure 16, the opposite might be true. In other words, if an obstacle is blocking the right lane, then, for safety reasons, the $LaneSL_{gtp}$ has priority until the obstacle is overcome (or until a predefined amount of time elapses). The decision making process handles all the possibilities and elects the optimum behavior (GTP). It's a large function of nested if-else decision making, but the advantage is that, for achieving a complete different navigational behavior, one has to change only this part of the code.

Another important behavioral decision is whether to go right or left when a road junction appears, i.e., an excessive road width is detected (Figure 17 and Figure 18). The decision is dependent on the priority of SL_{gtp} in relation to SR_{gtp}. If the first has precedence over the second, the robot will go left. This is usually decided by upper levels of the navigation algorithm.

VI. Pattern Recognition

A. Detection of the Zebra Crossing Area

Because of the advanced road filter, it is very easy to extract the pixels that are inside the road. This can be used with different functionalities. In our case, the zebra crossing areas (henceforth named simply "cross" or "cross area") needs to be detected by the vision system.

Figure 19 - A view of the cross.

Figure 19 illustrates a crossing area, which, due to the rough image merging, possesses a highly adulterated geometry.

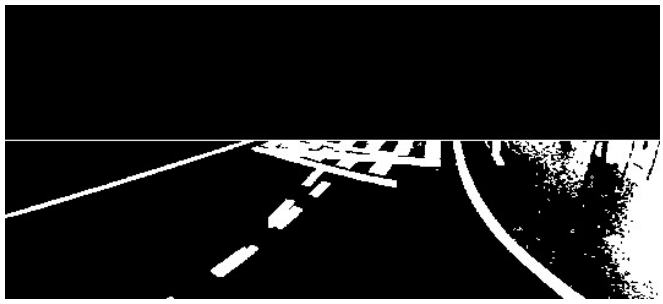

Figure 20 First fill for the cross view image.

Figure 20 shows the threshold and horizon placement stage of the advanced road filter view of the cross, while Figure 21 represents the final output.

Figure 21 - Second fill for the cross view image.

The inner road spots can be obtained by performing a logical AND operation between the images in Figure 20 and Figure 21. The result can be seen in Figure 22.

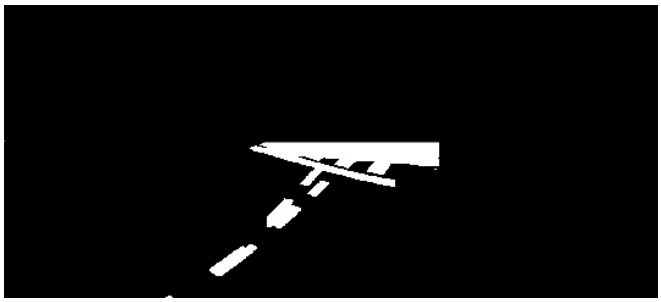

Figure 22 - Inner road spots including the cross area.

The resulting image possesses various spots which are separated based on their connectivity and are tested separately, for they are all cross feature candidates. For the actual test on each spot, the minimum area rectangle is calculated and its height to width ratio is compared to that of the actual cross (measured experimentally). Close values raise the flag that indicates the presence of the cross. This procedure, especially the segmentation part, is still computationally demanding, and therefore the road image is sliced horizontally (represented in the lower left part of Figure 23). The amount of white pixels in each slice is evaluated, and the algorithm only tests the spots that appear on the slice that has the biggest amount of white pixels. Some time can be saved by using this guessing method prior to the segmentation of the spots.

Figure 23 - Cross detection.

B. *Traffic Lights Analysis*

In the competition, the robot has to stop at the zebra crossing and recognize the signs displayed in the traffic lights panel. For this purpose, a third camera was installed on the robot.

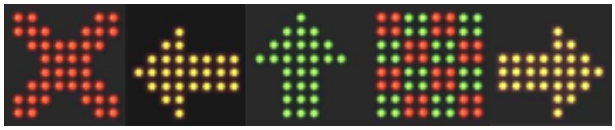

Figure 24 - Possible signs displayed by the traffic lights.

Figure 24 shows the possible signs that mean, stop, go left at junction, go forward at junction, last lap and park, respectively from left to right. Up until the 2005 competition, the traffic light was a custom-built equipment that displayed the signs trough the control of an array of LED's (Figure 25).

Figure 25 - Traffic lights in 2005(left) and algorithm output (right).

The recognition was achieved using a conversion to the HSV color space followed by the use of separate filters on each channel. This technique allowed color recognition. Complementarily, some very simple procedure of shape analysis was performed to detect the sign orientation [7].

This technique lost reliability for the 2006 edition, when the traffic lights were then displayed on a flat screen monitor. The monitor's brand and model was not available and so the variation of brightness, contrast, etc, were impossible to predict. Additionally, due to the variation in lighting conditions, a new method was attempted. Highly efficient, Open Source computer Vision (Opencv) and Intel Performance Primitives (IPP) based, template matching is then performed on the image. No scaling or rotation is taken into account, although most of the times the results are quite satisfactory. Real time template matching (10Hz to 15Hz doing template matching plus navigation) is achieved.

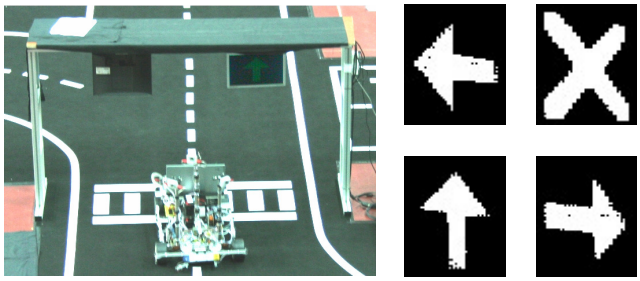

Figure 26 - Traffic lights in 2006 (left) and used templates (right).

Due to the fact that the zebra cross is detected before reaching it, template matching can be executed before the robot stops at the zebra. This enabled to use traffic lights recognition at distance and in motion; therefore, if the sign was not ordering a stop, the robot would not stop to perform recognition, it would simply keep up its speed and dash trough the zebra without slowing down.

C. *Open loop backwards maneuvering*

The robot uses Ackerman system and its size compared to the road is considerable and also its maximum steering is limited. Obviously, it is not a holonomic robot. Because of this, some of the elected GTPs steer angle that results from the navigation algorithm exceeds the physical limits of the car. When a steer angle above the maximum possible is requested, an open loop backwards maneuvering is performed. A special routine is launched that steers to a preset angle on the opposite quadrant of the one requested by the navigation. The robot's speed is then set to reverse for a preset duration, after which a new frame is processed. If the

new steer angle, got from the new image frame, is feasible, the general navigation is resumed, otherwise the reverse procedure continues. The outcome of this is that, whenever the robot finds out that it has to steer more than physically allowed, it drives backwards with opposite steer to gain clearance to move forward. Usually, after this maneuver the new steer angle is not as aggressive as earlier and the robot can overcome obstacles.

This routine has worked for various situations, from obstacle avoidance to road realignment proving itself to be a very reliable way to reorient the robot when it found itself in a situation where is maximal curving ability would not solve the problem.

VII. Results

The presented techniques have performed quite satisfactorily and led the robot to be the best performing in all challenges, and won the first prize in the 2006 edition. Very little prior knowledge of the road is given (only the expected width and maximal curvature), no model of the road and no odometry is used; all this reinforces the performance of the used techniques. The entire set of the described algorithms is processed for every road frame, at a rate of about 15 Hz on a common laptop running Linux.

The robot navigates quite fast and its speed is sometimes limited by the hardware, i.e. maximum steering and speed, instead of by software restrictions. It is also possible to use the same procedure for all road situations: normal navigation, obstacle contouring, confined road width, junction presence, zebra cross insight, and others. This means that no exception procedures are programmed for these special cases. Several navigation behaviors are also easily achieved just by changing the decision making process priorities without touching the core of the navigation code.

Extra redundancy to light conditions change and disturbance in the road's normal conditions is guaranteed by the advanced road filter. A great data reduction is performed through the box analysis, from a full image to only 4 points, without a significant loss on the road correct interpretation, and thus dropping the time demanded to process each frame.

Zebra cross detection is usually very accurate. The same can be said for traffic lights detection trough template matching, which have worked inclusively at the distance.

The philosophy of the whole algorithm is simple, which avoids complex calibration procedures. Exceptional cases such as the tunnel traversing or the road maintenance area have been solved within the global approach. In detail we can say the following: tunnel borders are white, therefore they simply are thicker road delimiters, and can be dealt with the very same algorithm. The maintenance area is delimited by orange and white stripes; segmenting the orange parts allowed the definition of new road delimiters and the same algorithm was then performed.

Open loop backwards maneuvering proved to be surprisingly effective, especially if one takes into account its simplicity. Finally, parking on the parking area was done in a short term open loop, except for the case when an obstacle was placed in one of the two possible places. There, simple vision techniques of area pixel counting in the image were used and worked effectively throughout the contest.

VIII. Perspectives and Final Remarks

The results obtained in the competition may suggest that the methods described could be attempted for the navigation in a real road. Naturally, several new issues would have to be attended, but the main idea might remain. One issue deals with the road border line connectivity that can be a problem in some real roads. However, future work could exploit techniques to fuse or join these fragmented lines in order to reconstruct the road boundaries. Another idea to explore is to try different camera positions, or to use a pan and tilt servo controlled unit that would provide the capability for active perception for, among others, cope with dynamic obstacles moving in the same space.

The bottom line of this work is that the methods used do not depend on metric distances, since only pixel counting is used. In consequence, the overall process is much simpler, does not depend on complex calibration procedures, and discards the need of accurate yet expensive cameras. The simplicity of the process also eases real time processing, and no odometry or time dependant methods are implemented. These facts lead the authors to think that the method may work on other scenarios of autonomous road following.

IX. References

[1] L. Almeida, J. Azevedo, C. Cardeira, P. Fonseca, P. Lima, F. Ribeiro, V. Santos, Festival Nacional de Robótica – ROBOTICA2001, *Robótica,* nº 45, 2º Trim. 2001, pp. 60-64 (ISSN: 0874-9019)

[2] Almeida, L., Azevedo, J. , Cardeira, C., Costa, P., Fonseca, P. Lima, P., Ribeiro, F., Santos, V., 2000. Mobile Robot Competitions: Fostering Advances in Research, Development and Education in Robotics. *Proc. of the 4th Portuguese Conference on Automatic Control, CONTROLO2000*, 4-6 October 2000, Guimarães, Portugal, pp. 592-597

[3] P. Afonso, J. Azevedo, C. Cardeira, B. Cunha, P. Lima, V. Santos, 2006. Challenges and Solutions in an Autonomous Driving Mobile Robot Competition, *Proc. of the 7th Portuguese Conference on Automatic Control, CONTROLO2006*, Lisboa.

[4] G. Beccari, S. Caselli, F. Zanichelli, A. Calafiore, 1997, Vision-based Line Tracking and Navigation in a Structured Environment, *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '97).*

[5] Hai-Bo Zhang, Kui Yuan, Shu-Qi Mei, Quing-Rui Zhou, 2004, Visual Navigation of an Automated Guided Vehicle Based on Path Recognition, *Proceedings of the Third International Conference on Machine Learning and cybernetics, Shangai, 26-29 August 2004.*

[6] D. A. Pomerleau, 1993. Neural Network Perception for Mobile Robot Guidance, *Kluwer Academic Publishers*.

[7] R. Cancela, M. Neta, M. Oliveira, V. Santos, 2005. ATLAS III: Um Robô com Visão Orientado para Provas em Condução Autónoma, *Robótica*, nº 62, pp. 4-11, (ISSN: 0874-9019).