Faculdade de Engenharia da Universidade do Porto

FEUPCAR 2.0: Condução Autónoma no Festival Nacional de Robótica

André Almeida Vidal

VERSÃO FINAL

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Professor Doutor Luís Paulo Gonçalves dos Reis
Co-orientador: Professor Doutor António Paulo Gomes Mendes Moreira

Janeiro de 2011

Resumo

Condução autónoma é a designação dada a um modo de controlo e navegação que possibilita um veículo deslocar-se sem qualquer intervenção humana de um ponto para outro no espaço. A visão por computador e fusão sensorial são, neste contexto, largamente utilizadas, pois são as ferramentas que até aos dias de hoje permitem mais fidedignamente recriar o complexo processo de percepção do ambiente envolvente pelo ser humano. Na análise de imagens convém distinguir quais as características de interesse a extrair, da grande quantidade de informação ambígua.

Encontros científicos internacionais na área de robótica começam a destacar a questão da condução autónoma, promovendo competições que visam incentivar o desenvolvimento destes sistemas. Em Portugal, surge igualmente uma competição que inclui a prova de condução autónoma. Esta prova é realizada numa pista em ambiente fechado e pretende recriar um cenário de uma estrada real. Neste trabalho pretendeu-se estudar a viabilidade da adaptação de um veículo telecomandado, para fins de condução autónoma.

Esta dissertação centrou-se na percepção do mundo através de visão, descrevendo metodologias para a extracção de pontos de referência essenciais, tais como a determinação dos limites da estrada, detecção de passadeira e determinação de sinais informativos. Esta informação é seguidamente interpretada e fundida com um sistema de sensores baixo-custo externos, permitindo obter a localização e posterior planeamento de trajectória. Neste documento são detalhadas todas as abordagens seguidas na análise e processamento sensorial, bem como o planeamento de acções a tomar.

Nas condições de teste, a detecção da passadeira é 100% eficaz até uma distância de cerca de 90cm, e a determinação dos sinais informativos até 120cm com distorções da forma acentuadas. A detecção dos limites da estrada é também bastante robusta, com valores de eficácia muito próximos de 100% em vários cenários. Os resultados obtidos a nível de navegação revelam uma solução modelar, adaptável e eficiente que poderá equipar qualquer veículo recreativo, escalável para veículos e ambientes maiores, providenciando uma condução autónoma eficiente.

Abstract

Autonomous driving is the designation given to a control and navigation mode that enables a vehicle to move without any human intervention from one point in space to another. Computer vision and sensor fusion are, in this context, widely used, as these are the tools that allow a more faithfully recreation of the complex process of perceiving the surrounding environment by human being. In image analysis one should distinguish the interesting characteristics to extract from the large amount of ambiguous information. This turns out to be quite complex with considerable computational costs, especially when it is required a real-time analysis.

International scientific meetings in robotics are beginning to highlight the issue of autonomous driving, promoting competitions designed to encourage the development of these systems. In Portugal, there is also an event that includes a competition of autonomous driving. It is held indoors on a track that aims to recreate real road scenery. This work aimed to study the feasibility of adapting a remotely operated vehicle for the purpose of autonomous driving.

This dissertation focused on the world perception through vision, describing methods for reference features extraction, such as determining the road limits, the crosswalk detection and then determination of informational signs. This information is then interpreted and merged with a set of low-cost external sensors, allowing proper localization and subsequent trajectory planning. This document details all the approaches followed in the analysis and sensory processing, as well as the planning of actions to take.

In test conditions, the detection of the crosswalk is 100% effective, up to a distance of about 90cm, and the determination of informational signs is efficient up to 120cm with considerable form distortions. The road limits proved to be very robust as well, with efficacy values close to 100% in various scenarios. The navigation results obtained reveal an adaptable and efficient system that can equip any recreational vehicle, scalable for larger vehicles and environments, providing an efficient autonomous driving.

Agradecimentos

Gostaria de começar por agradecer à minha família mais próxima, em especial à minha mãe, pela ajuda e apoio dado ao longo do projecto, e à minha irmã pela ajuda na revisão do documento.

Aos meus orientadores, Professor Luís Paulo Reis e Professor António Paulo Moreira, pela oportunidade, apoio e interesse ao longo da dissertação.

À minha namorada pelo apoio e compreensão da minha ausência durante o projecto. E a todos os colegas que de alguma forma ajudaram com ideias, sugestões, e melhoramentos, em especial ao meu colega e amigo Paulo Martins, que me acompanhou ao longo de todo o projecto.

Contents

List of Figures

List of Tables

Shortenings and Symbols

List of Shortenings (sorted alphabetically)

ADC	*Analog to digital Converter*
FEUP	*Faculty of Engineering of the University of Porto*
FNR	*"Festival Nacional de Robótica"*
GUI	*Graphic User Interface*
I/O	*Input / Output*
IR	*Infrared*
LED	*Light Emitting Diode*
LIACC	*Artificial Intelligence and computer science laboratory*
PWM	*Pulse Width Modulation*
Rpm	*Revolutions per minute (RPM, r/min, or r·min−1)*
RS-232	*Recommended Standard 232*
RTLS	*Real-Time Location Systems*
SPR	*"Sociedade Portuguesa de Robótica"*
UDP	*User Datagram Protocol*
UGV	*Unmanned ground vehicle*
USB	*Universal Serial Bus*
UWB	*Ultra Wideband*
WLAN	*Wireless Local Area Network*

Chapter 1

Introduction

1.1 - Motivation

Autonomous driving is the ability to imitate the complex task carried out by man while operating a vehicle, collecting sensory data and taking further action based on its analysis. It is an area of growing interest, especially in the world of robotics and automobile industry. In general, the aim of an autonomous driving system is to provide a vehicle with the ability to navigate from one point to another without the intervention of an operator, ensuring maximum structural safety to the vehicle, as well as its occupants and other vehicles, while complying with the road laws. Two main approaches are followed in this context: Fully autonomous vehicles or driving support systems operated by a human being (such as cruise control or electronic stability control). Essentially these systems aim to reduce (or cancel completely) the human error in driving, which is statistically proven to be the cause of more than 50% of millions of accidents that occur every year worldwide, claiming the lives of many. Furthermore, the energy efficiency is increased by using an acceleration profile more suitable for this purpose, consequently reducing air pollution as well.

This area of interest is increasingly being developed at both university and industrial levels and its attentiveness is growing. Prominent scientific events started to emerge worldwide, particularly in the early 20th century. Following this global trend, an event called FNR (*"Festival Nacional de Robótica"*) organized by SPR (*"Sociedade Portuguesa de Robótica)*, arises in Portugal. Originating in 2001, this event aims to promote science and technology through robotic competitions. Since its opening, this event has had a remarkable growth regarding both in teams and participants as well as audience. One of several competitions performed at the festival, is the autonomous driving competition, which has existed since its origin. This trial represents a challenge in which an autonomous mobile robot must travel a route along a closed track, which has strong similarities with a conventional road. The track consists in two lanes separated by a dashed line forming an eight-shaped profile, being its starting point defined by a crosswalk. At this landmark, there is an upper display which displays five possible signal drawings, indicating the vehicle which task to perform. Throughout the track, different physical traffics signs are placed, which should be detected, and its recognition should be reflected through different colored LEDs. There is also a parking zone, which should be used to park the vehicle, when ordered to do so. In

addition, the competition encompasses randomly placed features such as a working zone, a tunnel, obstacles, and continuous central line overlaid on the dashed central line.

1.2 - Background work

This work comes in the furtherance of an initial work already undertaken at FEUP, entitled "FEUPCAR: Autonomous Driving at the National Festival of Robotics" a Master Thesis by Pedro Gomes in June 2010 [1]. The final product of this work is an adaptation of a commercial recreative vehicle (a RC car) to a suitable vehicle both at a sensory as actuators levels to circulate in a track which pretends to recreate a normal road. To this end, the actuators electronics were developed, as well as the acquisition and processing electronics for the sensors data. These peripherals are connected to a board, which through a microprocessor, implements the low-level control algorithms as well as a communication protocol that communicates with an upper processing layer - a PC. In this upper layer, the high level control of actuators and sensors is performed, thus freeing resources from the microprocessor that performs higher time-constraining tasks.

1.3 - Objectives

Broadly, the main objective of this dissertation is to evaluate the feasibility to perform autonomous driving of an adapted radio-controlled vehicle, based on low-cost components and guided through computer vision. To this end, the final solution should be modular and scalable, easily adaptable to any regular vehicle.

Due to the characteristics of the autonomous driving competition held at the FNR, such competition was taken as the validation platform of the system. This way, its rules and features must be followed strictly and accordingly. However, the final solution should allow the vehicle to drive in any track, provided that it has all the common features of a normal road, with minimal adjustments required for the competition.

Special emphasis is given to navigation, since this is clearly the most complex and critical task of the system. Thus, in order to summarize, the sub-objectives for this thesis are defined as follows:

- Adapt / improve the vehicle already built, to support this system;
- Develop and implement vision-based algorithms to detect the track limits;
- Develop and implement vision-based algorithms to recognize the ordering signals, as well as the crosswalk landmark;
- Provide the vehicle with a system that allows autonomous and real-time navigation within the track limits;
- Follow the command orders given by the upper signaling panel;
- Integrate the previous topics in a decision layer, with well-defined data interface between the already developed control layer.

1.4 - Thesis Structure

This document discusses various aspects and approaches already performed in the subject under study, followed by explaining in detail the method developed, in order to facilitate the third-party use of the algorithms/methods. The document concludes with the validation of the approach, and subsequent result analysis. To this end, it is composed of 8 different chapters.

The first chapter (this one) introduces the thesis context, as well as the ultimate goals proposed.

The second chapter sets out the steps and strategies already developed in the context of autonomous driving in general. It also presents some science competitions that promote the theme, with special emphasis on the autonomous driving competition held at the national robotics festival.

Chapter 3 describes the main computer vision methods and concepts used throughout this work.

On the fourth chapter, it is illustrated the general organization of the system in terms of hardware, software and physical structure. It also details the previously existing platform, as well as the modifications performed to it, in order to meet the system requirements.

The fifth chapter revels in detail the most important sensory part of the system – computer vision. This chapter begins by setting out briefly the low level methods used and the process of image formation and acquisition. Subsequently, it details how the interesting features of the track are extracted.

The vehicle navigation system is detailed in Chapter 6, starting by briefly introducing the considered dynamics model, followed by the explaining the approach followed to perceive the world. This chapter ends detailing the approach followed to perform actions planning.

On Chapter 7, it is demonstrated the set of tests carried out in the developed platform, as well as result analysis.

Finally Chapter 8 presents the final conclusions of this thesis as well as possible future work.

Chapter 2

Autonomous Driving

2.1 - History of autonomous driving and robotics

The design of autonomous vehicles (also known as "driverless cars") has been increasingly developed until the present day, with important milestones over the past decades. The first steps in this direction required special hardware and software which resulted in slow systems with poor responsiveness. However, its contribution to scientific knowledge for robust and reliable systems available nowadays is clearly important.

Before modern systems based in a strong software component emerged, mechanical feedback systems took their first steps towards the automation of the driving process of a vehicle. James Watt (1736-1819) began by designing a throttle-valve and centrifugal governor which allowed automatic speed control of rotating machines. This invention was applied only to steam engines, appearing in cars by 1910 [2]. Ralph Teetor (1890-1982), was the responsible engineer for the invention that originated the cruise control systems used nowadays. His invention was inspired after a bumpy ride in a car driven by his lawyer, who being an avid talker had sudden accelerations and decelerations while talking [3]. His invention was first commercially assembled by Chrysler Imperial in 1958 in the New York and Windsor models. By 1960, all vehicles in the Cadillac brand were offered the cruise control system [4].

The history of autonomous driving itself began in 1977 in the Mechanical Engineering Laboratory in Tsukuba, Japan. The system consisted in an adaptation of a normal vehicle through special hardware to control it, since computers available at that time were much slower than today. It was a very rudimentary system and was based on following white lines marked on dedicated and cleared marked course, built for that purpose. The maximum speed achieved was 30 km / h [5].

In 1980, Ernst Dickmanns, and his team at UniBM (*Universität der Bundeswehr München*), were the pioneers of dynamic machine vision in autonomous driving. The system consisted in a re-engineered Mercedes-Benz van using saccadic vision, probabilistic approaches such as Kalman filters, and parallel computers. For safety reasons, the tests were held in traffic-free streets, reaching 100 km / h. The results achieved resulted in a very significant breakthrough in the field of autonomous driving [5]. Consequently, the European Commission initiated a major funding program in the field of autonomous driving, creating the largest autonomous

vehicle project so far, specifically in the EUREKA Prometheus Project launched by Daimler-Benz AG (1987-1995) [6].

One of the interesting results was achieved through the demonstration of VaMP and Vita-2 robots from Dickemanns and his team, who drove more than a thousand kilometers on a Paris multi-lane highway, with real and congested traffic conditions. The vehicle reached speeds up to 130km/h, using dynamic and real-time vision to detect up to twelve other cars and further avoidance. Although operated with some human intervention, it aimed to demonstrate autonomous driving in free lanes, convoy driving, and lane changing with autonomous passing of other cars in the left lane [7, 8].

This European project culminated in 1995 with the autonomous S-class Mercedes-Benz also from Dickmanns, which went through a 1678 km journey from Munich to Copenhagen and back on German public highways. The robot reached speeds up to 175 km/h with a mean time between human interventions of 9km, i.e. 95% of autonomous driving. Again, it detected other cars and performed maneuvers to overcome them [8, 9]. One of the greatest contributions to the area, was that called as the 4-D approach where spatiotemporal models the fourth dimension of a Kalman filter, through the incorporation of a world space model (3D) in time (1D), recursively adapted [10].

That same year, one project at the Carnegie Mellon University in the United States, obtained 98.5% of autonomous driving, on a trip entitled "No Hands Across America" where around 4800 km were travelled. It used a computer program called RALPH (*Rapidly Adapting Lateral Position Handler*) which used images to determine the location on the road and thus adjust the steering direction. However, the vehicle was in its nature semi-automatic, as neural networks were used for steering control, but the speed control was human-operated [11].

Also under the EUREKA Prometheus Project, another project arose from the *Artificial Vision and Intelligent Systems Laboratory* (VisLab) of the Department of Information Technology (Parma University), in Italy. A prototype called Moblab (MOBile LABoratory) in collaboration with the Polytechnic of Turin was developed. When the European project ended the research in VisLab continued, funded by the Italian National Research Council, and several prototypes were developed until now, such as ARGO, TerraMax and BRAiVE (among others, available in [12]).

ARGO consisted on a modified Lancia Thema equipped with specific peripherals for autonomous driving. It was developed during 1997 and 2001, and it was the first fully autonomous vehicle developed by VisLab. The first public demonstration of this vehicle occurred in 1998 in the tour *"MilleMiglia in Automatico"*. It aimed to show the scientific community and to the general public that it was possible to perform autonomous driving with a vehicle only through visual information and low-cost general purpose hardware. The most important sensory part consisted of only two low-cost black-and-white video cameras, and their analysis was based on stereoscopic vision algorithms. The tour covered 2000km throughout Italy, where 94% of distance traveled was fully autonomous [12, 13]. The basic constituents of the system are illustrated on the image of Fig. 2.1.

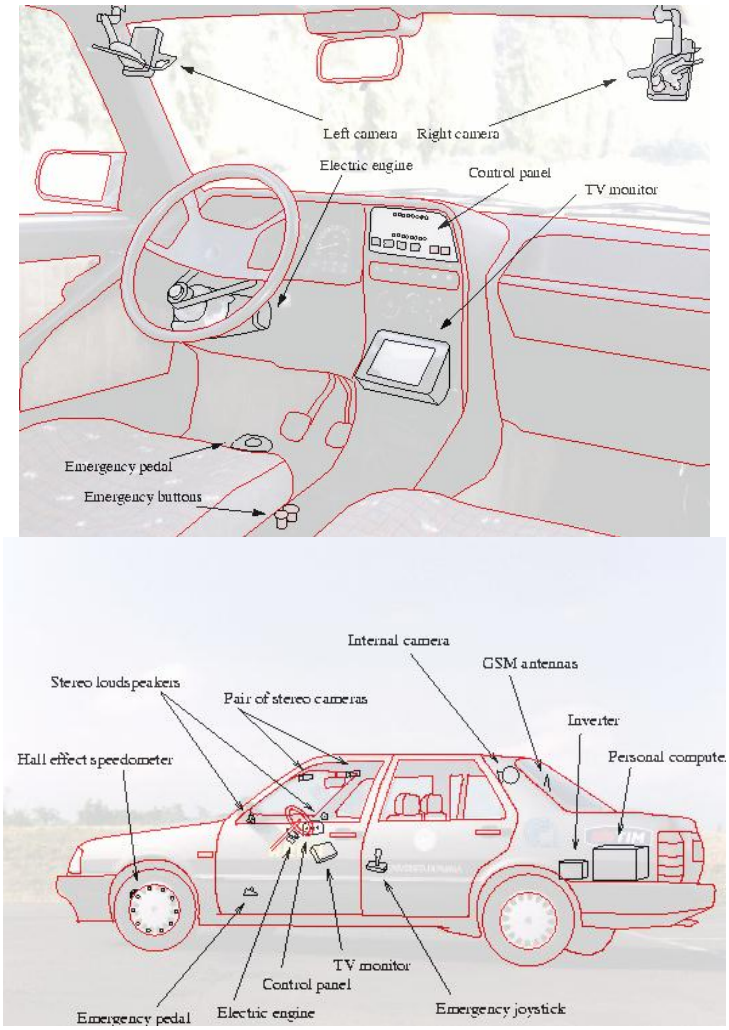


Fig. 2.1 The upper image shows the interior of ARGO and its main components, while the bottom image shows its exterior [12]

The ARGO project was the forerunner of the several projects of its institute and attracted the world attention to the real issue.

The TerraMax project[14] is composed of three autonomous cargo hauler vehicles to participate in the DARPA Grand and Urban Challenges (more information about this competition follows ahead). In its first edition, TerraMax 2004, completed only 2 km of more than 210 km of the course, and the vision system of VisLab was ultimately not used. In the second edition, TerraMax 2005 belonged to the group of the only five teams managed to finish the race, covering 220 km in total autonomy. The last vehicle of its series, the Terra Max T2, competed in DARPA, but did not finish the race.

In the United States in the early 1980s, the Defense Advanced Research Projects Agency (DARPA) created a robot called ALV (Autonomous Land Vehicle) that achieved the first approach for road-following, using laser radars, computer vision and a robotic mechanism control, reaching a maximum speed of 30km/h. In 1987, Hughes Research Labs demonstrated the first use of a system capable of autonomous driving in off-road environments applied to the ALV vehicle. The demonstration conditions that the vehicle faced, was a journey of 600m at an average speed of 3km/h with ravines, rocks and vegetation.

The Navlab (Navigation Laboratory) group created cars, trucks and buses with some autonomous driving capabilities, and driver assistance systems. Since 1984, 11 vehicles were

built from Navlab 1 until the Navlab 11 [15]. The main research areas have been: off-road scouting, autonomous driving in highways, run-off-road collision prevention, and driver assistance for maneuvering in crowded city environments. Their current work involves pedestrian detection, surround sensing, and short range sensing for vehicle control. The latest Navlab project, the Navlab 11 was implemented in a Jeep Wrangler encompassing these features and architectures.

BRAiVE (short form of BRAin-drIVE) appeared in 2009, and it was demonstrated for the first time on the first week of June, at the IEEE IV conference in Xi'an, China. This is a vehicle totally developed by VisLab and combines the knowledge and systems created by the group over the past 15 years. Its main goal was to test innovative concepts, consisting of a sensory fusion of a total of 10 cameras, 3 single plane laser scanners, one 4-plane laser scanner, 16 laser beams, GPS and IMU (Inertial Measurement Unit) [16].

In 2010 VisLab organized a trip of 13,000 km with a 3 month duration called VIAC (Vislab Intercontinental Autonomous Challenge). For this purpose small and electric vehicles were selected, namely the Porter model produced by the Italian scooter company, Piaggio. The main idea was to test the technology acquired over the years of experience through a journey in extreme conditions (including all sort of traffic, weather conditions, road infrastructures, and even off-road), with this type of urban vehicles. Two autonomous vehicles were moving at the same time during the trip, but either one with different goals: The first vehicle drove through most of the journey autonomously, functioning as a test platform, in terms of sensing, decision making and control sub-systems. Data was collected along the track, with some human intervention, for route definition and intervention in critical situations. The second vehicle followed the route defined by the first, without any human intervention (100% autonomous). This vehicle had the purpose of "learning", collecting information for further use in a set of vehicles to use in harsh environments. On the structural, sensory and control point of view both vehicles are identical. The system incorporates three laser scanners based on single plane laser beam to detect obstacles in the immediate surroundings of the vehicle , combined with an off-road laser scanner enabling the vehicle to drive in off-road conditions (where there are no lane markings) returning information from terrain ahead of the vehicle. A set of five cameras were placed in the front, providing a panoramic vision system to detect obstacles, terrain slope and markings on the streets. There are two other cameras in the back, used to detect obstacles close to the vehicle. All the automatic system is powered by a solar panel that stores energy in an auxiliary unit for the situations in which the sun coverage is not sufficient [17]. One of these vehicles is shown in Fig. 2.2.

Fig. 2.2 The Vislab prototype used in the VIAC tour [17]

2.2 - Autonomous Driving Competitions

Over recent years, some autonomous driving competitions have emerged all over the world, aiming to encourage research teams to the area of autonomous driving. Within these competitions, DARPA is undoubtedly the competition with more emphasis. It was first organized in 2004 targeted to US-based teams, in which the team would have to produce a vehicle that could navigate fully autonomously, reaching the target in a minimum time. It was organized at the Mojave Desert in the United States and the competition was 228.5Km long with no obstacles in the close proximity of the route. None of the 25 participating teams achieved the goal point, and the best mark was of 11.84km. On the 2005 edition, five teams finished the course and four of them under the 10 hour limit. Of all the participating teams, only one did not achieve the best mark reached in the previous edition [18].

Following the success of the Grand Challenge, DARPA organized an event called DARPA Urban Challenge held in November 2007. This event required the teams to build an autonomous vehicle capable of driving in traffic conditions, performing complicated maneuvers, such as merging, passing, parking, and negotiating intersections. This was the first autonomous driving competition in which vehicles had to interact with manned and unmanned vehicles in an urban environment. The journey involved a 96 km trail along an urban area. The vehicles had to comply with traffic rules, while avoiding other vehicles that shared the same route. Out of the 35 participating teams from around the world, six teams managed to finish the race, with an average speed of 20km/h. It should be noted that although there was no prior course rehearsing, there is a precise mapping through about 3000 "way-points, with plenty of way-points per curve, which allows the extensively use of GPS by the teams [18].

Although, not exactly a competition as the U.S. DARPA, the ELROB event (European Land-Robot Trial) is also an important European event about the demonstration of autonomous driving in robotics. The first ELROB was organized in May 2006 by the German Federal Armed Forces and was held in Hammelburg. The aim was, above all, to encourage the development of unmanned ground vehicles to military applications. This event alternates each year between the military and civilian components: C-ELROB (Civilian ELROB) and M-ELROB (Military ELROB). The civilian scenarios are unknown until the start of the race, and the location of the route is given by GPS points and aerial images. Along the trail, there are specific landmarks accordingly marked. Within the task of reconnaissance and Surveillance, the robot must correctly identify these points and localized them. Along with this there is a task of camp security in which robots guard one predefined area. During the competition, properly marked "intruders" enter this area and should be detected, tracked and pursued. The first civil event occurred in 2007 and the length of the autonomous navigation track was several kilometers long, encompassing three versions of Reconnaissance and Surveillance. The first was placed in an urban area, and the second in a nonurban area, which required off-road mobility of the robots. A total of 14 teams participated in the event: nine from Germany, two from Poland one from Portugal and one from Switzerland. The 2009 civil edition took place in Oulu, Finland in June. The scenarios were similar to those found in previous editions, and there were a total of 10 participating teams.

The military ELROB is dedicated to the realization of autonomous systems capable of meeting the requisites of common military tasks. Therefore, the scenarios simulate real-world missions as close as possible, matching the challenging level of actual military missions,

and the robot tasks must performed with maximum autonomy, although teleoperation is admissible. The first edition, held in May 2006, took place at the infantry school of the German Forces, and the scenarios were concentrated in autonomous driving. The results were somewhat disappointing, given that the teams had to fight against hardware problems, and many were widely operated remotely via WLAN. There were a total of 18 participating teams: 11 from Germany, three from the UK, two from Switzerland and two more from Portugal and France. The 2008 M-ELROB happened in the same place in June, showing that UGVs (Unmanned ground vehicle) are ready for their development under a semi-autonomy basis. 17 participating teams were involved: 11 from Germany, two from the UK, one from Italy, the Netherlands and two from France and Finland [19].

2.2.1 - National Robotics Festival

The National Robotics Festival (*FNR*) had its first edition in 2001, and aims to promote science and technology among young people of primary, secondary and higher education, as well as the general public, through robotic competitions. The festival, which takes place every year in a different Portuguese city, includes a scientific meeting where national and foreign researchers in the field of robotics come together to present the latest results of their activity.

This event has had since its creation a tremendous growth both in the number of teams and participants, as well as in terms of audience. The National Festival of Robotics is currently an initiative of the Portuguese Robotics Society [20] *(SPR*).

The next event will be held in Lisbon, at the "*Instituto Superior Técnico"*, in April 2011 [21].

One of the competitions covered in this festival since its origins is the competition of autonomous driving, in which the rules and characteristics were considered as the testing platform of this dissertation. At the beginning of this work, the publicly available rules were those of last year, 2010. Therefore, these rules were the ones considered, although in its genesis they remain the same from year to year. The 2010 rules are publicly available in [22].

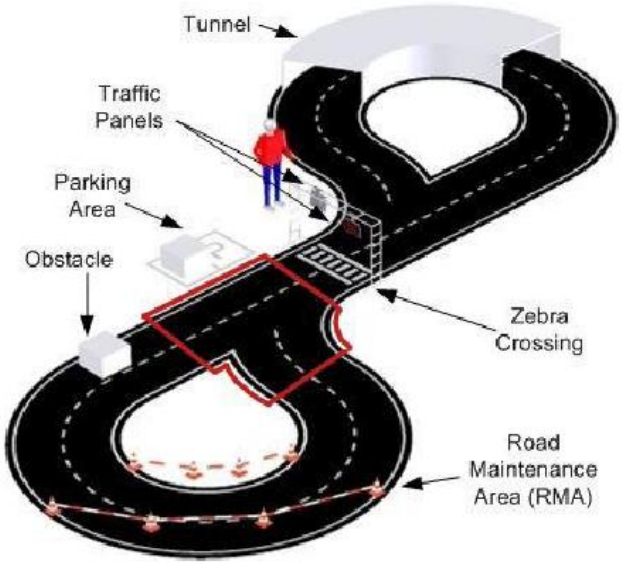


Fig. 2.3 The competition configuration in the National Robotics Festival [23]

It takes place in three rounds, being the main objective to go twice through the entire track in the shortest possible time, with no penalties. The difficulty degree grows for each new round: the first leg deals with a time trial over two laps on the track; in the second leg the traffic lights are added; in the third leg, a tunnel and a simulated work zone are added.

The upper signaling panels indicate the robot what action to take once it crosses the crosswalk, namely: end trial, stop, go straight ahead on the intersection, park and , turn left on the intersection, shown (in this order) in Fig. 2.4.

Fig. 2.4 The five possible signals place on the track

2.2.2 - Related Work

Throughout the various editions of the competition, several solutions have been presented which are worth highlighting.

ROTA Project

The “ROTA” project (Tricycle autonomous robot) began participating in 2006 and reached the 3rd place, and 2nd in 2007. The navigation of the robot is based on following the right line markings, adjusting the steering angle with respect to the distance to the right line. This is determined by a coordinate transformation of image coordinates into the real world coordinates. This solution was based on a reactive system with some memory and state notion. In 2008 it was also a participant in the competition, with minor upgrades in its

original platform, earning a 6th place [24]. Despite not taking place in the subsequent editions, the continuation of the work done in the platform should be noted. World representation work has been improved, through an intelligent system based on an abstract representation of a-priori knowledge of the world and the acquired information as the vehicle navigates. This is achieved through a hybrid model that combines the present section of the track and local metrics (odometry)[25]. Other vision approach was developed, based on removing perspective effect on the image, thus mapping the elements perceived in actual distances. In this context, new methods of self-localization and consequent intelligent planning were developed [23].

The approach used in the sign determination was based on color segmentation, narrowing the search with posterior analysis of any horizontal or vertical symmetry, with respect to the centroid of the resulting blob. However, this method has been reformulated to a template matching method, comparing along the various templates against the obtained image by the upper camera, and computing their correlation factor. Different template images are used for each signal, encompassing rotated templates. Thus, for small rotations and scale variations, the algorithm worked correctly [23].

The obstacle detection is performed by detecting great radial transitions, in the green space from the HSV color plane. These transitions are detected with a Canny edge detector filter. Frontal sensors are also used to perform obstacle detection [23].

The crosswalk is detected thorough the integrated odometry system.

ATLAS Project

The "Atlas" project should also be mentioned in this context. Its first appearance was in 2003, earning a 4th place. The vision solution encompassed a webcam looking at mirror, in order to obtain a visualization of the entire track. Thereafter it was continuously improved, showing a remarkable participation in the 2005 edition, with the Atlas III robot, earning the 2nd place. The initial approach was based on a vehicle with an Ackermann architecture, in which its control was based purely on image analysis. This analysis involves filling an area between the lines that define the track limits, and making decisions based on the position of the corners of the resulting quadrilateral [26, 27]. In 2006 and 2007, the ATLAS car was ranked first in the autonomous driving competition. In the 2008 edition the team introduced a new robot to the competition, the ATLASMV, designed to be smaller, lighter and faster than its predecessor, whilst competing with the earlier robot. This new robot was equipped with a laser range finder in 2009 to ease the task of obstacle detection and avoidance. ATLASMV is equipped with a distributed software architecture where multiple programs perform multiple tasks [28].

Throughout the various editions of FNR, the approaches used in the ATLAS project have changed and improved. The robots started by using only one camera, which was later replaced by two cameras with wide angle lenses to obtain a wide track view. The images coming from both cameras were merged into one through image transformations. In this approach, neither the modeling parameters of the lenses nor perspective transformations were performed, due to associated computational costs. Instead, a manual calibration of the distortion parameters is achieved manually for each camera, resulting in an image without geometric precision or consistency. However, in terms of navigation purposes, good results were obtained [26, 27]. The approach used most recently, uses a multi-camera platform mounted on a pan / tilt unit, to allow a more efficient world perception through active

vision. In this solution, all the cameras in the system are corrected by the pre-determined distortion parameters. Based on the kinematic model of the platform that incorporates four cameras, and through a perspective transformation of each camera, each image point is mapped to a point in the real world [29].

Regarding the world perception, two main algorithms were developed for road segmentation. The first takes advantage of the road homogeneity, i.e., the connectivity between border lines. For this purpose, a virtual horizontal line is demarcated defining thus, the allowed navigation area. The formed trapezoid is the basis-concept of the robot navigation, defining exactly which pixels belong to the road. This approach also allows the easy definition of the curvature region in the road [27]. Another approach was developed, being this a more robust and accurate solution used by the latest robots, taking advantage of the integrated multi-camera system. The analyzed image is a top view of the road, obtained through perspective transformations and multi-camera arrangement. The algorithm performs a search for lines, obtaining statistical indicators which are compared with a model, to infer the actual line presence [30].

Concerning the crosswalk detection, it consists in performing a search for a similar crosswalk pattern, or some relevant part of it. This search is delimited to the region enclosed by the lane markers. Hence, any of the previous algorithms for road segmentation are suitable for this end. With the first camera arrangement its detection was based in analyzing white blobs of pixels, and further computation of the form factor (area/perimeter) to infer the crosswalk presence. With the multi-camera arrangement, the undistorted perspective image is correlated with crosswalk template previously obtained. In order to overcome the limitations of the template matching detection method (sensitiveness to patter scaling and rotation), the crosswalk template is rotated according to the robot position. Its angle is obtained by computing the angle of one of the lane markers with respect to the bottom of the image.

As for the obstacle detection, the first vision approach used in the first robot assumed the obstacles as an integrant part of the line, since the obstacles were painted in white in the 2006 edition. However, they were later changed to green boxes. This fact was overcome through HSV color filters, and the further centroid computation to derive the position of the object [27].

The signal detection is done through the color and shape of the signal by computing the image into HSV components. Then, the resultant blob is evaluated based on the region centroid and the minimum involving rectangle.

VERSA Project

Another important participant was the VERSA robot (2005), equipped with vision and encoder sensors to perceive the world. The system starts by running a region labeling algorithm. Then the noise is filtered out and the centered moments of the resultant objects (center of mass, area, and orientation) are computed to distinguish the lines. The absolute position of the robot is computed by measuring the distance to these lines. This way, the navigation is based on object bounding, by mapping the detected lines into real world coordinates [31].

The crosswalk is assumed to be present if during the search for objects two thin and horizontally disposed regions (line markings) exist. The signal determination is based on

computing the moments on the image once it has been binarized, and comparing them to a-priori established characteristics.

The determination of the signal state is made following the earlier bounding of the various objects in the scene. Thus, for each of the objects, the resulting moments are compared with those moments known a-priori about the possible signals to determine the present sign.

2.3 - Chapter Conclusions

This chapter covers some of the work done under the autonomous driving in general, as well as its more evident historical development. In order to contextualize this work more specifically, some of the existing competitions are covered, particularly the FNR. The most important characteristics of the competition are exposed as well as three participants of clear success. The approach followed in these cases is overviewed, so that they can be compared later with developed method in this work.

Chapter 3

Computer Vision: Concepts and Methods

This chapter presents the common techniques and tools used in digital image processing. The low-level processing techniques are designed to reduce the amount of information in the original image while preserving the structural and interesting information for further processing. This can be either by removing part of the data available or by emphasizing the interesting regions. Thus the computational cost is reduced drastically, as well as it facilitates the task of determining required features in the image.

3.1 - Grayscale

Image processing aims to remove unwanted features from images. These features are often independent of color, which makes this aspect unnecessary, not only because it uses excessive memory space, but it is also more computationally expensive to operate on color images. Moreover, many operators work only on images in grayscale. The color model returned by most color cameras is the RGB, i.e., with three color components: Red, Green and Blue. The RGB color is based in human perception of colors and therefore there is a strict relationship between the two color spaces – the RGB and the grayscale [32]. The gray value of a certain pixel is determined by the perceptually weighted formula shown in equation 3.1.

$$Y(x,y) = 0.299 \cdot R(x,y) + 0.587 \cdot G(x,y) + 0.114 \cdot B(x,y) \quad 3.1$$

where:

$$R : the\ red\ component\ of\ the\ (x,y)\ pixel$$
$$G : the\ green\ component\ of\ the\ (x,y)\ pixel$$
$$B : the\ blue\ component\ of\ the\ (x,y)\ pixel$$
$$Y : the\ resultant\ gray\ value\ of\ the\ (x,y)\ pixel$$

3.2 - Convolution

Most of the filters used in computer vision use a convolution matrix. Convolution is an operation in which the value of the final pixel is the weighted sum of the neighbouring pixels. This convolution operation is based on a matrix which adds weight to each one of the neighbour pixels. Convolution is thus, the treatment of a matrix by another one which is called the convolution kernel. This matrix is typically a square 3x3, 5x5, 7x7 dimension matrix and so forth depending on the wanted result.[33]. It is for all purposes of a linear filter, as each resultant pixel is a linear combination of the neighbour pixels in the original image.

In Fig. 3.1 it is depicted a 3-by-3 kernel with weights w_1 to w_9. The result of this operation is illustrated in equation 3.2, for a generic point in the original image - $f(x,y)$. [34]

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Fig. 3.1 A generic 3-by-3 kernel

$$g(x,y) = w_1 \cdot f(x-1,x-1) + w_2 \cdot f(x,y-1) + w_3 \cdot f(x+1,y-1) + w_4 \cdot f(x-1,y) + w_5 \cdot f(x,y) + w_6 \cdot f(x+1,y) + w_7 \cdot f(x-1,y+1) + w_8 \cdot f(x,y+1) + w_9 \cdot f(x+1,y+1) \quad 3.2$$

3.3 - Smoothing

Smoothing is an operation widely used in statistics and image analysis that aims to eliminate noise, and filling in of missing data values. In the context of image analysis it tries to attenuate image distortion reducing noise or camera *artifacts*. This type of operation is called by local operation, since each point of the resulting image is achieved by applying a transformation function that takes as arguments the same point of the original image and a set of neighbouring points. These points consist of a "window" of variable size (usually 3x3) centered in the point to be transformed. The transformation function may be a linear combination of values set through the window (convolution with a kernel), a logic function or a nonlinear combination of the points in the vicinity of that to be analyzed.

In order to perform smoothing in a given image, several methods exist with different results. Therefore the method should be chosen carefully so that it causes the desired result in the image. All smoothing techniques are effective at removing noise of an image, but adversely affect edges, which might be undesirable. The best known methods are those of *simple blurring*, *median filtering* and *Gaussian blurring* (or *filtering*). For all of them, it is a defined window in which the operations are carried out. It is also possible to use more complex patterns of the window, such as the *cross* patterns.

The *simple blur* is the simplest case, and the less computationally expensive. Each pixel of the image-result is an arithmetic average of all pixels around the corresponding pixel in

the original image. Thus, it is performed through a convolution kernel as defined as in Fig. 3.2.

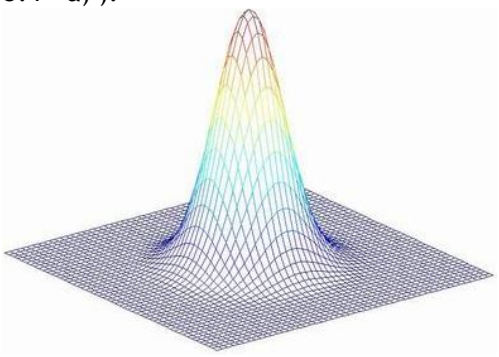
1	1	1
1	1	1
1	1	1

Fig. 3.2 A 3-by-3 Kernel for *simple blur* operation

The *median filter* is a nonlinear digital filtering technique that removes noise while preserving edges, under certain conditions. The algorithm sorts the values of the original image defined through a window. The pixel value of the resulting image is then the central value after this sort. Regarding the preservation of edges in the image, it has been proven that , the median filter is better than *Gaussian blur* at removing noise whilst preserving edges for a given, fixed window [35]. Even so, for higher levels of noise, it is only partially effective. Another disadvange is that it is more computationally expensive, especially because it involves sorting.

The Gaussian smoothing is performed by convolving each point in image, with a Gaussian kernel, exemplified in Fig. 3.3-b). The kernel used follows the Gaussian distribution and is therefore defined as a 2-D distribution of the Gaussian probability density function. It depends on the displacement of a point with respect to the center of the kernel, and the standard deviation, as shown in Equation 3.3. Gaussian smoothing provides is widely used in operations to preserve edges, acting as a low-pass filter removing high spatial frequency components from an image. Although this filter returns the best results in most cases, it is computationally expensive.

Fig. 3.4 shows the effect of these three filters when applied to the same input image (Fig. 3.4 - a)).



a)

$$\frac{1}{273}\times$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

b)

Fig. 3.3 a)An example of 2-D Gaussian distribution with (0,0) mean and σ=1.0; b) The discrete approximation of the Gaussian distribution in a) in a 5-by-5 Kernel

$$G_{2D}(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad 3.3$$

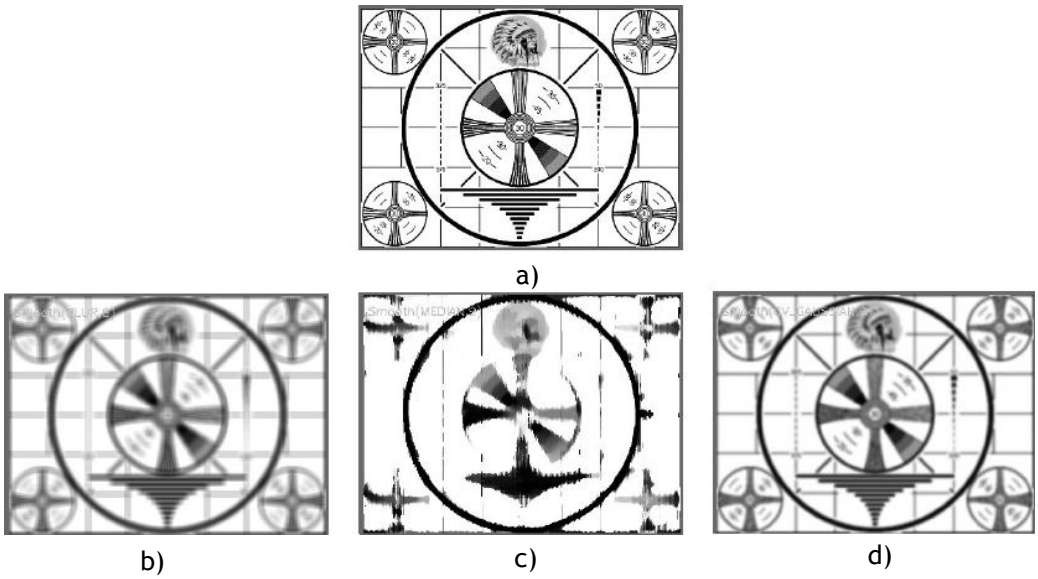


Fig. 3.4 Various image smoothing techniques applied to the same input image. a) the input image. b) resultant image after a block averaging filter. c) *median smoothing*. d) *Gaussian smoothing* [36]

3.4 - Image Binarization

In image analysis, the desired result is usually not another image, but its description, i.e. the most important aspects of it. The process of decomposing an image into its constituent parts is called segmentation, i.e., splitting the image into fragments.

Image binarization is used to remove parts of the image that fall within a specified intensity range, by converting an image of up to 256 gray levels to a black and white image. Frequently, binarization is used as a pre-processor before the main processing task. The simplest way to use image binarization is to choose a threshold value, and classify all pixels with values above this threshold as white, and all other pixels as black. This method is called Simple Threshold. However, in many cases, finding one threshold value compatible to the entire image is very difficult or even impossible. This happens mostly when there is a strong illumination or reflectance gradient, where it is needed to threshold relative to the general intensity gradient. Therefore, Adaptive Threshold is needed so the threshold level itself is variable for different regions in the image. This technique is set on a pixel-by-pixel basis by computing a weighted average of a region around each pixel location, minus a constant.

3.5 - Morphological Transformations

Morphology is a broad set of image processing operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its

neighbours [37, 38]. It aims to transform the images into simpler ones, to be later analysed. This operation is often done on binary images.

The two most common morphological operations are erosion and dilation, which can be combined together to achieve a more interesting image or region. These morphological operators used together provide two other valuable operators: opening and closing.

3.5.1 - Dilation

The basic effect of the dilation operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels. The value of the output pixel is the maximum value of all the pixels in the neighbourhood of the input pixel. For binary images, if any of the pixels around the resulting pixel is set to the value 1, the output pixel is set to 1. This results in growing the size of foreground pixels, while holes within these regions become smaller.

Mathematically, the dilation is a convolution between two pieces of data inputs. The first is the image which is to be dilated. The second is a set of coordinate points called structuring element, the kernel. It is then performed a local maximum operator. As the kernel is scanned over the image, the maximal pixel value overlapped by the kernel is computed. This value will then replace the image pixel value overlapped by the center cell of the kernel. This effect is illustrated in Fig. 3.5.

The dilation of A by B is denoted by A$\oplus$B and is defined by equation 3.4.

$$A \oplus B = \bigcup_{b \in B} (A)_b \qquad 3.4$$

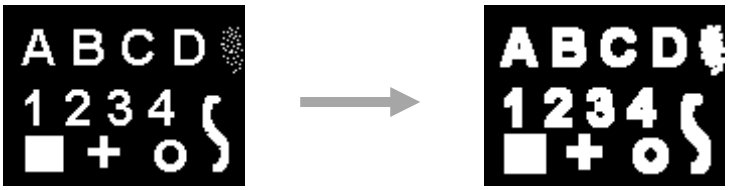


Fig. 3.5 The effect of dilatation on image A, under the kernel B [36]

3.5.2 - Erosion

Erosion is the inverse of dilation. In erosion, every object pixel that is touching a background pixel is changed into a background pixel. Therefore the basic effect of this operator on a binary image is to erode away the boundaries of regions of foreground pixels. Thus, areas of foreground pixels shrink in size, and holes within those areas become larger.

The action of the erosion operator is equivalent to computing a local minimum over the area of the kernel. As the kernel is scanned over the image, it is computed the minimal pixel value overlapped by the kernel. This value will replace the image pixel value overlapped by the center cell of the kernel. This operation causes brighter regions within an image to shrink. The erosion of A by B is denoted by A$\ominus$ B as shown in equation 3.5.

$$A \ominus B = \bigcap_{b \in B} (A)_{-b} \quad 3.5$$

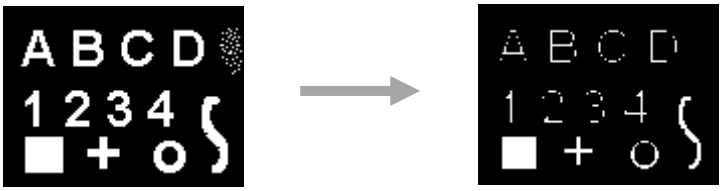


Fig. 3.6 The effect of erosion in an image A, under the kernel B [36]

3.5.3 - Opening and Closing

These operations are combinations of dilation and erosion operators.

The opening operation is about performing a dilatation operation proceeded by erosion, denoted by $A \circ B = (A \ominus B) \oplus B$. Opening is commonly used to make a more efficient counting of regions in a binary image.

As for the closing operator, it is used in most of the more sophisticated connected-component algorithms to ensure connectivity between segments. This disconnection may be due to noise, or due to the presence of unwanted segments, possibly generated by binarization. Essentially, the closing operator tries to merge neighboring regions that undesirably, may not be merged, by performing erosion followed by dilation on the original image. Its mathematical operations is defined as $A \odot B = (A \oplus B) \ominus B$.

Both operators appreciable preserve the area of the regions as the most prominent effect of these operators is changing the arrangement of regions in the original image.

Fig. 3.7-b) and Fig. 3.7-c) respectively show the effect of opening and closing when applied to the same input image, Fig. 3.7-a).

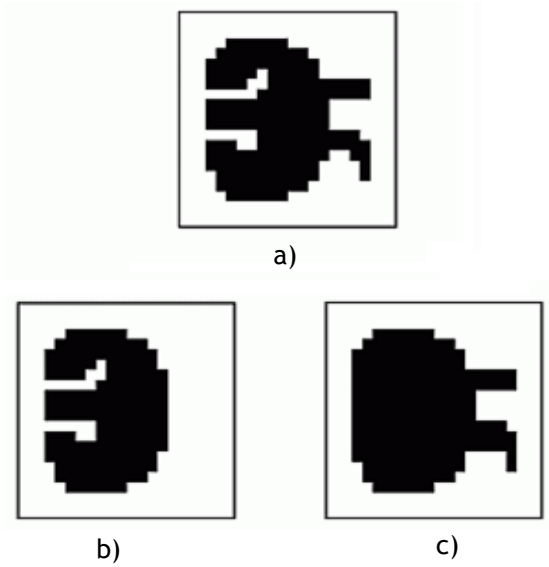


Fig. 3.7 The results of opening and closing operations the original image - a).In b) it is shown the effect of opening, and c) shows the resultant image after the closing operator.

3.6 - Hough Transform

The *Hough* transform is a technique used to isolate features of a particular shape within an image that has been *edge enhanced* previously. Because it requires that the desired features be specified in some parametric form, the *classical Hough* transform is most commonly used for the detection of regular curves such as lines, circles and ellipses [39] .

Due to imperfections in either the image data or image pre-processing, there may be missing points on the desired curves as well as spatial deviations between the ideal forms. This method performs groupings of edge points into object candidates by performing an explicit voting procedure over a set of parameterized image objects. This makes the technique tolerant to gaps and is relatively unaffected by image noise, under certain conditions.

The algorithm starts with a transposition of the points represented in the Cartesian (x, y) coordinate system of the original image to another space (ρ, θ). This conversion is based in the fact that, to represent a line in the plane (x, y), at least two points are required, while their representation in the (ρ, θ) plane is done through a point, as illustrated in figure Fig. 3.8. This conversion is done by re-writing the equation of a line as shown in 3.6. Rearranging this equation one obtains equation 3.7.

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{\rho}{\sin\theta}\right) \quad 3.6$$

$$\varrho = x\cos\theta + y\sin\theta \quad 3.7$$

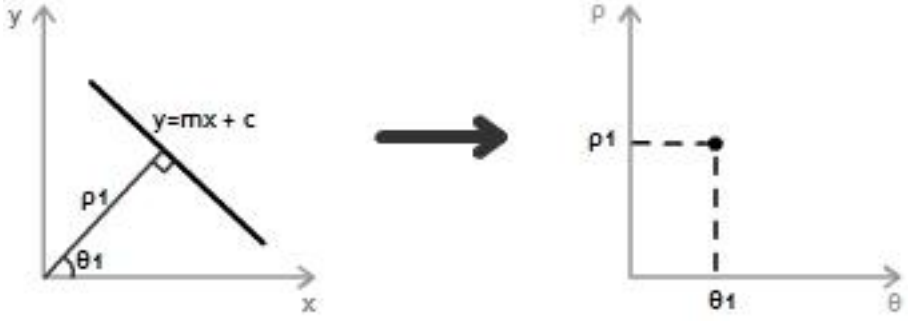
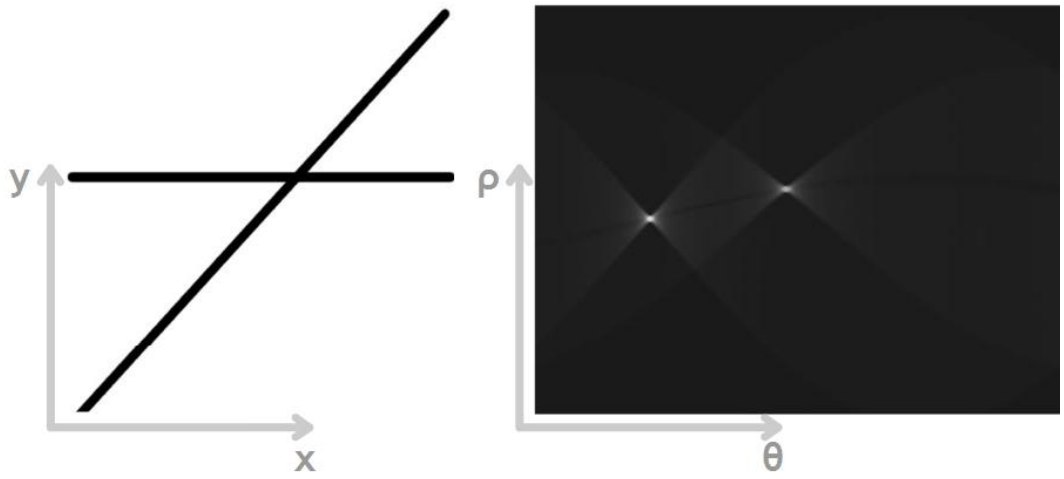


Fig. 3.8 Conversion from two space planes for a line

$$\rho = x_1\cos\theta + y_1\sin\theta \quad 3.8$$

As for the representation of a point in the image plane (x, y), its representation in the plane (ρ, θ) is a sinusoidal curve, as shown in equation 3.8. This curve represents the infinitude of lines that can cross the point in the image (x, y) plane. It turns out that for several points that define a line in the (x, y) plane, when represented in the (ρ, θ) space, the intersection of the several sinusoidal curves, results in a single point. This point of coordinates (ρ_1, θ_1) represents, as previously seen, a line in the (x, y) space. As seen in Fig. 3.9 the whiter points in Fig. 3.9-b), show the two points where the two curves intersect. These points represent the two lines shown in Fig. 3.9-a).



a) b)

Fig. 3.9 The result of the *Hough* transform in b), for the input image a). Adapted from [40]

The implementation of this method is based in an accumulator, which has the dimension of the unknown parameters of the Hough transform problem. In the simplest case, the linear Hough Transform problem, the two unknown parameters are *m* and *c*. The algorithm starts by creating a grid of dimensions ranging in θ from -90º to +90º and in ρ from 0 to the diagonal of the image. The size of each cell is defined by the resolution set for both parameters: ρ and θ. Then, for every non-null pixel found, the formula shown in Equation 3.8 is applied, for θ values in the range of [-90+,90º] and the corresponding point in the grid is incremented. Once the full image has been scanned, a search in the bins with the highest values is performed, by looking for local maxima in the accumulator space. The simplest way of considering that these peaks actually correspond to a line, is by applying some form of threshold, but different techniques may yield better results in different circumstances [40].

There are more complex versions of the Hough transform, in particular applied to the detection of curves and ellipses, and determining their corresponding length. However, the methods will not be addressed in this context, as they are just variations of the explanation just stated.

3.7 - Flood Fill

Flood Fill is an algorithm that determines the area connected to a node in an image. It is very useful to mark certain areas of an image or for isolation purposes, for further processing and analysis. It allows isolation of a processing area, thus reducing the computational cost, as further processing is restricted to a defined set of pixels. It is also used in large-scale paint programs to determine which parts of the image will be painted.

It takes a seed point (start node), the target color and a "replacement" color. The result of applying the algorithm is a colored, contiguous region. Next it is illustrated an exemplary algorithm of the method, with 4-connectivity, applied to binary images:

```
Flood-fill (seed, target value, replacement value):
1.  If value of node ≠ target value, Go to 4.
2.  Set value of node to replacement value.
```

```
3.  Flood-fill (1 step west of node, target value, replacement value).
    Flood-fill (1 step east of node, target value, replacement value).
    Flood-fill (1 step north of node, target value, replacement value).
    Flood-fill (1 step south of node, target value, replacement value).
4.  quit.
```

3.8 - Canny Edge Detector

The purpose of edge detection in general is to significantly reduce the amount of data in an image, while preserving the structural properties to be used for further image processing. Canny Edge Detection[41] is considered to be the ideal edge detection algorithm for images that are corrupted with white noise. This algorithm aims to assemble the individual edge candidate pixels into contours.

For that purpose, the algorithm starts by computing the first derivatives in x and y directions, combining them into four directional derivatives. The points where these directional derivatives are local maxima are then candidates for assembling into edges. The contours are formed by applying a hysteresis threshold to the pixels. So, if a pixel has larger gradient than the upper threshold, is accepted as an edge pixel. On the other hand, if a pixel is below the lower threshold, it is rejected. If the gradient of the pixel is within the thresholds, then the pixel will be accepted only if it is connected to a pixel that is above the upper threshold.

a) b)

Fig. 3.10 The result of Canny Edge detector in b), for the input image a)

3.9 - Template Matching

Template matching is a technique that creates a model of an object of interest (the template) and then it searches over a certain image for objects that match the template. It could be used also to check how “alike” two objects are [42].

The implementation on OpenCV is based on matching a template image against an input image by “sliding” the template over the input image, computing the correlation factors. There are several methods used to compute the correlation factors, but in the context of this work, it was used the normalized square difference matching method. This method was chosen do its low computational costs when compared to other methods. The resultant matching value is given by equation 3.9. Note that a perfect match returns a value of 0.

If the dimensions of images are equal, there is no "sliding", which means that R is a 1D matrix. Otherwise it the minimum of R is determined and its location reflects where the matching was higher on the input image.

$$R(x,y) = \sum_{x,y} [T(x',y') - I(x+x', y+y')]^2 \qquad 3.9$$

where:

$$I : input\ image \ ,$$
$$T : template\ image \ ,$$
$$x', y' : template\ positioning \ .$$

3.10 - Chapter conclusions

This chapter introduces some of the general tools used throughout the vision subsystem developed and detailed in Chapter 5. Although not all methods are present here, this chapter gives, in essence, an understanding of tools and concepts that are used abstractly in the computer vision subsystem.

Chapter 4

System Overview

4.1 - Vehicle Description

4.1.1 - Initial Setup

As previously stated, this work is the continuation of a previously developed. From the conceptual point of view, it consists of a low-level platform and its respective communication system that allows an upper level integration with other control layers. In order to interact with the system, it was also developed an application running in a PC, which allows the interaction with the various elements that constitute the vehicle.

Structure and Hardware

The final work preserves almost totally the basic structure provided by the manufacturer, with some added elements, essential for autonomous driving. The original control system of the vehicle consisted of a radio frequency operated transmitter / receiver pair, so that one could manipulate the vehicle remotely, controlling the speed and direction. From the electrical standpoint, the vehicle consisted of a traction motor, coupled to a differential system which operated on wheels; a servo motor, that acts on the steering system; and variable speed controller, acting in the motor. All these devices were connected to a communications module that received radio frequency commands from a remote control.

Fig. 4.1 The chassis of the vehicle [43]

In the image of Fig. 4.1 the vehicle chassis is shown in its original configuration, without any adjustments. In Fig. 4.2, some of the main features of the original vehicle are illustrated.

In Fig. 4.2 - a), the suspension system is shown, which is applied to all four wheels. It consists of adjustable *aluminum oil-Filled Shocks* [43] equipped with adjusters, in order to tune the hardness of the springs. In Fig. 4.2 - b), an image of the transmission system is shown, called *brushless-ready All-Metal Gear Transmission*. Its transmission ratio is 1:10, and it drives on the differential of the vehicle. As for Fig. 4.2 – c), the motor that drives the vehicle is illustrated. It is comes from the same manufacture of the vehicle, and it is called *12T 550-Sized Performance Motor*. Although the electrical or dynamical features are not available, it was designed for large accelerations and top speeds.

a) b) c)

Fig. 4.2 Some of the most important aspects of the initial structure of the manufacturer. Where in a) it is represented the suspension system; in b) the gear system; and in c) the traction motor [43]

This configuration has been modified so that the control was accomplished via a PC connected by RS-232 to a control board that not only receives control signals, but also sends control variables. These control variables are provided by sensors added to the structure, subsequently acquired and processed by the control board - the motherboard of the vehicle.

In order to quantify the distance traveled by the robot and, indirectly the speed of it, an encoder is coupled directly to motor shaft. This encoder has only one output signal, which makes it impossible to detect the direction of rotation. Fig. 4.3 shows its placement in the motor shaft. Since there were no planned sudden changes in the direction of rotation, this is fact did not compromise the use of this encoder.

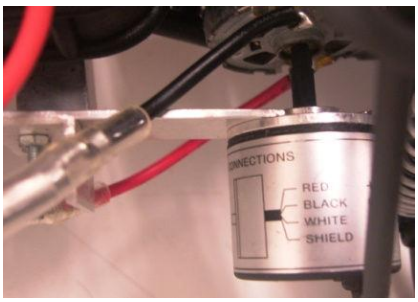


Fig. 4.3 The placement of the encoder in the motor shaft at the initial setup [1]

The powering of the robot was done with the battery belonging to the remotely operated vehicle provided by the manufacturer. It was an 8.4V operating nickel-metal hydride (Ni-MH) battery. This battery had the responsibility of feeding all the devices in the robot, including the motors, the drivers and the sensors.

Two sensors that detect white lines were placed in the vehicle. These sensors had the primary purpose of detecting the line boundaries of the track in extreme situations. The implemented solution is a set of two optical reflectors mounted in front of the robot, pointing to the ground and tilted slightly so that they inform the control system of the

detection of the lines. The infrared emitted from these devices reflected on the floor, and consequently detected by a photo detector. The emitter used was a photodiode, and the detector a phototransistor. In Fig. 4.4 the placement of the sensor in the frontal bumper is shown.

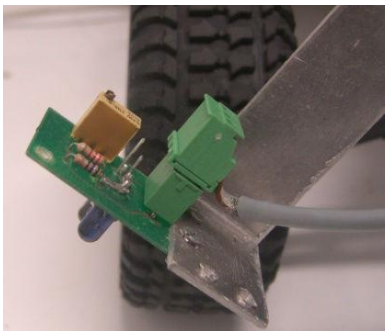


Fig. 4.4 White lines sensor detector placed in the structure of the vehicle [1]

In order to detect obstacles, infra-red sensors were placed in the structure of the vehicle. The sensors used were the *Sharp GP2D12* [44]. These sensors have a range between 10 and 80 cm and return voltages between 0 and 2.6 Volts. Two of these sensors were placed in the frontal bumper and two on each side of the vehicle. Fig. 4.5 illustrates the placement of one of these sensors in one of the sides of the vehicle.

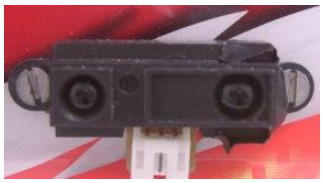


Fig. 4.5 The sensor that detects distances placed in one side of the vehicle [1]

The mainboard (see Fig. 4.6) was responsible for controlling the actuators and reading the sensors, as well as the interconnecting two layers of control. It was used an *ATmega8 Generic* module, developed at FEUP. This module incorporates an *ATmega8* microprocessor, a RS-232 communication interface, and also a voltage regulator that converts the battery voltage to 5 Volts, which is the voltage level required for most of the hardware elements. This module communicates with the pc via a RS232/USB converter and was placed in the rear side of the vehicle.

Fig. 4.6 The motherboard placed on the rear side of the vehicle

Software

There are two different software levels developed in the earlier work. These levels were denominated as low-level and high-level layers, respectively. The higher processing layer, implemented in a PC, is responsible for controlling the robot in a transparent way to the internal and specific characteristics of the devices. The control software was developed based on event-driven programming. In this layer the control cycle is triggered only from the moment that the serial port connection is opened. In order to operate with the board of the vehicle through the PC, a specific application was developed. This application was developed in *Lazarus*¸ a *free Pascal* cross-platform IDE [45]. It allows not only the interpretation of data coming from the microcontroller, but also sends the desired values for the traction motor and servo motor.

The raw data coming from the distance sensors is processed in the high level layer, in order to free processing resources on the lower level. The data is sent on a scale of 0 to 1023, as its conversion is set to have a resolution of 10 bit in the lower level. Secondly, there is a re-scale to volts and its value is placed on a mathematically model of the sensor. This function is a linearization curve that returns the distance given a certain voltage value. Thirdly a filter was implemented in order to improve the sensor accuracy. As for the sensors that detect white lines, no filters were applied to its returned value.

Through the number of steps obtained by the encoder, this layer calculates the approximate speed and implements also a relative location system, based on odometry. To this end, a kinematic model of the robot was computed, which takes two parameters - the distance traveled during a time instant and the steering angle. Hence, this layer returns two location parameters - the reference point in Cartesian (x, y) system and its orientation angle (θ).

A Graphic User Interface (GUI), shown in Fig. 4.7, was developed so that the user can motorize and control the motion of the vehicle.

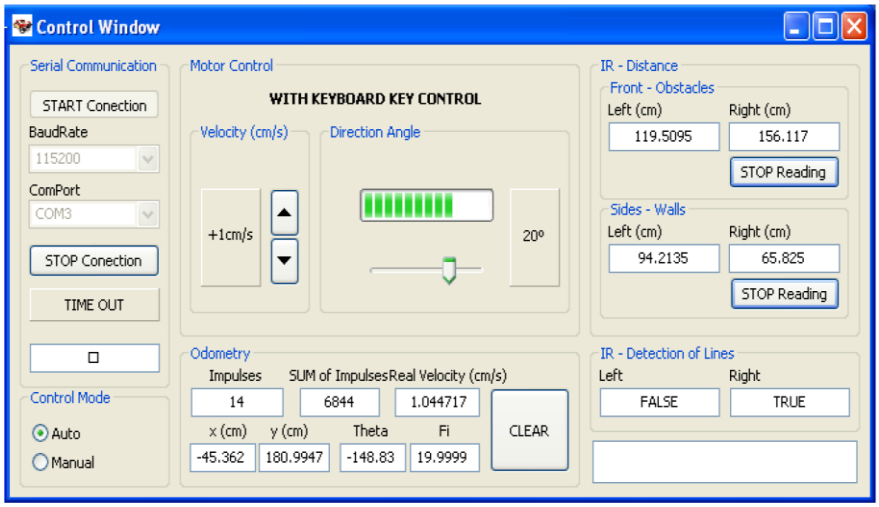


Fig. 4.7 The GUI developed in the earlier work [1]

The layer below is based on a microcontroller – *AVR ATmega8* from *Atmel*[46], programmed using the C language. The actions that demand a lower level processing, such as

reading the sensors and the generation of PWM signals responsible for controlling the actuators, are performed at this level.

The speed controller is responsible for operating the motor coupled to the wheels, ensuring that the robot is running at the reference speed established. So, in order to make the robot move at a certain speed, a reference number of pulses per sampling time is given. For that purpose the approximated dynamic model of the robot was determined and the controller was designed from a closed-loop PI controller, with input filter in a feed-forward configuration. In order to generate the PWM signals, the *ATmega8* features were used.

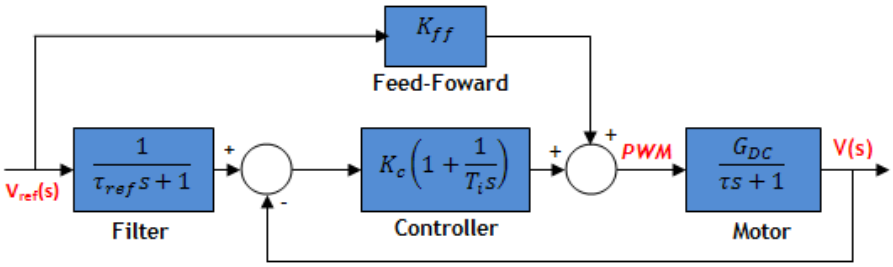


Fig. 4.8 The speed controller implemented in the earlier work. Adapted from [1]

As for the servo motor, its control was also performed by PWM signals generated also in microcontroller. The operating intervals of the motor [-45°, 45°] were divided in 91 different positions to obtain the desired pulse length for various steering angles of the robot. For greater resolution, the 16-bit timer was used and the PWM signal operated in a 20ms period.

In order to transmit the vehicle information from the PC to the controller, and vice versa, a communication protocol via RS-232 in a Master-Slave configuration was implemented. Broadly, the Master, the lower-level layer, sends data every 40ms, and under normal operation a frame is placed in a buffer and consequently sent to the Slave. Then, and before the time limit is exceeded, a response is received. After seven failed frames sent, i.e., no response was received, the robot enters a state designated time-out, where the traction motor is turned off. In addition, the protocol also implements a way for verifying the correctness of data, through a check-sum algorithm.

4.1.2 - Modifications and improvements

Structure and Hardware

For the purpose of this work, several changes had to be done. It should be noted that this was a vehicle in its nature, prepared for difficult environments - mainly off-roading, including jumps (whose height exceeds large times the height of the vehicle), water environments, great speeds of operation and high accelerations/decelerations. However the nature of the competition bears no resemblance to the features in the vehicle, so in certain situations it hindered the operation of the vehicle. These modifications were performed not only to address certain obvious shortcomings both at a structural and electrical level, but also to include structural support for certain features not taken into account previously, such as, the vision based system.

After a first contact with the platform, it was noticed poor electrical contact on the mainboard with the peripherals. This fault is critical especially due to the motion-

characterized system. Since the original board was a generic one, i.e. not adjusted to the particular application, certain devices were not plugged into PCB connectors; instead they were connected (welded) directly on it, which made the board, not so robust and versatile. To correct this flaw, a new PCB was designed specifically for the application. For that purpose the number of I/O peripherals was taken into account, as well as their physical location. Fig. 4.9 shows the new designed motherboard.

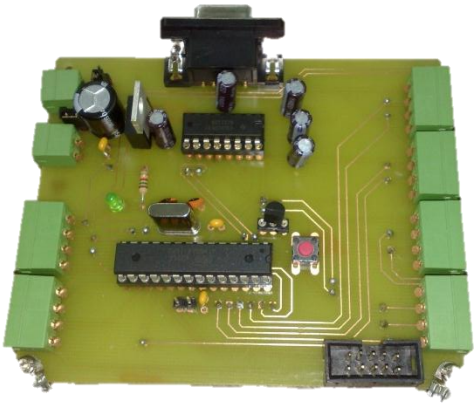


Fig. 4.9 The new PCB board designed with its components placed on it.

Another PCB circuit (see Fig. 4.10) was designed for the white line infrared detector sensors. This is due to the fact that poor electrical contacts were also found between the various components.

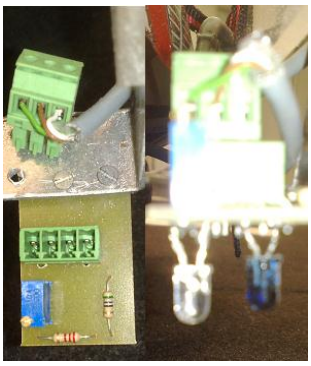


Fig. 4.10 The PCB board developed for the white lines detector circuit.

A horizontal bar made of acrylic was placed in the front bumper of the vehicle. This allows an improved position of the infrared sensors to detect distances. This way, these sensors are completely parallel to the ground. This configuration also allows the placement of the infrared sensors to detect white lines at greater distances from the center of the vehicle, thus anticipating a possible situation in which the vehicle steps lines.

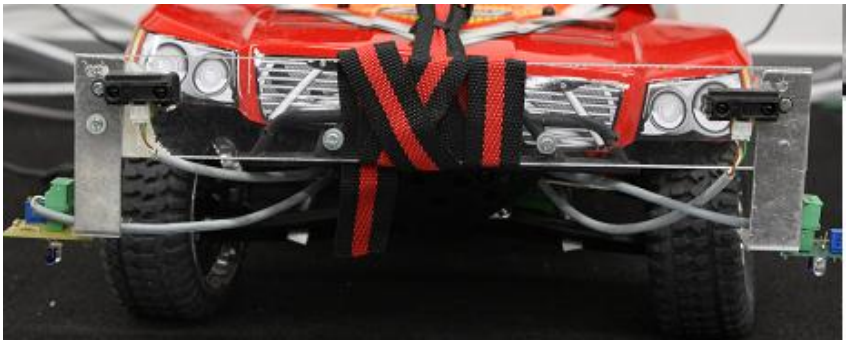


Fig. 4.11 The horizontal bar in acrylic placed in front bumper.

The controller driving the traction motor of the vehicle was also replaced. This is due to the fact the original one was only constituted by a transistor, which base was controlled by PWM pulses coming from the motherboard. This simple setup is flawed in several aspects: the engine was only able to rotate in one direction; there was no current control, nor temperature control. Therefore, this controller was replaced by an H bridge (Fig. 4.12).

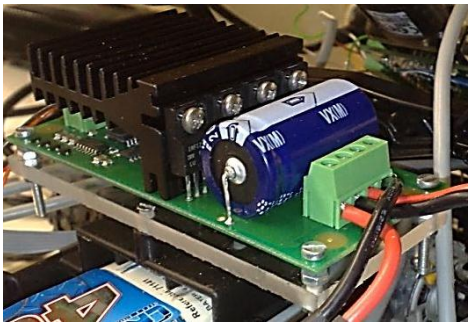


Fig. 4.12 Assembly of the circuit that implements the H bridge and its support in acrylic.

It should be noted that there was no design of this controller, since there was no information available on the characteristics of the motor, and this driver was already set to take the place of the previous driver before this work started.

It was then found, the maladjustment of the motor for the application in question. This is because the motor has not been designed to operate in the regimes of operation required by the application. Note that the motor was not changed after the adaptation from the recreational car. Its original purpose was to move the car at high speeds, reaching almost 60km/h in normal conditions. This made the motor very unstable and inefficient at low rpms. It turned out that at speeds around 10cm/s – 50 cm/s, the current went up to 15 A.

This was clearly a problematic situation in several aspects: the autonomy level was reduced considerably, a great voltage drop on the internal resistance of the battery, which meant that in critical cases, the voltage at the mainboard input reached a certain level that the voltage regulator did not function properly, taking the logic power down to 0 V. To correct this problem, it was opted for a motor specifically designed for slower speed applications. In this engine a planetary gearbox was fitted, with a reduction ratio of 4.8:1. The motor used was the *Maxon A-max 242467* [47].

Fig. 4.13 Assembly of the circuit that implements the H bridge and its support in acrylic. Adapted from [47]

This new motor, unlike the previous one, did not have its shaft available on both sides, which invalidated the placement of the encoder in the same position as before. This problem was solved through the design of a mechanical part with dual function. On the one hand, it strengthens the gear structure coupled to the motor shaft (See Fig. 4.14-a)), as Its inner radius had to be enlarged to fit the shaft diameter of the new motor, which was slightly larger than the previous one. On the other hand, it also eliminates an unintended effect of "slipping" between the motor shaft and the encoder. This problem previously existed since the two shafts (encoder and motor), were bound only by a rubber sleeve, which after some wear, started to loosen. A 3D model of this part is shown in Fig. 4.14-b.

In Fig. 4.14-c) an "exploded view" with all the components connected to the motor shaft is shown. From left to right: the motor with the gearbox; the gear shown in a) that drives the differential; the designed part that enclosures the gear, shown in b); a joint connecting the traction motor to the encoder, the last piece in the view. The assembled configuration is shown in Fig. 4.15.

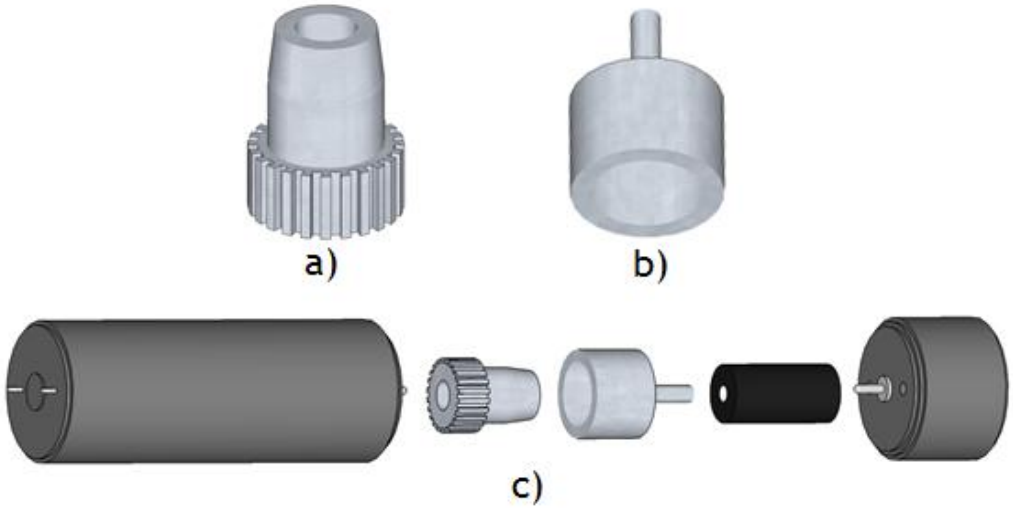


Fig. 4.14 The new mechanical system coupled to the original gearbox and the encoder. a) representation of the gear that existed previously in the motor shaft. In b) designed mechanical part. c) exploded view showing the coupling system to the encoder and gearbox of the engine.

With the goal of providing greater autonomy, and to fit the new power needs, the battery was also replaced to one having 12V in its output, with a 10Ah capacity. This battery is

composed of 40 *NiMH AA* 1.2V batteries, disposed in sets of four batteries in parallel placed in series with each other sets.

It should be noted that the encoder was also changed in order provide one more variable in its reading – the direction of rotation, as opposed to the previous one, which only delivered one channel in its output. Only small changes had to be done to make this adaptation. The supply voltage is greater than the previous one (12V as opposed to 5V), and the output type is NPN open collector. Therefore it was fed directly from the battery, and its outputs were also connected to the mainboard through a *pull-up* resistor. The encoder was the *Omron E6A2-CW5C* [48].

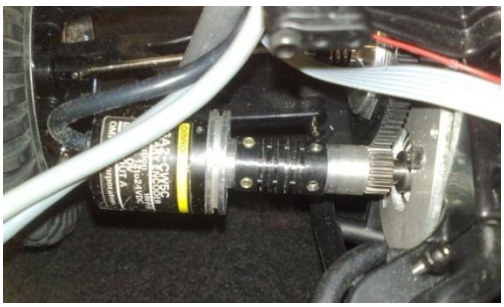


Fig. 4.15 Placement of the new traction system and coupling of the encoder.

With this new configuration a much more stable operation at low linear speeds is obtained, reaching only a maximum current of 2.5 A in normal operation. Thus, the controller was later changed to one with more appropriate features, and with a more robust control, as a monitoring of temperature and current is done turning it off if the values exceed a predetermined level. The controller chosen was the *Pololu Dual VNH3SP30 Motor Driver Carrier MD03A* [49], which has a continuous maximum output rating of 9A, a 30A of peak output and operates from 5.5V to 16V. The assembled controller is shown in Fig. 4.16.

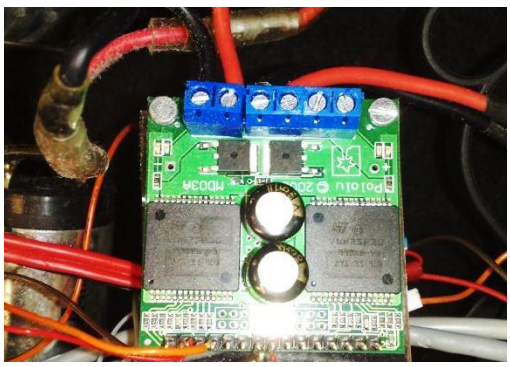


Fig. 4.16 The new driver that operates the motor.

Concerning the structure that holds the PC, it was also changed in order to provide not only more strength and rigidity to the structure but also to make the use of a larger laptop PC possible (up to 15.4 ‘’). Attached to this structure, two stands were also placed, to allow the placement of the cameras, as shown in Fig. 4.17. Note that the position of the cameras was chosen carefully, as they could capture the front of the car, or the screen of the laptop PC, and such elements are not an interesting part to the vision system.

Fig. 4.17 The left figure illustrates the new structure that holds the PC and the support structure for the cameras. In the right figure it is illustrated the positioning of the camera in detail.

Due to the suspension system that existed in the vehicle, it happened that the weight placed on the vehicle body (mainly due to the PC and cameras), moved the body down, closer to the chassis. In critical situations this made the body touch the wheels, which compromised the correct behavior of the system. To address this problem, the suspension system was substituted by handmade rigid metal links for all the four wheels. An example of such modification is shown in Fig. 4.18.

Fig. 4.18 The replacement of the suspension system by rigid metal links.

Software

Specific changes had to be made to the firmware implemented in the motherboard of the vehicle and also to the application running on the PC.

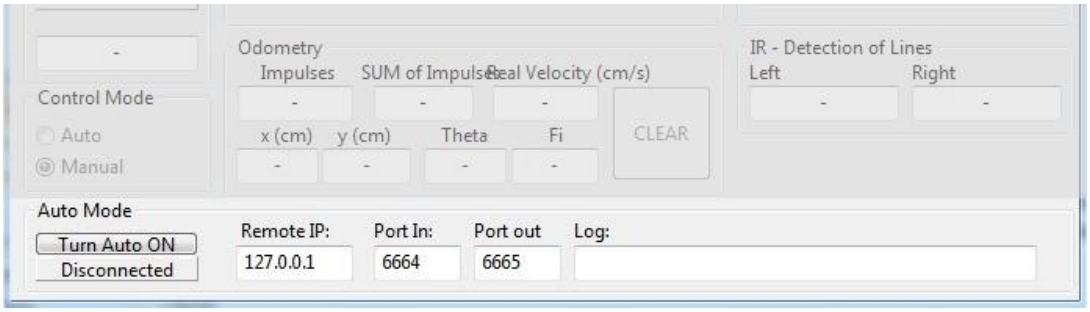


Fig. 4.19 The new design of the GUI that controls the vehicle motion, with the UDP settings in detail.

As for the firmware, due to the introduction of the new traction system (motor and gearbox), the gains of the digital speed controller were manually changed to improve the response of the controller. The new motor driver had several operating modes (explained briefly in the next section). These modes of the driver were therefore set by the motherboard firmware, which sends a signal to the physical controller.For the application that runs on the PC, a communication protocol based on UDP sockets was implemented, so that it can receive data from other applications, and possibly other PCs. This will be explained in 4.3 - Software Architecture. This new feature is configurable through the GUI, as seen in Fig. 4.19.

4.2 - Hardware Architecture

The hardware architecture is shown in Fig. 4.20. As the diagram makes clear, this system is composed greatly by a sensory component - represented by blue boxes. It consists of only two outputs – represented by the green boxes.

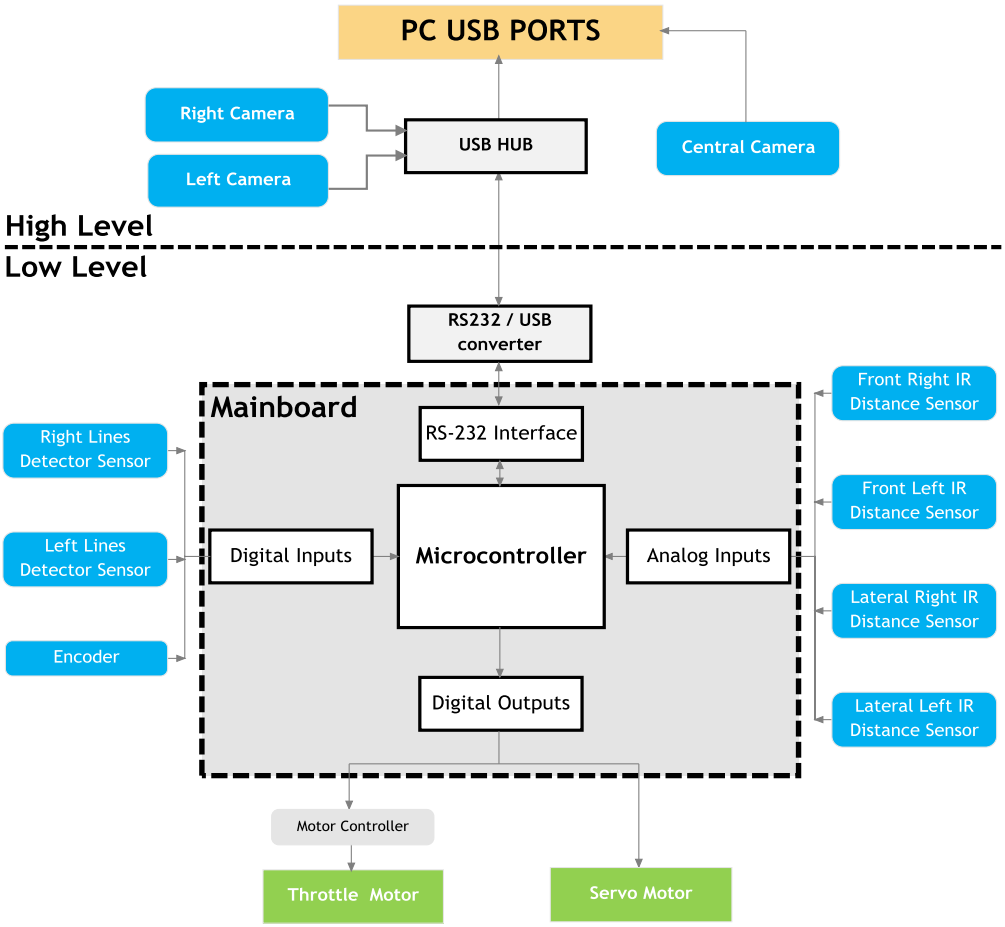


Fig. 4.20 The hardware architecture of the system.

4.2.1 - Low Level Layer

The low level layer takes care of the tasks involving the acquisition and processing of signals coming from the sensors and sends signals tailored to the needs of the actuators. This

level also implements a communication protocol that allows the data to be sent /read to and from the outside, thus interfacing with the upper layer level.

Sensors

The processing of data coming from the sensors is done differently depending on its type - digital or analog.

Regarding the processing of data coming from digital sensors (ON / OFF signals), that is done through the digital ports available in the *ATmega8* microcontroller. Concerning the sensors that detect white lines, its circuit returns analog signals which output is within the range of 0 to 5V.The fact that the microcontroller has a digital comparator in the input of each digital pin is taken in advantage. The comparator threshold is 2.5V, so the sensors are calibrated to have a voltage value below 2.5V in the presence of white lines, and over 2.5V in the opposite situation.

The ouput of the encoder is a pulsed square wave. Its resolution indicates the number of pulses sent in one complete revolution. The microcontroller counts then the edges of the signal (in this particular case the *falling edges*), to determine the number of pulses received.

As for analog sensors, these are connected to the ADC ports available, handled later by the respective firmware.

Actuators

The actuators that constitute the system are the traction motor that imposes speed onto the vehicle and servo motor, which controls the angle of the steering wheels. In the first case, the microcontroller does not act directly on the motor, but on a controller that acts as an interface between the control level and the power level of this subsystem. As for the second case, due to the specificities of the device, an embedded controller is already in place which makes it possible for the microcontroller to interact directly with it.

For the traction motor controller, it consists of a *full bridge motor driver* specifically designed for applications of this nature. The power part consists of an H bridge that operates in four quadrants (current and voltage). This power layer is interfaced with a controller, which monitories the various performance parameters of the controller, such as current, voltage and temperature, thus allowing protection against short circuits and extreme currents. Essentially, it accepts as control parameters a PWM signal and two bits that define the operating mode of the controller - Brake, Clockwise or counter clockwise. The PWM signal, sets the timing for the two voltage values (+12 V and-12V) of the signal applied to the motor terminals. The H bridges accept control signals at the bases of its two switches in complementary pairs. So if the sent PWM has a *duty cycle* of 0%, a pair of switches is always on and the other is always off, and the opposite happens in the case a *duty cycle* of 100%. Therefore to stop the motor, a PWM signal of 50% will be sent to the bridge. This driver is unique so that, once set the mode of operation, e.g. rotation in one direction, the command signal varies in a range from 0% to 100%, where 0% one finds the motor completely stopped, and 100% the total power available is delivered to the motor. Thus, the *duty cycle* indicates the percentage of power actually delivered to the engine. This aspect of the controller meant that there was no need to re-scale the signal sent by the motherboard, due to the controller

behaving with a similar configuration to the one that originally existed (note that previously the driver was only one transistor and the PWM signal was sent directly to its base).

To control the steering direction, a regular RC servo was used, which is a mechanism powered by a motor connected to a potentiometer. Once it is commanded to rotate, it turns until the potentiometer reaches the desired position. The way used to control this device is also through PWM. The pulse length indicates the desired position to the embedded electronics, which translates it into a position. It should be noted that there is no feedback as to the current position of the servo, that is, the motherboard does not have the information of where its shaft is over time.

4.2.2 - High Level Layer

The high level layer sends data coming from the various devices that make up the system to the PC, to be later analyzed and processed by the software. After that, this layer receives the control data that are sent to the motherboard the vehicle, which is responsible for sending signals to the actuators. The data exchange is thus performed in a completely transparent way. In this layer all devices have the same physical interface with the environment, a USB cable. Although the output of the board of the vehicle is done in the RS-232 protocol, it is connected to a RS232/USB converter that emulates a serial port on the PC. The converter used was the *Aten UC232A* [50].

As the PC used had only three USB ports available, a USB hub was used in order to expand the number of available ports on the PC. The devices were connected to high-speed USB 2.0 ports.

4.3 - Software Architecture

The software architecture is depicted in Fig. 4.21. As shown, the software level is divided into two layers: decision and control, which interact with the environment - the track - through the vehicle.

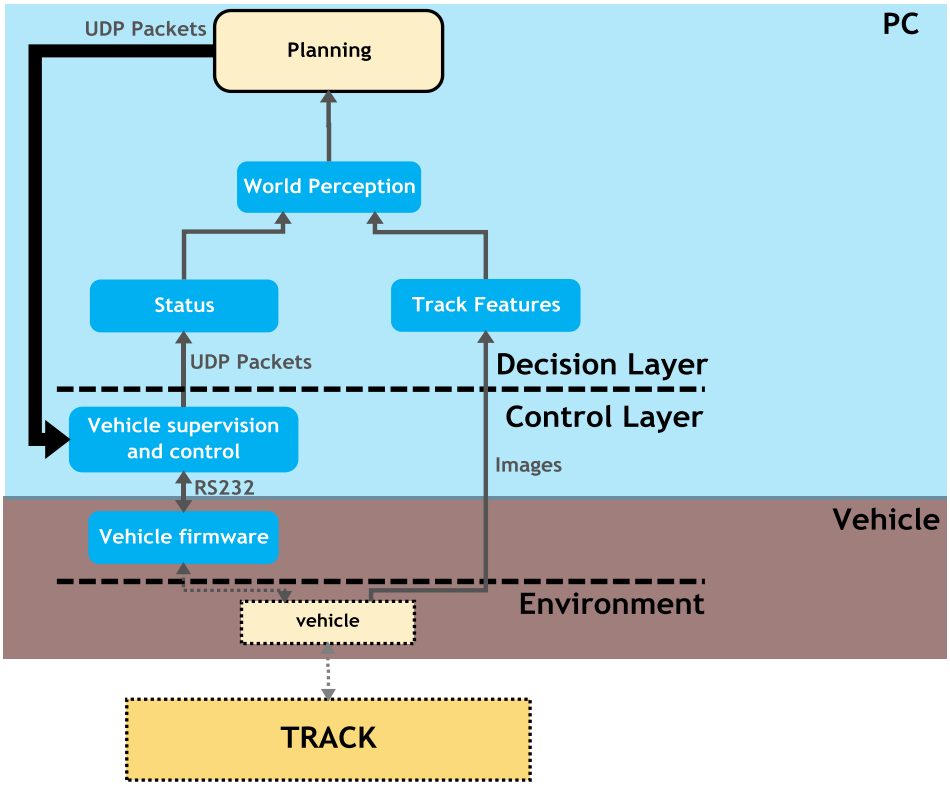


Fig. 4.21 The software architecture of the system.

The decision layer contemplates the high level control of the vehicle, which implements the algorithms that provide autonomous driving to the vehicle. The heaviest computational part of this layer consists in the vision system that returns the track features needed to perceive the world. Additionally, the odometry and the sensors provided by the control layer are used either as a backup resource (for example the detector of white lines), or along with the vision system to provide efficient world perception. The world perception block performs essentially filtering to the outputs of the vision system, preceded by metrical analysis of the vectors returned by the vision system. An approximate location of the vehicle on the track is then carried out. Once determined the present state, a planning of the operations to take for the current situation of the vehicle is performed. This block outputs then two parameters: the speed (including whether it is positive or negative), and the steering wheels angle. In order to communicate to the actuators the wanted control parameters, the information is sent to the application that implements the "Vehicle Supervision and Control ". This is done via UDP packets.

The UDP protocol is a simple connectionless transmission protocol without implicit *hand-shaking* dialogues to provide reliability, ordering, or data integrity [51]. In spite of these disadvantages, this protocol was chosen by its simplicity of operation and implementation. Also, it is assumed that both applications run on the same PC, i.e. the *host* and *client* are on the same machine, which reduces the likelihood of lost frames, the information received is incorrect, or the order of arrival is swapped. Once configured and started the exchange of data between two applications, it is important to ensure that the main task of the application that monitories and controls the vehicle is not compromised. This task is obviously communicating with the motherboard of the vehicle, which has great timing constraints. Another advantage of this configuration is the ability to run the decision application outside

the computer placed in-vehicle, through a wireless network, making it easier for debugging purposes. The algorithms were made robustly so that in case any packet gets lost, the global task is not compromised, as long as this event occurs with a reduced frequency.

Apart from time lost in sending data, mathematical operations are involved, as well as updating the values in the GUI. When adapting the application to include this new feature, this aspect was taken into consideration. Thus, a multi-threading architecture was used. At each processing cycle of the global system, a request is sent by the decision layer to update vehicle data (represented by the "Status" block in Fig. 4.21), or if necessary to change the actual set points (speed and direction). This request generates an event in the control application. So that the management of this event does not clash with the actions that were previously in progress, such as managing the events of the serial port, a new thread that runs *concurrently* with the main one is thrown, handling it. Note that truly concurrency is only possible in multi-core processors, otherwise it is done through time-division multiplexing. Once the data is sent out of the control layer to the decision layer, the thread is destroyed. An example of a chain of events is shown Fig. 4.22. In the decision application, the UDP communications are handled through a class, *CUDP*, which makes the startup / configuration of the protocol, and the messages exchange.

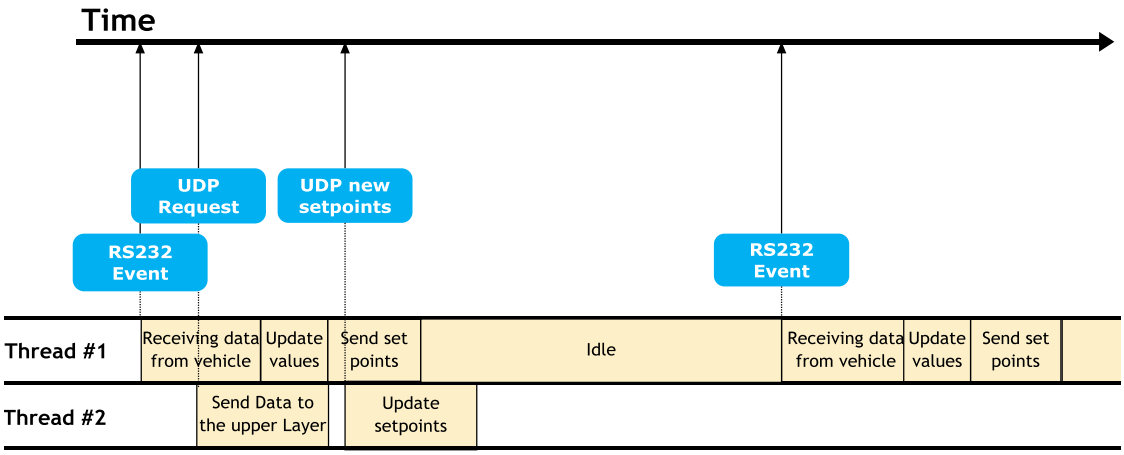


Fig. 4.22 An example of a chain of events existing in the Vehicle Supervision and Control application.

The conceptual framework found in Fig. 4.21 translates in practice into a routine that performs the system initialization (loading of the settings; initialization of cameras, communications and classes), an infinite loop where it is carried all the high-level control and once the process is over, the used memory space is released. These procedures are represented in Fig. 4.23. Note that the first three blocks of the main cycle ("Fuse Images", "Detect Track Limits" and "Find Crosswalk") will be discussed in detail in Chapter 5.

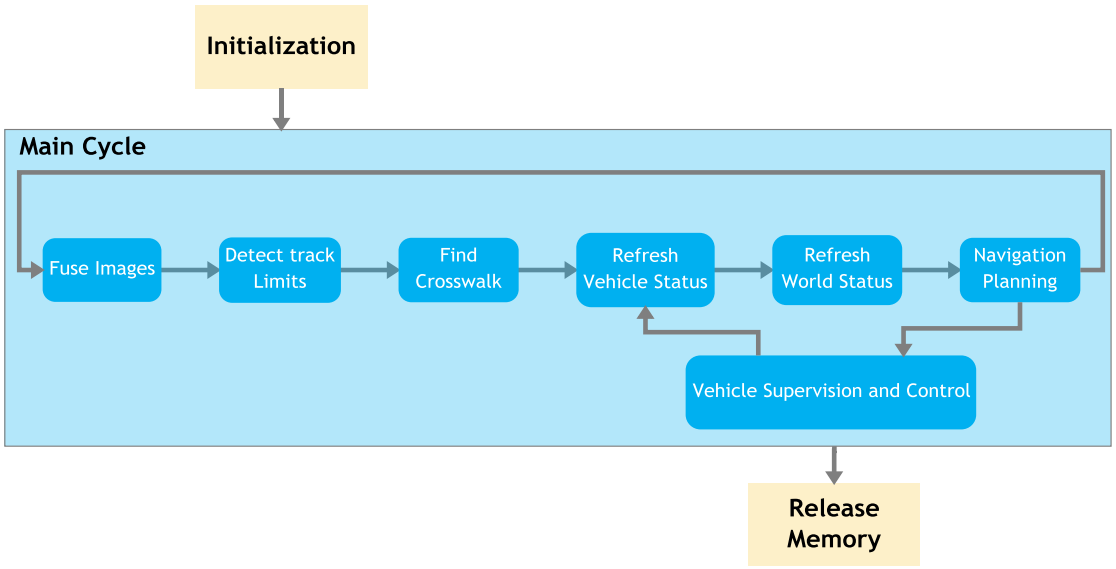


Fig. 4.23 The system main cycle.

This layer was implemented in another application developed in C++, linked with the OpenCV set of libraries. This language was used due to the fact that it is portable, object oriented, provides classes, lists and exception handling.

As for the control layer, this is done in a different application in *Lazarus* (named “Vehicle Supervision and Control” in the scheme of Fig. 4.21) and by firmware embedded in the *ATmega8* microcontroller which communicates with it via RS232. For more details about this layer, refer to [1].

4.4 - Chapter Conclusions

This chapter starts by introducing and detailing the initial vehicle platform. Some of the modifications that aim to fill some flaws are presented, also allowing the integration of this new system. Based on the modified platform, hardware and software architectures are presented, targeting the integration of all subsystems. In the following chapters, the decision layer subsystems are detailed, namely the vision system integrated in the "Track Features" block, and the world perception system based on vision and data coming from vehicle sensors. Finally the planning subsystem is detailed. The data transport method between the control and decision layer is also described below.

Chapter 5

Computer Vision System

5.1 - Image formation and acquisition

The first step of the vision system involves acquiring images from the cameras to the PC, to be further read by the image analysis application.

The process of capturing an optical image converts the default lighting scene on the image plane, by assigning each point a spatial direction of the plan, with determined spatial coordinates and brightness. The cameras are typically made out of lenses, an image sensor, and some support electronics. The sensors have the role of measuring the light pattern on the image plane in different spectral bands. In the case of color cameras, sensors with appropriate spectral sensitivities are used. The lenses capture the light, concentrating it in a single point to perform focusing. As for electronics, they perform the acquisition and processing of data returned from the sensor, sending it to a capturing device (for instance a PC).

5.1.1 - Pinhole Camera Model

The pinhole camera is the simplest and the ideal, model of a camera. In this model, light is envisioned as entering from the scene or a distant object to the camera, and only a single ray enters from any particular point. This point is then projected onto an imaging surface.

It consists of an infinitesimally small hole through which light enters, before forming an inverted image on the camera surface facing the hole. As a result, the image is always in focus, and the relation of its real size with respect to its size in the projected plane is affected by a single parameter of the camera, the *focal length*. To bypass the inversion effect, the image plane in the model is placed between the focal length of the camera and the object, so that the image is not inverted. To map a 3D point in the real world onto a two dimensional point in the image plane, perspective projection is performed. This arrangement is the one shown in Fig. 5.1, where a point Q in the world coordinates is projected onto the image plane by the ray passing through the center of projection and the resulting point on the image plane is q_1.

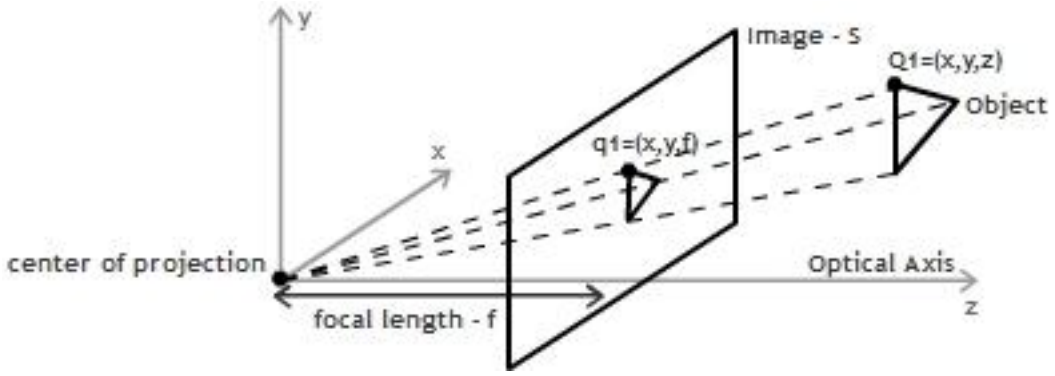


Fig. 5.1 Adapted pinhole camera model.

As seen in Fig. 5.1, the similar triangles relationship can be used to derive the relationship between points Q_1 and q_1, as shown in equations 5.1 and 5.2.

$$\frac{x}{f} = \frac{X}{Z} \quad 5.1$$

$$\frac{y}{f} = \frac{Y}{Z} \quad 5.2$$

In fact, the center of the image sensor is usually not in the optical axis of the camera, which leads to the introduction of two new parameters, c_x and c_y, to model that displacement. There are also two different focal lengths for each coordinate. This is due to the fact that the pixels in a typical camera are rectangular shaped rather than squared. This yields to a reformulation of equations 5.1 and 5.2, as shown in equations 5.3 and 5.4

$$x = f_x \frac{X}{Z} + c_x \quad 5.3$$

$$y = f_y \frac{Y}{Z} + c_y \quad 5.4$$

In homogeneous coordinates, the perspective projection onto the plane is summarized by equation 5.5. Note that the resultant x and y values should be divided by w, as w is equal to Z.

$$q = MQ, \text{ where } q = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \quad 5.5$$

5.1.2 - Lens

Little light passes through the pinhole, so it is necessary to wait until enough light is accumulated on the imager to get a satisfactory image, making this an impracticable configuration. So that this capture runs faster, more light is needed. This is overcome through a larger capturing area that “bends” the light so that it converges in the center of projection. This process is called focusing. To accomplish this, lenses are used to focus a great amount of light in one point, thus increasing the rate of image acquisition. However the lenses introduce distortion in the captured imager. The two distortion effects are: radial distortions and tangential distortions.

The radial distortion comes from the fact that the lenses distort the pixels near the edges of the imager. This phenomenon is the origin of the well-known *barrel effect*. This effect is illustrated in Fig. 5.2.

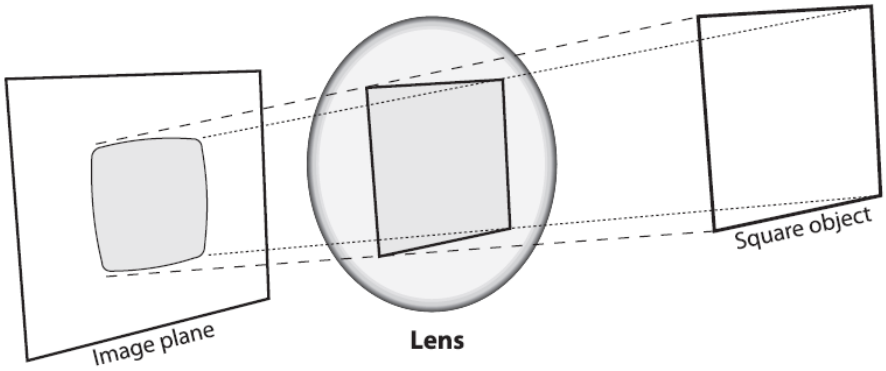


Fig. 5.2 Radial distortion effect [36]

Distortion is zero at the optical center of the image, and increasingly higher as we move to the periphery. Mathematically, this distortion can be characterized by expanding the Taylor series around r = 0 (where r is refers to the distance of the optical center of the imager). For lenses that impose a relatively low distortion, the first two terms are used, while for lenses with larger distortions (e.g. *fish-eye* lens) a third term may be necessary. The expressions used for the correction of the coordinates are shown in equations 5.6 and 5.7, where *x* and *y* correspond to the original location of the distorted points.

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad 5.6$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad 5.7$$

The other main effect of distortions introduced by lenses is the tangential distortion. This distortion comes as a result from the manufacturing process, which does not place the lenses parallel to the image sensor. This arrangement is illustrated in Fig. 5.3, and is characterized mathematically by equations 5.8 and 5.9, where p_1 and p_2 denote the tangential distortion parameters.

$$x_{corrected} = x + [2p_1 y + p_2(r^2 + 2x^2)] \quad 5.8$$

$$y_{corrected} = y + [p_1(r^2 + 2y) + 2p_2 x] \quad 5.9$$

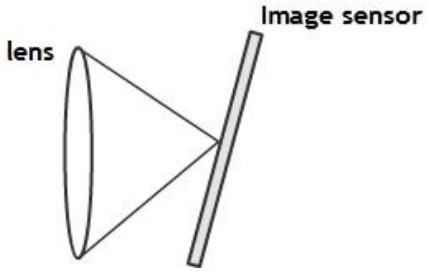


Fig. 5.3 Non-perpendicular position of the image sensor facing to the lens

5.1.3 - Calibration

Calibration is the process of correcting the distorted image, through mathematical operations. This need comes from the fact that although the camera can in fact capture the whole scene, it is not straightforward to identify certain aspects of an object and accurately localize an object on a given field, due to these distortions. Because of the radial distortion, a squared object may be mistaken for a circular one, while the displacement of image points due to tangential distortion, can affect the location perception of a certain object. In an imaging fusion stage, camera calibration should be performed previously, so that a correct fusion is achieved.

As seen previously, there are a total of 5 coefficients of distortion in a camera. These parameters, k_1, k_2, p_1, p_2, k_3 are denominated by distortion vector. Note that this specific order for the parameters is imposed by *OpenCV*. *OpenCV* provides a set of algorithms for automatic determination of these parameters, and therefore correction of the image. In order to do so, OpenCV requires a picture with defined patterns or identifiable points, such as a chessboard. Given various images of a chessboard in different positions and orientations, enough information is gathered to completely solve for the locations of those images in global coordinates and camera *intrinsics*. These parameters encompass focal length, image format, and principal point. Conveniently, OpenCV also provides functions to find the chessboard corners, and to draw the corners found, for debugging purposes. An image of that process is shown in Fig. 5.4.

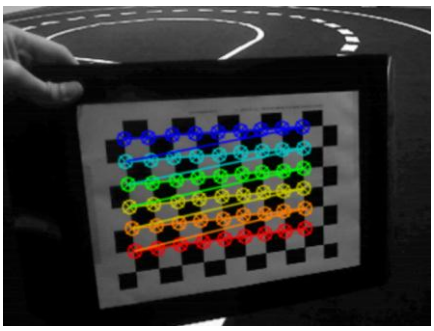


Fig. 5.4 The calibration process using a chessboard pattern to determine the distortion parameters

After a certain number of successful set of points found in the captured images, the distortion parameters are determined, and stored in a file. This file can be later accessed so that the captured images from the calibrated camera can be corrected. Consequently the calibration procedure is need to be performed only once.

5.1.4 - Acquisition of images

The sensor in the camera used is a Complementary metal-oxide-semiconductor (CMOS). When light reaches the chip, a small electrical discharge occurs in this sensor. This voltage is then converted by the embedded electronics into digital information, namely through ADCs. This conversion is done one pixel at a time as they are read by the chip. The electrical signal returned by the sensor is a one-dimensional electrical signal, which ensures a direct correspondence between the time coordinate and the spatial coordinates of the image. It is also used a color sensor, which separates the light rays in the three essential components: red, green and blue.

An image is a two-dimensional function ***f(x,y)*** (in Cartesian coordinates) of light intensity incident on the image plane. Since light is a form of energy, light intensity values verify the condition 5.10.

$$0 \leq f(x,y) \leq \infty \tag{5.10}$$

The considered function is continuous in continuous coordinates. In order to get a digital image, it is necessary to perform a spatial sampling, quantifying the brightness levels of each spectral band. The spatial sampling is generally done using a square grid. Thus, it is generated a continuous function in discrete coordinates – ***i*** (horizontal/columns) and ***j*** (vertical/lines) - which are limited to the range of variation, as shown in equations 5.11.

$$\begin{aligned} 0 \leq i \leq M-1 \\ 0 \leq j \leq N-1 \end{aligned} \tag{5.11}$$

Usually M and N are powers of base 2, and the point (0,0) is matched to the upper left corner of the sampled area. Thus, the sampled image can be represented as a matrix, as shown in 5.12.

$$\begin{bmatrix} f(0,0) & f(1,0) & \cdots & \cdots & f(M-1,0) \\ f(0,1) & \cdots & \cdots & \cdots & f(M-1,1) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ f(0,N-1) & f(1,N-1) & \cdots & \cdots & f(M-1,N-1) \end{bmatrix} \tag{5.12}$$

The quantification of brightness levels of each spectral band is generally done by subdividing the range in an uniform sampling in $K = 2^b$ levels, being assigned the code 0 to black to *K-1* to white. The values used for *M, N* (also known as the resolution of the image) and b, vary with the type of application, being the typical values 32 to 2048 for *M* and *N*, and 1 to 12 bits for *b*.

5.2 - Image Fusion

While performing the complex task of handling a vehicle, the human being executes continuous movements of the various elements that compose him, primarily through the movement of eyes and head. Thus, he has a much wider perception of the environment compared to what he would have if the receptors collecting the environment information were static. That way, the task of perceiving the environment is much simplified. In this concrete application, most of the sensory information provided to the navigation system is held through vision, which tries to recreate the visual perception achieved by humans. The

movement held by the cameras throughout its course is imposed by the motion of the robot, having no additional movement whatsoever. For this reason, it is important to obtain a capturing area as wider as possible. This is accomplished by the pre-stage of image fusion that, as the name implies, fuses both images into one single image to be later processed and analyzed.

Image Fusion is defined as the process of combining substantial information from several cameras using mathematical techniques, in order to create a single composite image that will be more comprehensive and thus, more useful for the computer vision tasks. The automated procedure of conveying all the meaningful information from the input sensors to a final composite image is the goal of a fusion system [52]. A diagram of an image fusion system is illustrated in Fig. 5.5.

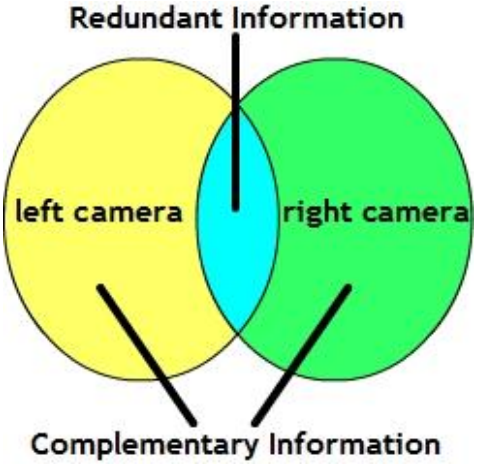


Fig. 5.5 Image Fusion diagram

Such a configuration was implemented in the system. The camera arrangement was made so that the information provided from each camera is complementary, i.e., providing two images of the left and right side of the vehicle.

It was also taken into consideration that the acquisition of both images is made at the same instant in time, so that the *merging* process of both images corresponds to the same real-world scenario. For this purpose, OpenCV implements a method that tries to synchronize several cameras. Although not ensuring that the images are grabbed at the exact time instant (for this purpose an external hardware triggering should be used), its gap is not considerable large in the context of this work

In Fig. 5.6 the green region shows the area captured by the right camera, while the yellow area depicts the image capture by the left camera. Both cameras capture an area of shared vision, represented by the gray color in the image. It was opted for an arrangement in which the cameras had a minimum common area in order to achieve a global viewing area as wider as possible.

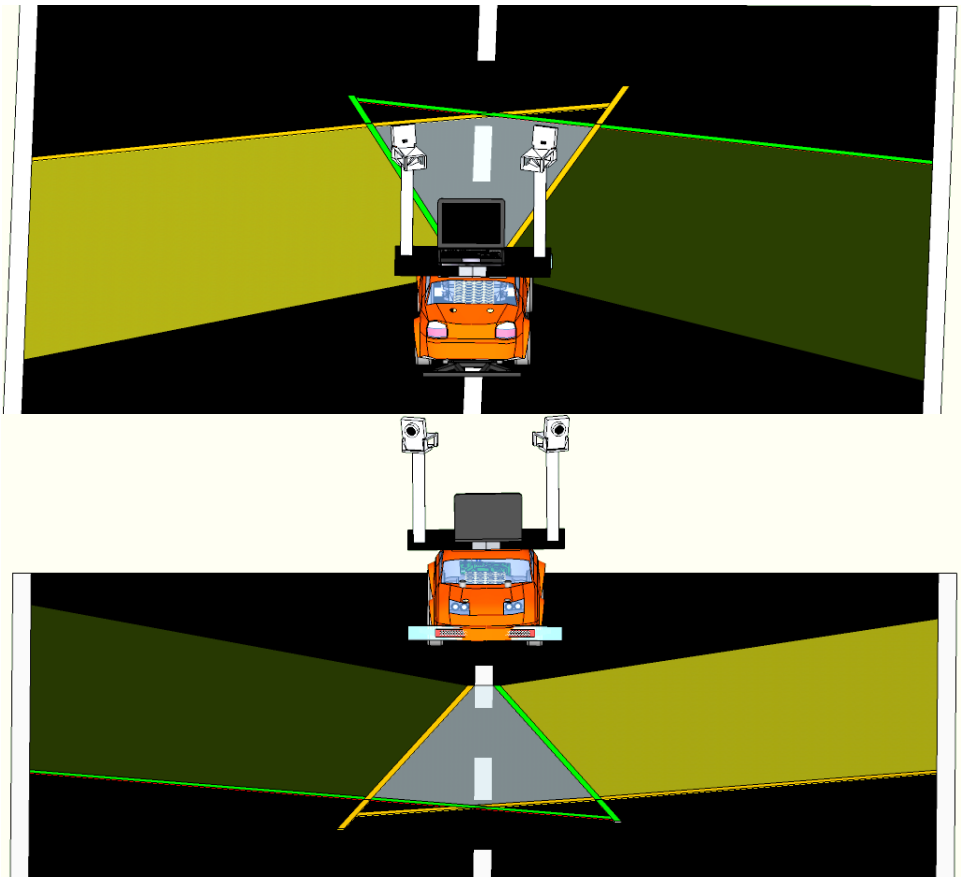
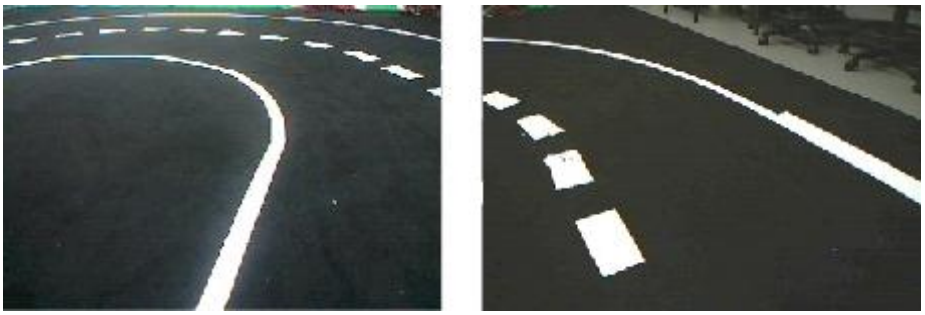


Fig. 5.6 A 3d model representing the camera arrangement used in the vision system. The top image shows a rear view, and the bottom image shows a front view

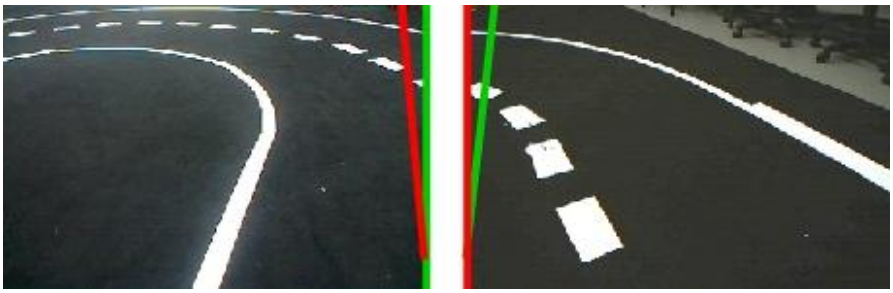
There are numerous techniques to perform this procedure. These techniques, despite using fusion methods in which the image-result contains almost no noticeable discontinuities to the human eye, come at high expense of computational cost. Therefore, it was tried to obtain a solution with minimal impact on speed of execution of the global algorithm.



a) b)

Fig. 5.7 Two captured images in the time instant. a) represents the left camera, while b) represents the right camera.

The image in Fig. 5.7 represents two real images collected at the same time instant, from the left and right camera, respectively. As can be seen, these are complementary, although there is an area of redundant information, where both capture the same scene. The determination of this common area was performed manually.



a) b)

Fig. 5.8 Two images captured from captured in the same instant in time. a) represents the left camera, while b) represents the right camera.

The procedure considered the following: the last points viewed by the left camera, i.e. a vertical line of *x*-coordinate equal to the horizontal size of the image, are displayed on the right camera as a sloped line. These lines are represented by the green line that in Fig. 4.17-a) represents the last points seen by the left camera, and in Fig. 4.17-b), its projection on the right camera. An analogous effect applies to right camera, represented by the red line in Fig. 5.8. The slope of the projected line is related to: the placement of the cameras with respect to each other, the angle of the cameras to the ground and the lens aperture (which is 72° for the cameras used), as shown in Fig. 5.9.

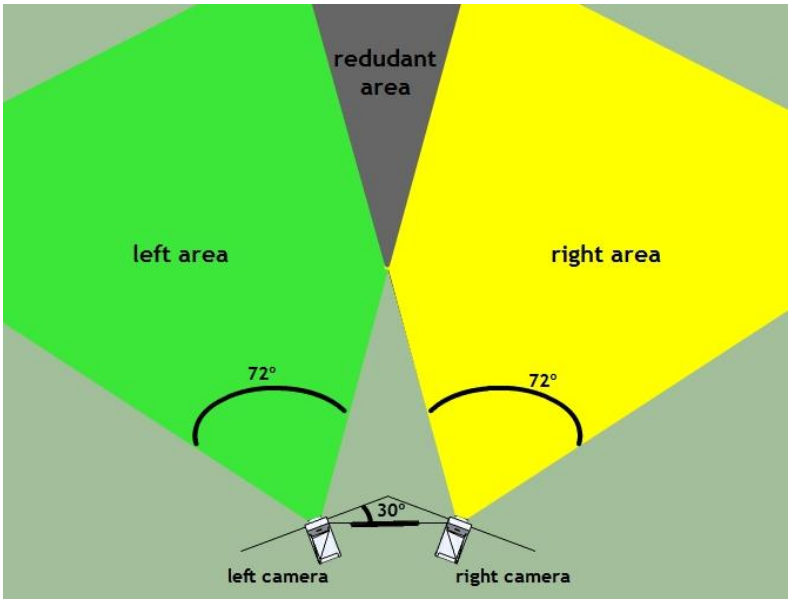


Fig. 5.9 The camera arrangement used in the image fusion task

Thus, a line was drawn on the floor that defines the set of points last viewed by their respective camera. This configuration was performed following the setup shown in Fig. 4.17.

Then, two points of each of the projected lines were determined, hence defining the set of points projected in the adjacent camera. With these two points it was possible to define the equation of its lines, to define in order to define the regions of interest. These regions are defined as the white area represented in Fig. 5.10.

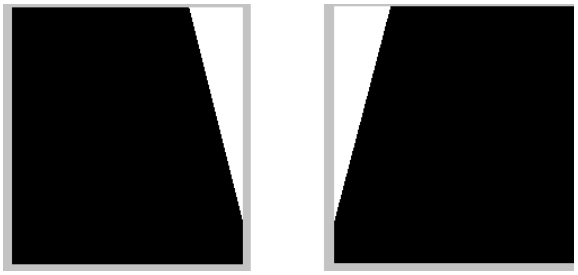
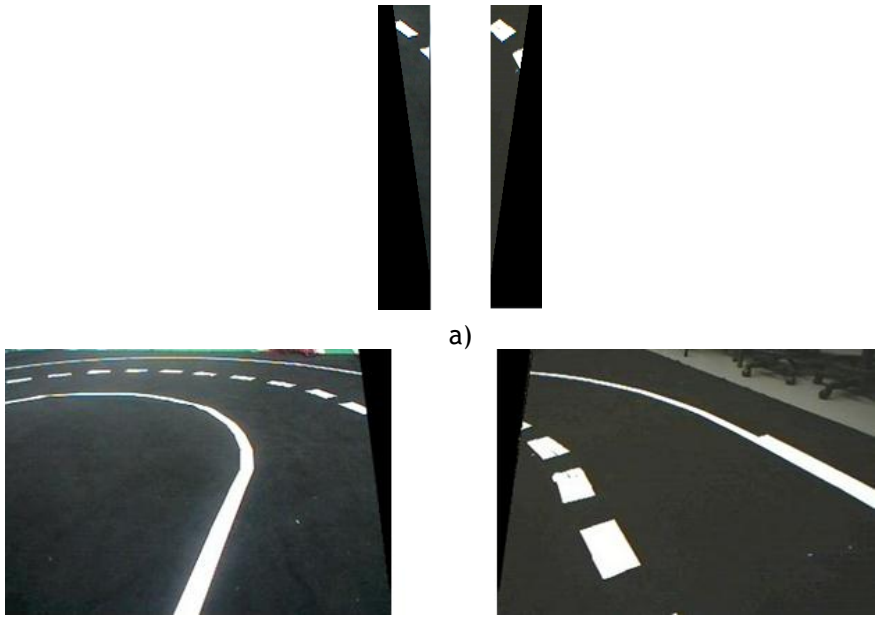


Fig. 5.10 The defined *masks*, that define the redundant region

Such regions are called by OpenCV as *masks*, which have the property of working along with binary operators, where any null pixel in the gray scale represents a 0 logical value, and any non-null pixel signifies a logical 1. Through an AND operator, the image area corresponding to the original image was achieved, as shown in Fig. 5.11-a).

The same was operation was performed to obtain the complementary images. By preceding a NOT operator to the mask and thereby applying an AND operator, as shown in Fig. 5.11-b) and Fig. 5.11-c).



a)

b) c)

Fig. 5.11 The separation between the various components of the image in the process of image fusion. a) shows the redundant area. b) shows the left complementary region and c) shows the right complementary region.

For computational simplicity, it was chosen to use the average method of the two redundant parts, to obtain the remaining part of the image. To this end, two affine transformation operations were performed.

Affine transformation is a geometric manipulation of images that makes use of a 2-by-3 matrix to map a given set of points in a plane, to any other set of points in the same plane. Such transformation is shown in Equation 5.13.

$$[X'] = [T] \cdot [X] \qquad 5.13$$

where:

$$[X'] = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} a_{00} & a_{01} & b_0 \\ a_{10} & a_{11} & b_1 \end{bmatrix}, [X] = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} .$$

OpenCV computes this transformation matrix automatically, through a list of three points on the original image and its three points in the resulting image.

The first performed transformation aims to “overlap” the two redundant regions, to subsequently calculate the mean of the two images. The image chosen to be transformed was the left one, to fit the dimensions of the redundant right image. This choice did not follow any criteria, as its choice is indifferent. Such transformation is show in Fig. 5.12.

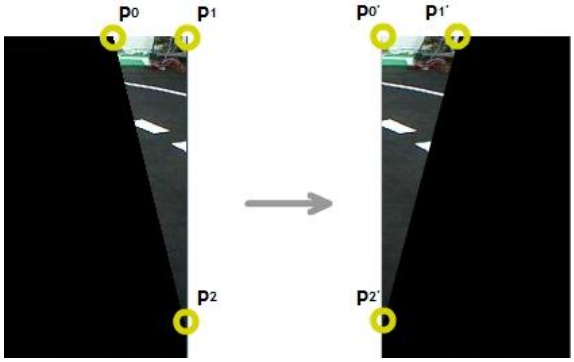


Fig. 5.12 The right image shows the resultant affine operation to the input image shown in left

The second operation warps the three points of the common part to finally be fit in the complementary area. The result is illustrated in Fig. 5.13. Note that in this picture the mean is already computed, and the image dimensions are those of the final image.

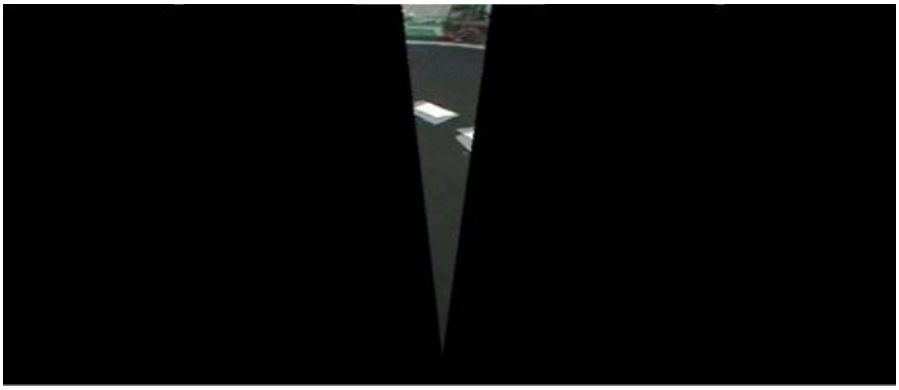


Fig. 5.13 An example of a redundant region resultant from the combination of two cameras

Finally the three parts of the image are composed into a single one. Note that the size of the resultant image is the same vertically and doubled horizontally. The whole process is summarized in the block diagram shown in Fig. 5.14.

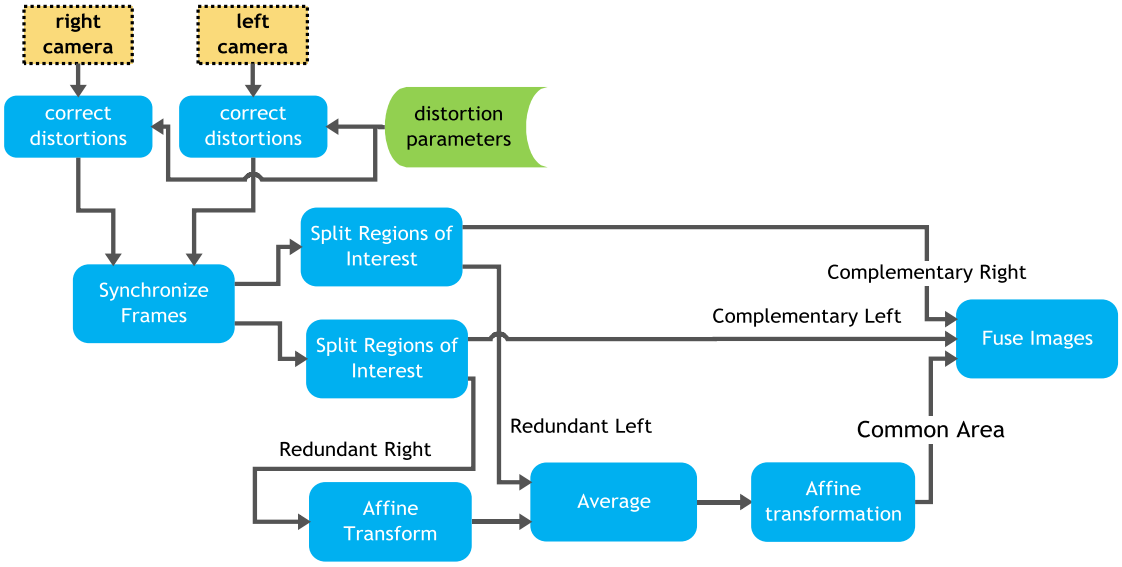


Fig. 5.14 Block Diagram of the Image Fusion Sub-system

5.3 - Extraction of Track Limits

The procedure of detecting the track limits is the most important part of the vision system. It indicates to the navigation system the position of the car in the track, so it can act accordingly. The tasks carried out in this section are the detection of the lateral lines (which are actually the road limits), the detection of the central line, and finally, the computation of the respective measures.

Due to the effect of perspective, these lines "converge" on the horizon, but for the purpose of vehicle navigation, the only relevant part of the lines is their behavior in the closest points to the vehicle. Therefore it is intended to determine the tangent at the point closest to the vehicle, as shown in equation 5.14 for the left line, and equation 5.15 for the right line. These equations are the mathematical formulation of a line tangent to curves $l(x)$ and $r(x)$, representing the lateral limits. x_{min} denotes the minimum x coordinate of the respective curve and the analogously, the same goes for $x_{max}$and y_{max}. Note that the coordinate reference system is located at the upper left point. In Fig. 5.15 a visual representation of such lines is shown.

$$L(x) = \frac{\partial l(x_{min})}{\partial x} x + y_{max} - \frac{\partial l}{\partial x}(x_{min}) x_{min} \quad 5.14$$

$$R(x) = \frac{\partial r(x_{max})}{\partial x} x + y_{max} - \frac{\partial r}{\partial x}(x_{max}) x_{max} \quad 5.15$$

Fig. 5.15 A visual representation of the interesting lines L(x) and R(x).

The lateral lines are physically always parallel and its projection on the imager filters the area of interest which is the inner region formed by these two lines. These lines have also an important property that is to maintain connectivity between all the pixels that define them.

The algorithm starts by performing a preprocessing task to the image provided from the sub-system of image fusion. Then, the left and right markings are separated into two different images, to determinate a vector that defines them. Based on this vector and on the image provided from the pre-processing stage, it is possible to create a distinctive image for the central region, where the central markings are contained. After this set of operations, a vector that defines the orientation and position of these markings is also computed. These steps are summarized in the diagram of Fig. 5.16.

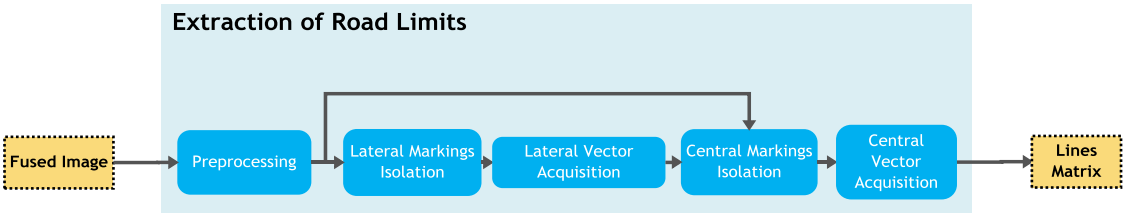


Fig. 5.16 Block diagram of the tasks followed in the extraction of road limits stage

5.3.1 - Pre-processing

This initial task prepares the raw image returned from image fusion block to be examined later. Pre-processing constitutes the earliest phase that ultimately returns a binary image, with well-defined areas of interest. The pre-processing stage has a great responsibility in the final result of the vision system, since a poorly performed preprocessing invalidates the entire system. Basically, the pre-processing stage provides consistent information to the other stages, indicating which parts of the received image should be analyzed.

For these reasons, it is important to define what areas of interest should be considered, and therefore exclude other parts of the image. The track consists of a standard black pattern, where it is superimposed white markings. These markings indicate the limits of the track, i.e. the continuous boundaries where the vehicle should be located; a crosswalk made of the same material; and auxiliary markings, which define separate the lanes of the track. Thus the task of pre-processing should return a binary image containing solely these marks, preserving its continuity where it applies. It should be robust to variations in brightness and color. Another important aspect is that the interesting area to be analyzed is the one nearest to the vehicle, as it is intended to obtain the lines tangent to the bottom of the image. Apart from that, this approach also reduces the execution time of the algorithm. For that purpose,

the algorithm defines the bottom half as the area of interest to analyze, ignoring the rest of the image.

The algorithm starts by converting the original image to a gray one. Then, smoothing is conducted by the method of simple blurring, with a 3-by-3 window, since a more uniform distribution of pixels is required, thereby mitigating large variations of brightness. The simple blurring method was chosen because from the computational standpoint it is clearly faster than other methods. Also, the markings are regions with natural high contrast relative to the background, so the edges are naturally well defined. Therefore, it is intended above all, that the markings are in a relatively equal distribution of brightness.

Then, the image is submitted to a threshold, which results in a binary image. This is the most sensitive step of this sub-block, and requires some manual refinement to obtain the desired result. So that the threshold takes place on invariant light conditions, it was chosen an adaptive thresholding method. After several experiments with the Gaussian method and the average unit method, it was opted for the average unit, for it can better preserve the markings, introducing less noise into the final result, and is less computationally expensive.

Ideally the result of this threshold should be an image in which the background is scored by the following way: logical 0 to the background, and logical 1 to the markings. This approach is not possible in practice, because in order to ensure a null background, the markings become discontinuous along its length. To meet the continuity requisite, threshold parameters were chosen so that the markings remain as continuous as possible, even though some background noise is introduced. Next, the image is eroded in order to reduce some of this noise, followed by a close operation, which is iterated eight times, aiming to close any points of discontinuity in the markings.

It should be noted that it is necessary a great commitment between threshold and closing operations, since this set of operations must ensure that the markings are solid, but the background noise introduced by the thresholding operation does not "merge" with the markings. These steps are illustrated in Fig. 5.17.

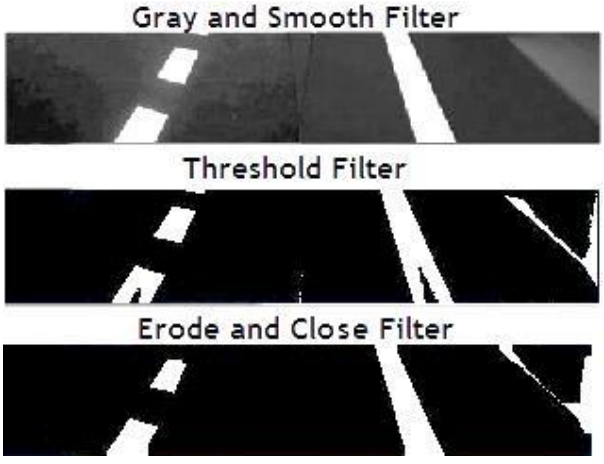


Fig. 5.17 An example of some of the thresholding applied to an image.

Finally, it is obtained an image with background noise, but with the markings isolated and continuous, and therefore well defined. This is the result actually intended, as will be seen in the following steps. The set of operations that constitute the threshold task is illustrated block diagram of the Fig. 5.18. An example of these operations is also shown in Fig. 5.17.

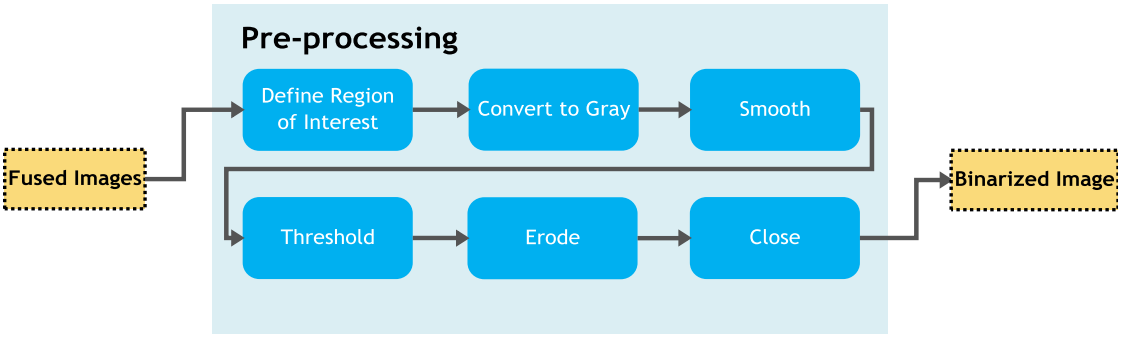


Fig. 5.18 Block Diagram of the preprocessing task

5.3.2 - Lateral Markings Isolation

The previous task returns a binary image, where the markings are well defined and their continuity is ensured as well as their separation from the rest of the image. It is sought as a result of this step, two new binary images, whose content represents only the left and right markings, respectively. For this end, it is performed an isolation by the area restricted by these markings, i.e., the area of the track. Thereafter, based on this information, it is searched for the markings, and if found, these are isolated from the rest of the image.

These steps are depicted in Fig. 5.19

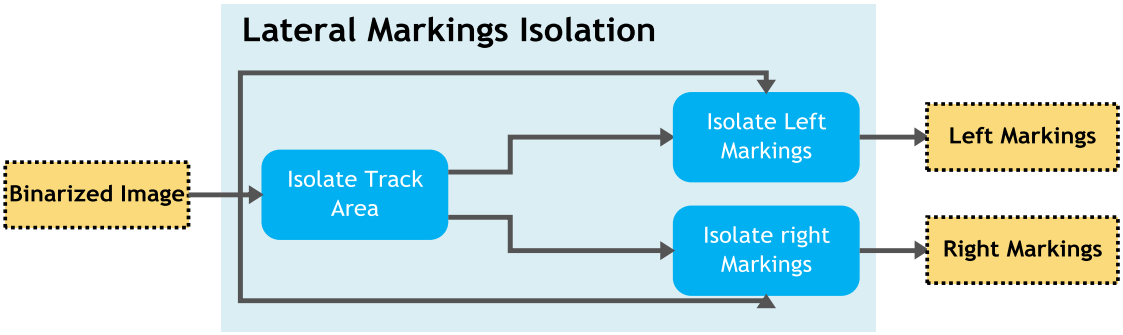


Fig. 5.19 Block Diagram of the Lateral markings isolation task

5.3.2.1 - Track Area Isolation

Taking advantage of the fact that the lateral lines are always parallel and continuous, a flood filling procedure was used in order to isolate the track area. This method (explained in 3.7 - Flood Fill) is already implemented in OpenCV. To this end, the algorithm paints the inner region defined by the right and left lines of the track, defined as the track area.

This implementation is focused in finding the correct seed point, assuming the continuity of the markings. The algorithm performs, through an iterative process, successive paintings in the original picture, until it is considered that the painting is actually the desired area. This process uses a starting seed point, and if the painted area is not correct, a new point is tried, to paint the correct area. The starting point is (width/2, height) where width is the width of

the incoming picture, and height is the height of the picture. The choice of this point is related to the fact that the track "arises" from the bottom of the image. Thus, the bottom of the image contains always the track area. After a non-successful painting, the x-coordinate of this point is incremented in a step of $\frac{(width/2)}{10}$. Once the search ends in an upward direction of the image, it searches in the opposite direction, starting one step before the middle of the image. It is considered that the painting was not successful, when the ratio between the painted area and the area of the original image is lower than a certain value. This value was experimentally found to be 40%.

Next, a procedure to isolate the painted portion from the rest of the image is performed, thus, in order to actually isolate the track area. This is done through binary operators as shown in Equation 5.16.

$$Painted\ Image \cap \overline{Original\ Image} = Track\ Area \tag{5.16}$$

The result of this operation is illustrated in Fig. 5.20, as well as a flowchart representing the tasks performed at this stage in Fig. 5.21.

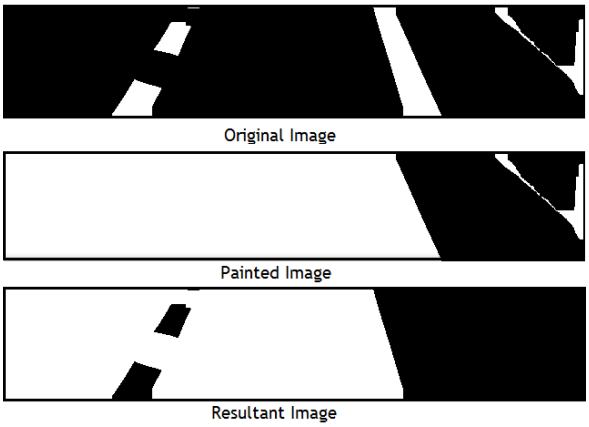


Fig. 5.20 An example of the AND operator used in the last stage of track isolation task

When the continuous central is present, the algorithm interprets that the central line is in fact a lateral boundary. However, this fact does not compromise normal measurement analysis or the navigation system. In practice the normal width of the track is transformed into half, and the (normal) central line is considered as absent.

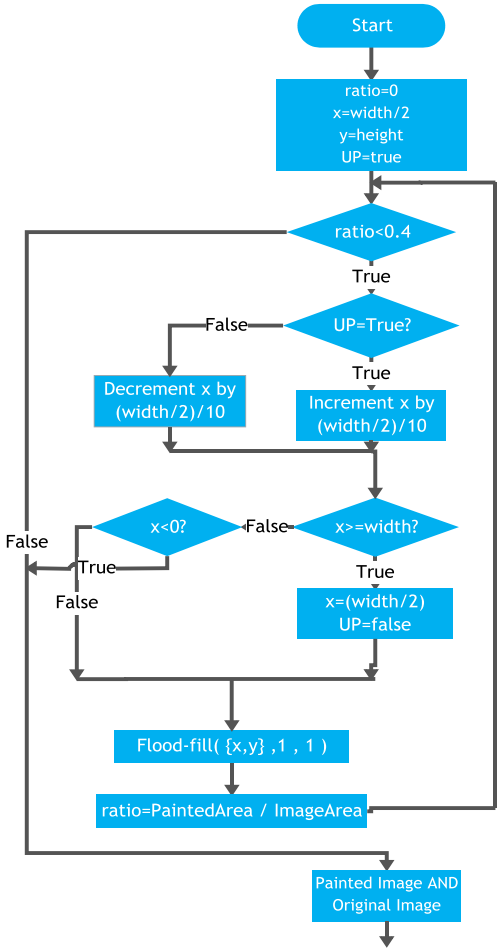


Fig. 5.21 Flowchart of the track isolation task

5.3.2.2 - Left and Right Markings Isolation

In the previous step, the resultant image covers the area of the track, excluding the markings (as shown in Fig. 5.20). In fact two images are inputted to this block: the resultant image from the previous step and the image returned from the threshold block, which will be called “original image”. As previously stated, this step aims to obtain two images containing only the two lateral boundaries. Again, taking advantage of the continuity of lines across the length of the image, the flood fill method is applied to the lines. This takes advantage of the information returned from the flood fill method implemented from OpenCV, which returns the painted region.

The algorithm is based on the previous method, since an iterative method is also carried out, by painting attempts. For both limits of the track, there are variations on the method of determining the initial seed point. In general, both methods perform a search for points on left and right borders of the region defined previously. If eventually no points are found, or if they have been found, but the painting has not been successful, it is returned a flag, which tells the navigation system that the respective line is not available.

Below, an explanation of the determination of the right seed point is presented, which is analogous to the left seed point determination.

The algorithm starts by determining if the right line exists. In order to do so, it is considered the following: as the vehicle approaches this line, the input image is less populated in terms of white pixels in the far-right region. In the opposite situation in which the vehicle is approaching the left limit, the input image is more populated in the far-right, until eventually this line is no longer visible. In this situation, the far-right region of the image will have a high density of white pixels (around 100%). Taking this into account, the algorithm starts by examining such region, to determine if the right line is visible. This region was defined as a strip corresponding to a percentage of the image width, as depicted in the yellow region of Fig. 5.22.

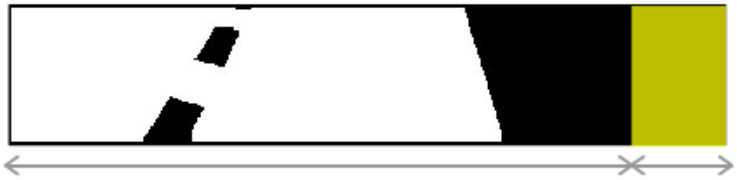


Fig. 5.22 The yellow strip is the region used to examine if the right line is visible

The examination performs the average of white pixels available in that strip, and if below a certain threshold, it is considered that the right line exists. The parameters were chosen through analysis of practice situations. The yellow strip was set to 5% of the width of the image and the maximum density of white pixels was set to 80%. Note that in the example of Fig. 5.22, the yellow region has a 0% density of white pixels, which indicates the right line is visible.

Next, the algorithm performs the search for white pixels, naming this process by “Find Right Seed”. Note that this process is continuous, i.e., the algorithm starts traveling the image matrix from right to left and bottom-up, and if the white pixel density in the yellow region is below the set threshold, the search is continued; otherwise, it stops and a flag is returned. The first white pixel found outside the region indicates that the right line is disposed in the point immediately to the right of that point in the original image, as illustrated in Fig. 5.23.

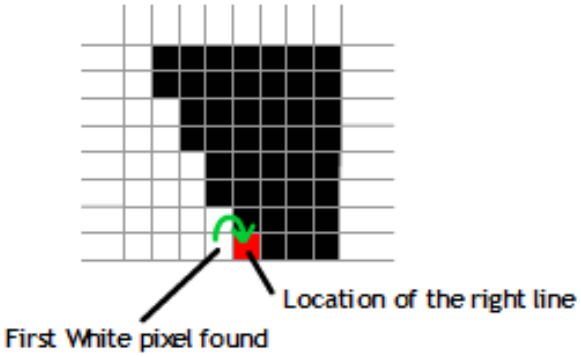


Fig. 5.23 Determination of the location of the right line based on the isolated image of the track – “Find Right Seed” process

When a white pixel is found, to avoid the detection of a false seed point, it is also checked whether the point to the right in the isolated image of the track is black, as seen in Fig. 5.23, where the point to the right of the white pixel found is black (represented in red, for illustration purposes). In Fig. 5.24 a flowchart of this process is shown.

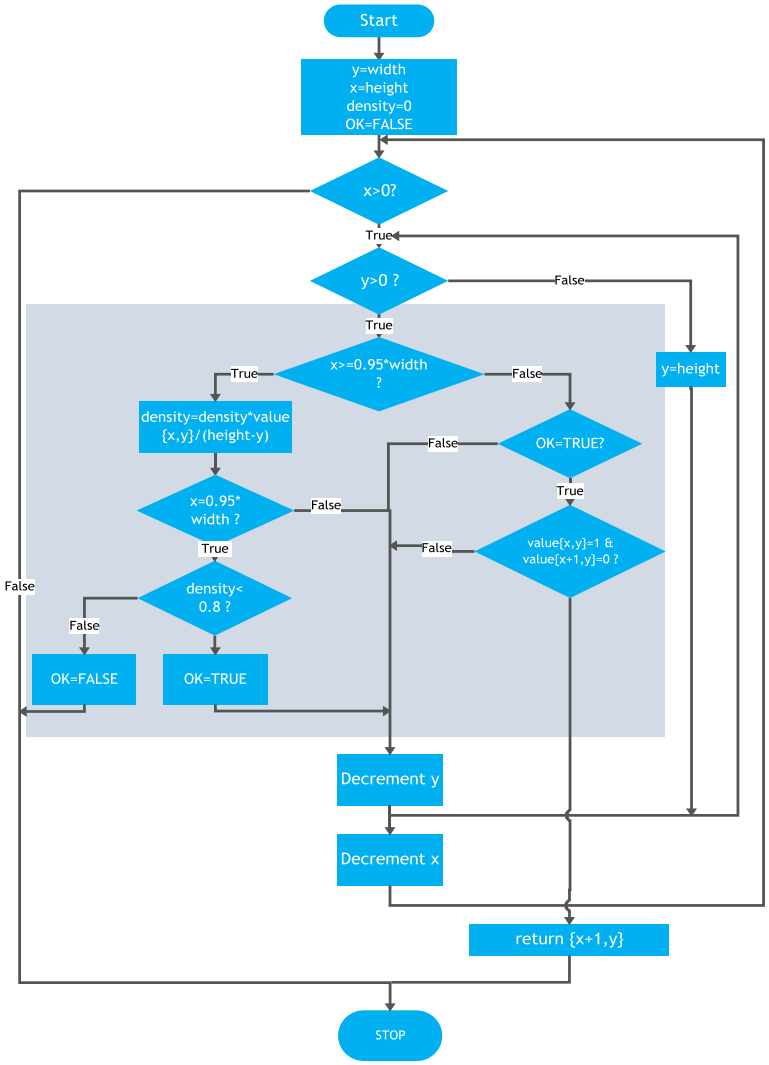


Fig. 5.24 Flowchart of the "Find Right Seed" process

Thereafter, the flood-fill method is applied to the original image (top image in Fig. 5.20), where the seed point was the one just determined. Analogously to the method of isolation of the track, it is considered that one line was found when the area of the painted region is below a certain ratio between the area of this region and the image area. This way, the algorithm starts by painting the original image, and checking the resultant ratio. If it is over the defined ratio, then another set of points in the vicinity of this one are tried. This ratio was found experimentally to be 10%. A diagram of this step in shown in Fig. 5.25.

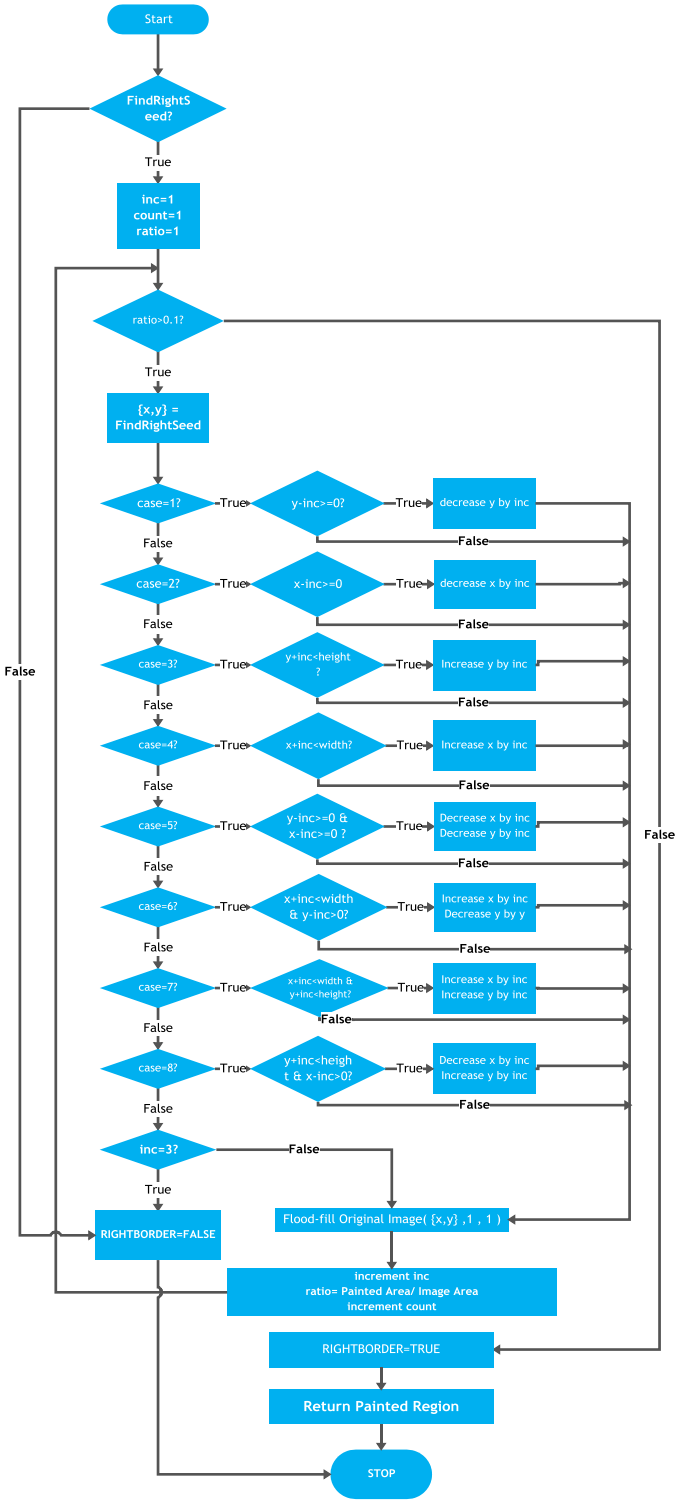


Fig. 5.25 Flowchart of the steps taken to isolate the right line markings.

Basically, if the painting in the original image was not successfully performed with the seed point obtained initially, a painting is performed with adjacent seed points in the horizontal, vertical and diagonal directions. If still not successful the search radius is increased around the original point, up to a radius of 3 pixels. Fig. 5.26 illustrates this process, where the red pixel represents the starting point, the set of blue dots are the first outer layer of attempts, and the green and yellow dots represent, the second and third attempts, respectively. If after all these attempts no solution was found, a flag RIGHTBORDER is set to false, indicating the right line is not available.

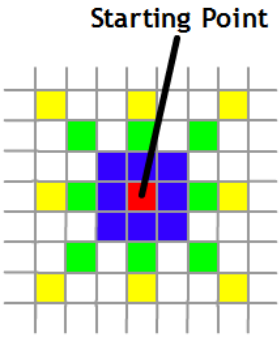


Fig. 5.26 Illustration of the process used in order to obtain the seed point used in the flood fill method to isolate the right line.

Once the correct region was painted, the flag RIGHTBORDER is set to true, so that the following tasks analyze the image returned in this step, which contains only the respective line. After the line is obtained, a canny edge detector is applied in order to obtain the edges of the markings. In Fig. 5.27 an example of these steps is shown.

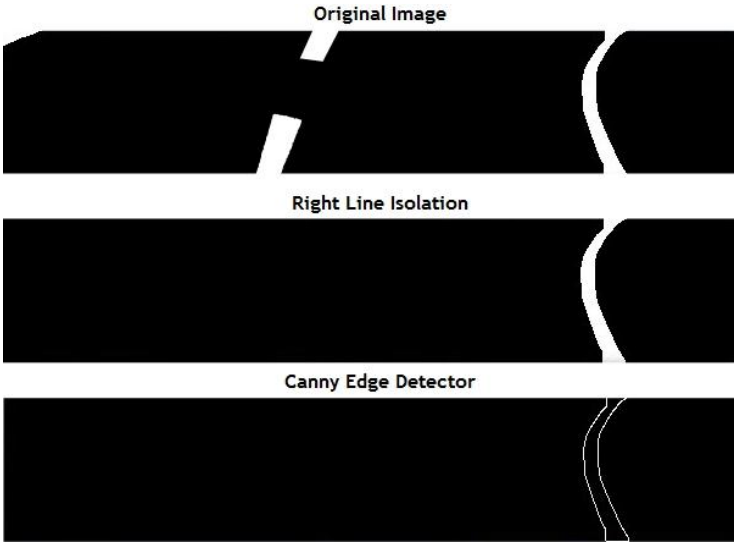


Fig. 5.27 A sample image of the steps taken to obtain an image with the isolated right line. The bottom image shows the result after applying a canny edge detector

5.3.3 - Lateral Vectors acquisition

At this stage of the sub-system, the images of the left and right markings are analyzed (when available) and a vector that represents the approximation of the lines tangent to the bottom of the image is returned. This method is based on Hough transform applied to lines, which returns all lines found, defining them by a set of two points. A block diagram showing the task performed at this stage is shown in Fig. 5.28.

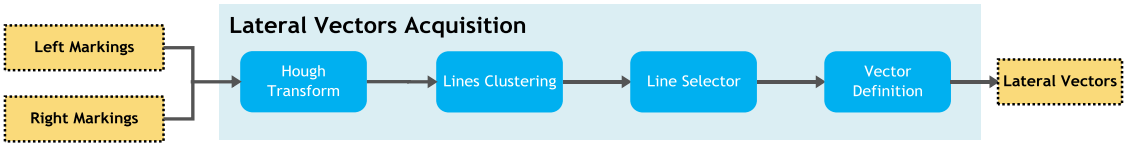


Fig. 5.28 Block Diagram of the Lateral vectors acquisition task

5.3.3.1 - Hough Transform

The result of applying the Hough transform to the input image through OpenCV is a set of lines, with a specific length. The fact that more than one line is returned lies mainly on situations where the vehicle is in the curved region of the track. However, in the straight region this effect is also present is some situations, caused by the perspective, or some remaining distortion from the camera. Fig. 5.29 shows an example situation of the output of the Hough transform applied to an input image, where the right marking is present. Note that the vehicle is in the straight region of the track.

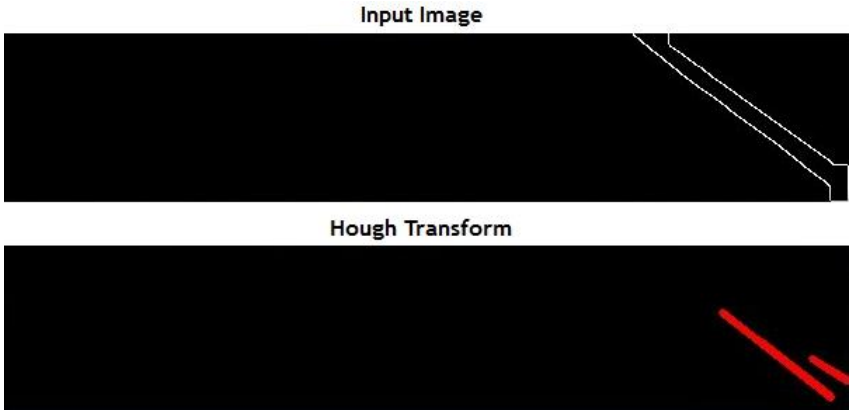


Fig. 5.29 The bottom image shows the resultant lines returned by the Hough Transform applied in the top image

The Hough transform algorithm implemented in OpenCv stores the resulting data in a dynamic stack data structure, organized in a matrix as shown in Equation 5.17.

$$Hough\ Transform(input\ image) \xrightarrow{yields} \begin{bmatrix} p_{0,0} & p_{1,0} \\ p_{0,1} & p_{1,1} \\ \vdots & \vdots \\ p_{0,N} & p_{1,N} \end{bmatrix} \quad 5.17$$

where $p_{0,n}$ and $p_{1,n}$ are the two points that define the *n*-th line found, and *N* the total number of lines found.

5.3.3.2 - Lines Clustering

Because there are several lines returned, there is a need to group similar lines. This grouping is performed according to a similarity criteria between the different obtained lines and their proximity to the bottom of the image, as it is seek for a tangent line.

The algorithm starts by determining which of the two points that define the lines are closer to the bottom of the image. Consequently the whole matrix is traversed in order to find the value denominated by max_y in equation 5.18. Remember that two points are checked due to the fact that the output of Hough Transform in openCV, does not specify an order in the presentation of points, i.e., the point p_0 is not necessarily closer to the x axis than p_1.

$$max_y = max\{p_{0,n}(y), p_{1,n}(y)\} : n \in [0, N] \quad (5.18)$$

where:

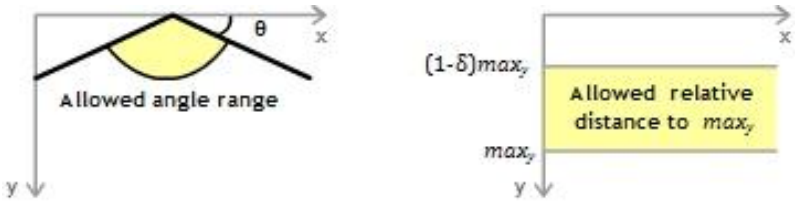
$p_{0,n}(y)$: $the\ y-coordinate\ of\ p_{0,n}$,

N : $total\ number\ of\ lines\ found$.

Then, the returned lines are filtered by its angle with respect to the ***x***-axis, and its relative distance to max_y. The angle is obtained as shown in equation 5.19. Note that it checked which of the two points is closest to the bottom of the image, to normalize to the angle within the range of [0,+90º] and [-90+,0]. The relative distance to max_y is verified through a δ parameter, that defines the relative distance allowed. As for the angle, only lines with φ_n greater than a parameter θ are considered, as well as φ_n lower than $-\theta$. Therefore, lines that do not satisfy condition 5.20 are discarded. This filtering procedure can be visualized in Fig. 5.30. The θ was empirically set to 20º and δ to 0.3. Note that this manipulation in held in the image plane.

$$\varphi_n = \begin{cases} \tan^{-1}\dfrac{p_{0,n}(y) - p_{1,n}(y)}{p_{0,n}(x) - p_{1,n}(x)}, & p_{0,n}(y) > p_{1,n}(y) \\ \tan^{-1}\dfrac{p_{1,n}(y) - p_{0,n}(y)}{p_{1,n}(x) - p_{0,n}(x)}, & p_{0,n}(y) < p_{1,n}(y) \end{cases} \quad (5.19)$$

$$(\varphi_n < -\theta\ OR\ \varphi_n > \theta) \quad \text{AND} \quad (p_{0,n}(y) > (1-\delta)max_y \quad \text{OR} \quad p_{1,n}(y) > (1-\delta)max_y) \quad (5.20)$$



a) b)

Fig. 5.30 Filtering of the lines returned by Hough transform. a) shows the allowed angle range, while b) shows the allowed relative distance to $\boldsymbol{max_y}$

After this filtering, the search for similar lines is started by parameterizing the lines, so that they can be compared with respect to their m and c coefficients. For each considered

line, a vector is filled as shown in equation 5.21. As seen, $a_n(p_{0,n}, p_{1,n})$ denotes the slope of the line, when its value is not infinite ($p_{0,n}(x) \neq p_{1,n}(x)$). The same applies for $b_n(p_{0,n}, p_{1,n})$, which is equal to the c parameter of the line (note that lines are defined as $y = mx + c$). If it is infinite, the line is represented by $x = k$. Thus $a_n(p_{0,n}, p_{1,n})$ represents that k value, which is equal to the x-coordinate of one of the two points of the line. In this situation, $b_n(p_{0,n}, p_{1,n})$ is set to 0, as well as a flag indicator, F_n which is set to 1, indicating the respective vector refers to a line with infinite slope.

$$L_n = \left[a_n(p_{0,n}, p_{1,n}) \quad b_n(p_{0,n}, p_{1,n}) \quad F_n(p_{0,n}, p_{1,n})\right]^T \tag{5.21}$$

where:

$$L_n : n - th\ line\ vector,$$

$$a_n = \begin{cases} m_n, & p_{0,n}(x) \neq p_{1,n}(x) \\ p_{1,n}(x), & otherwise \end{cases},$$

$$b_n = \begin{cases} c_n, & p_{0,n}(x) \neq p_{1,n}(x) \\ 0, & otherwise \end{cases},$$

$$m_n = \begin{cases} \dfrac{p_{0,n}(y) - p_{1,n}(y)}{p_{0,n}(x) - p_{1,n}(x)}, & p_{0,n}(y) > p_{1,n}(y) \\ \dfrac{p_{1,n}(y) - p_{0,n}(y)}{p_{1,n}(x) - p_{0,n}(x)}, & p_{0,n}(y) < p_{1,n}(y) \end{cases},$$

$$c_n = p_{0,n}(y) - p_{0,n}(x) \times m_n \ ,$$

$$F_n = \begin{cases} 1, & p_{0,n}(x) = p_{1,n}(x) \\ 0, & otherwise \end{cases}.$$

Then, this vector is introduced into a function that compares the value of the current line with groups of existing lines. If this line is not considered "similar" to any of the existing line groupings, a new group is created. So that the organization of the lines grouping is dynamic, the respective parameters are stored in a stack. In order to provide information for the line selector task that follows ahead, the groups are organized analogously to the vector in equation 5.21, with the inclusion of a new parameter which is the number of similar lines added to a group. The output of this function is a matrix represented as the H matrix in equation 5.22. Each row of this matrix refers to the line parameters of a certain group. The lines are grouped firstly according to its type, which is returned by F_n. Then if $a_n(p_{0,n}, p_{1,n})\ and\ b_n(p_{0,n}, p_{1,n})$ have similar values with respect to the grouping parameters, this line is appended to that group, where the arithmetic mean is computed to set the new grouping parameters. Thus, $\overline{a_{\{X-k\}}}$ denotes the mean of a_n values for the $\{X-k\}$ group, as well as $\overline{b_{\{X-k\}}}$ represents the mean of b_n values for the $\{X-k\}$ group.

$$H = \begin{bmatrix} \overline{a_{\{X\}}} & \overline{b_{\{X\}}} & ctr_{\{X\}} & F_{\{X\}} \\ \overline{a_{\{X-1\}}} & \overline{b_{\{X-1\}}} & ctr_{\{X-1\}} & F_{\{X-1\}} \\ \vdots & \vdots & \vdots & \vdots \\ \overline{a_{\{X-M\}}} & \overline{b_{\{X-M\}}} & ctr_{\{X-M\}} & F_{\{X-M\}} \end{bmatrix} \tag{5.22}$$

where:

$$\{X\}, \{X-1\}, \{X-2\}, \cdots, \{X-M\} \ : \ line\ groupings \ ,$$

$$\overline{a_{\{X-k\}}} = \frac{1}{ctr_{\{X-k\}}} \sum_{ctr_{\{X-k\}}} a_n \,, \qquad if\, a_n, b_n \, and F_n \; satisfy \, \{X-k\} \;,$$

$$\overline{b_{\{X-k\}}} = \frac{1}{ctr_{\{X-k\}}} \sum_{ctr_{\{X-k\}}} b_n \,, \qquad if \; a_n, b_n \; and \; F_n \; \; satisfy \, \{X-k\} \;,$$

$$ctr_{\{X-k\}} : number \; of \; lines \; appended \; to \; \{X-k\} \;,$$

$$F_{\{X-k\}} : grouping \; type \;.$$

The conditions for a specific line to be added to an existing group are specified in equation 5.23. The γ factor represents the relative disparity between the group parameter and the respective parameter of the line being analysed. It dictates thus, the maximum allowed disparity between the line being analyzed and the groups, for the line to be appended. This factor was found experimentally to be 0.3.

$$L_n \in \{X-k\} \Leftrightarrow \begin{cases} \left|\frac{\overline{a_{\{X-N\}}} - a_n}{a_n}\right| < \gamma \wedge \left|\frac{\overline{b_{\{X-N\}}} - b_n}{b_n}\right| < \gamma, & F_n = F_{\{X-k\}} = 0 \\ \left|\frac{\overline{a_{\{X-N\}}} - a_n}{a_n}\right| < \gamma \,, & F_n = F_{\{X-k\}} = 1 \end{cases} \qquad 5.23$$

where:

$$n \in [0, N] \;,$$

$$k \in [0, M] \;,$$

$$N : \; total \; number \; of \; existing \; lines \;,$$

$$M : total \; number \; of \; existing \; groups \;.$$

When conditions displayed in equation 5.23 are verified, the current line L_n is appended to the respective group as shown in equations 5.24, 5.25 and 5.26, where $\overline{a_{\{X-k\}}}_{corrected}$ denotes the new value of $\overline{a_{\{X-k\}}}$, as well as $\overline{b_{\{X-k\}}}_{corrected}$ and $ctr_{\{X-k\}_{corrected}}$ analogously. For $\overline{a_{\{X-k\}}}_{corrected}$, the updated value is simply the arithmetic mean of the current values in this group, taking into account the new value that is appended to the group. In $\overline{b_{\{X-k\}}}_{corrected}$ the same procedure is performed, except for types of groups where the lines are defined as $x = k$, which brings no meaning to this parameter. The $ctr_{\{X-k\}_{corrected}}$ update denotes simply an increment to the current counter value, which denotes the number of lines appended to such group. Note that although storing the sum of the values would avoid an additional multiplication, this solution avoids a possible overflow, allowing a generic application where a significant number of lines might be considered.

$$\overline{a_{\{X-k\}}}_{corrected} = \frac{\overline{a_{\{X-k\}}} \cdot ctr_{\{X-k\}} + a_n}{ctr_{\{X-k\}} + 1} \;, \qquad 5.24$$

$$\overline{b_{\{X-k\}}}_{corrected} = \begin{cases} \frac{\overline{b_{\{X-k\}}} \cdot ctr_{\{X-k\}} + b_n}{ctr_{\{X-k\}} + 1} \;, & F_n = F_{\{X-k\}} = 0 \\ 0 \;, & F_n = F_{\{X-k\}} = 1 \end{cases} \;, \qquad 5.25$$

$$ctr_{\{X-k\}_{corrected}} = ctr_{\{X-k\}} + 1 \;. \qquad 5.26$$

As for lines where the conditions of equation 5.23 are not satisfied for any *k* group, it is created a new group, as shown in equations 5.27, 5.28, 5.29 and 5.30.

$$\overline{a_{\{X-k\}}} = a_n \;, \qquad 5.27$$

$$\overline{b_{\{X-k\}}} = \begin{cases} b_n\,, & F_n = F_{\{X-k\}} = 0 \\ 0\,, & F_n = F_{\{X-k\}} = 1 \end{cases}, \tag{5.28}$$

$$ctr_{\{X-k\}} = 1 \ , \tag{5.29}$$

$$F_{\{X-k\}} = F_n \tag{5.30}$$

In Fig. 5.31, the process of lines clustering is summarized. The matrix returned by the Hough transform runs through system that starts by determining the angles for each row of this matrix, computing in the end, the max_y parameter. Then the lines outside the relative distance to max_y and angle ranges are discarded. Subsequently the process enters a loop where each row of the filtered Hough Transform matrix is parameterized. This parameter goes into an inner loop that checks all the existing groups of lines one by one. If the current line is similar to an existing group, it is appended to the respective group. If the loop reaches the end of the current H matrix and a similar line has not been founded, a new group is created with the settings of the current line. Then this process is repeated for all other lines, resulting in the end, with the defined above H matrix, which stands for Grouping lines Matrix. Fig. 5.32 shows an example where the output lines from the left markings are grouped into two sets.

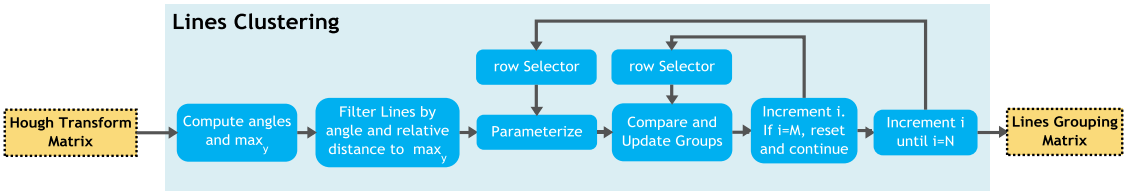


Fig. 5.31 A block diagram describing the task performed at the Lines Clustering block

Fig. 5.32 An example image showing two drawn lines overlaid to the raw image, representing the output of the lines Clustering block for this situation

5.3.3.3 - Line Selector

Once the lines that define the markings are parameterized and grouped, it should be determined the group of lines that correspond to the best approximation of the tangent line to the lowest point of the image.

This process is accomplished by checking the counter element in the H matrix returned in the previous step. The group that contains the greatest number of appended lines is assumed to be the group with the greatest probability to be the correct one. Therefore it is checked for the group that has the highest counter value, as shown in equation 5.31, for the left line, and analogously in equation 5.32 for the right line, then a vector is returned. This vector is

defined as row of the H matrix, and it is called G^{left} for the left line, and G^{right} for the right line.

$$G^{left} = H^{left} : \max\left(H^{left}_{ctr\{X-0\}}, H^{left}_{ctr\{X-1\}}, \cdots, H^{left}_{ctr\{X-n\}}\right) \quad 5.31$$

$$G^{right} = H^{right} : \max\left(H^{right}_{ctr\{X-0\}}, H^{right}_{ctr\{X-1\}}, \cdots, H^{right}_{ctr\{X-n\}}\right) \quad 5.32$$

where:

$$n \in [0, M] ,$$
$$M : number\ of\ groupings ,$$
$$H^{left} : the\ H\ matrix\ for\ the\ left\ line ,$$
$$H^{right} : the\ H\ matrix\ for\ the\ right\ line ,$$
$$H_{ctr\{X-n\}} : the\ ctr\ value\ of\ the\ vector\ defined\ for\ \{X-n\}\ in\ H\ matrix ,$$

5.3.3.4 - Vector Definition

In the previous step, two vectors were defined that contain the parameterization of lines, as shown in equations 5.33 e 5.34. Note that these vectors are stored for further use.

$$G^{left} = [a_{left} \quad b_{left} \quad ctr_{left} \quad F_{left}]^T \quad 5.33$$
$$G^{right} = [a_{right} \quad b_{right} \quad ctr_{right} \quad F_{right}]^T \quad 5.34$$

The final step defines the two points representing the line, illustrating them in the application for debugging purposes. These points also serve as reference for the rest of the vision algorithm, and mainly, to the navigation system. They are obtained by applying the line equation to fill the vertical extent of the screen, as shown in 5.35 and 5.36.

$$p_0 = \begin{cases} \frac{b}{a}\hat{\imath} + 0\hat{\jmath} , & F = 0 \\ a\hat{\imath} + 0\hat{\jmath} , & F = 1 \end{cases} \quad 5.35$$

$$, \quad p_1 = \begin{cases} \frac{height - b}{a}\hat{\imath} + (height)\hat{\jmath} , & F = 0 \\ a\hat{\imath} + 0\hat{\jmath} , & F = 1 \end{cases} \quad 5.36$$

where:

$$height\ denotes\ the\ height\ of\ the\ image\ screen$$

In fact, four points are determined: ${p_0}^{right}, {p_1}^{right}, {p_0}^{left} and\ {p_1}^{left}$ and stored in a matrix called p, as shown in equation 5.37. Note the last row introduces two new points: ${p_0}^{center}$ and ${p_1}^{center}$. These points will be discussed in the following section.

$$p = \begin{bmatrix} {p_0}^{left} & {p_1}^{left} \\ {p_0}^{right} & {p_1}^{right} \\ {p_0}^{center} & {p_1}^{center} \end{bmatrix} \quad 5.37$$

5.3.4 - Central Markings Isolation and Vector Acquisition

A process similar to that described above, is also made for the central region, i.e., it starts by creating a new image including merely the central region and then the vector corresponding to the position and orientation of the marking is computed.

5.3.4.1 - Central Markings Isolation

Based on the initial binarized image and the matrix that defines the side lines, the central region is isolated from the rest of the image, obtaining a new image that includes only the central region. This process is shown in the block diagram of Fig. 5.33.

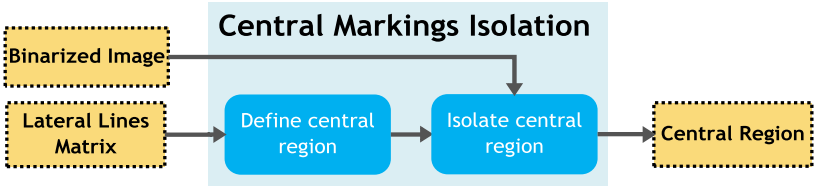


Fig. 5.33 The central markings isolation block

Once defined the parameters of the lines corresponding to the left and right markings, the inner region defined by these two lines, is isolated. This region, called the central region, corresponds to the yellow region illustrated in Fig. 5.34.

Fig. 5.34 An example of the central region, defined by the yellow area

Through the vectors G^{left} and G^{right}, defined in Equations 5.33 and 5.34, the corresponding equation of the lines is defined, as shown in equation 5.38 and 5.39. The λ displacement factor represents a small displacement of the actual regions. This is due to the fact that the inner region defined by $L(x)$ and $R(x)$ contains some parts of the left and right markings. This displacement factor eliminates those parts on the resulting region.

$$L(x) = \begin{cases} a_{left}x + b_{left} + \lambda\,, & F_{left} = 0 \\ a_{left} + \lambda\,, & F_{left} = 1 \end{cases} \quad 5.38$$

$$R(x) = \begin{cases} a_{right}x + b_{right} - \lambda\,, & F_{right} = 0 \\ a_{right} - \lambda\,, & F_{right} = 1 \end{cases} \quad 5.39$$

Thus, the resulting binarized image (obtained in section 5.3.1 - Pre-processing) is restricted to the region defined by these two lines. If the two lines are not visible (note that a flag is returned indicating the visibility of the lines), this region is restricted by the visible line and the remaining visible area to the left or right of this line, as shown in equation 5.40.

$$Central\ Region = \begin{cases} LR \cap RR \cap BI\ , & if\ LR\ and\ RR\ exist \\ RR \cap BI\ , & if\ only\ RR\ exists \\ LR \cap BI\ , & if\ only\ LR\ exists \end{cases} \quad (5.40)$$

where:

$$BI : Binarized\ Image\ ,$$

$$LR > \begin{cases} a_{left}x + b_{left} + \lambda\ , & F_{left} = 0 \\ a_{left} + \lambda\ , & F_{left} = 1 \end{cases},$$

$$RR < \begin{cases} a_{right}x + b_{right} - \lambda\ , & F_{right} = 0 \\ a_{right} - \lambda\ , & F_{right} = 1 \end{cases}.$$

The λ factor was set empirically to the 50 pixels. An example of the steps taken to obtain this region is shown in Fig. 5.35.

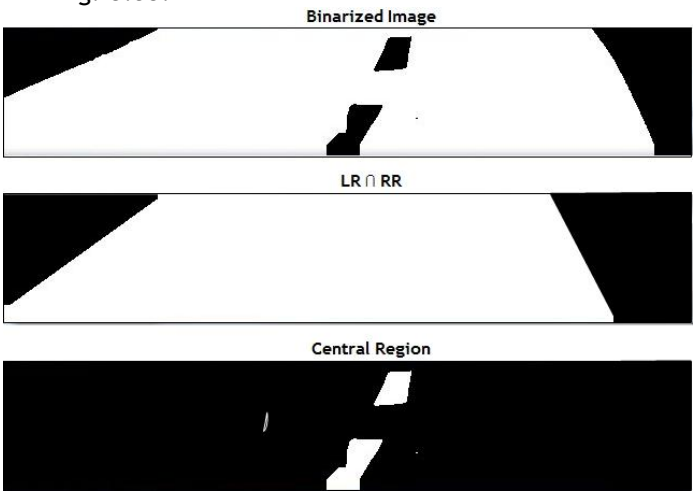


Fig. 5.35 The operations perform to isolate the central region to an example image

5.3.5 - Central Vector Acquisition

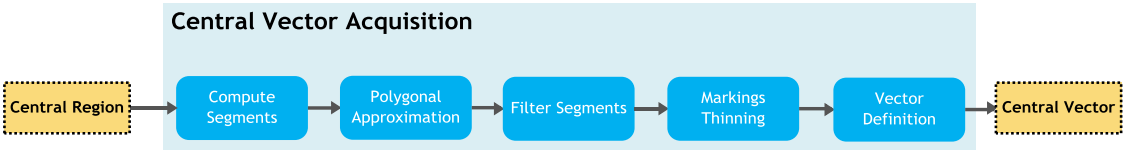


Fig. 5.36 The Central Vector Acquisition block

5.3.5.1 - Segments

The resulting image from the previous step may contain portions of the lateral markings, as well as some noise. Thus the resulting image is checked for inconsistencies, by checking all the segments found in the image.

OpenCV provides a tool that performs this procedure, in which the segments are referred to as *contours*. A *contour* is a list of points representing isolated segments in a binary image. The content of a *contour* is a segment. The information is stored using a data structure called *sequences*, which can be though as a multi-dimensional dynamic matrix. Each index of this structure represents a segment, and for a given segment, a list of the corresponding points is

available. It is possible to perform various analysis and operations, such as determining the area and perimeter of the segment, the center of mass, or a polygonal approximation of the shape of the segment [36].

Therefore, this task starts by searching all the contours in the resulting image. A matrix as shown in equation 5.41 is returned.

$$Segments(Central\ Region) = \begin{bmatrix} points_n \\ points_k \\ \vdots \\ points_0 \end{bmatrix} \quad 5.41$$

where:

$$n : number\ of\ segments\ found$$
$$points_k :\ the\ list\ of\ points\ of\ the\ k\ segment$$

A propriety that the central markings have is fixed size and form. Although the effect of perspective distorts the shape and area, the projection is still a trapezoid with its area within a certain range. Note that, because only the bottom of the incoming image is analyzed, the markings placed further away from the vehicle (which in infinity, have zero area) are not considered. Thus the segments in which the area is beyond a certain threshold and the respective polygonal approximation of it does not return a polygon with four sides, are discarded.

Analogously to the determination of the lateral markings, the interesting part of the central markings is the closest part to the bottom of the image. That way, only the closest marking to the bottom of the image is considered.

Condition 5.42 shows the conditions that the segments must satisfy to be considered in the next steps.

$$w < Area(Segments_i) < z \quad \wedge \quad PollygonalApproximation_{sides}(Segments_i) = 4 \quad 5.42$$
$$\wedge \quad CenterOfMass_y(Segments_i) < height - l$$

where:

$$w : lower\ area\ limit\ ,$$
$$z : upper\ area\ limit\ ,$$
$$l : maximum\ displacement\ of\ the\ y$$
$$- coordinate\ of\ the\ segment\ center\ of\ mass\ to\ the\ bottom\ of\ the\ resulting\ image\ .$$

5.3.5.2 - Markings Thinning

At this stage, the markings on the resulting image are prepared for the Hough transformation. This step took a different approach from that used in the detection of lateral markings, as the canny edge detector is not carried out. This is due to the fact that in the case of lateral markings, the ratio between length and width of the segments is much higher than that of the central markings (when they are dashed). Therefore, in order to prevent that the Hough transform detects horizontal lines, the resulting segment is thinned as much as possible, preserving mostly only the vertical component of the segment.

For each segment that satisfies the condition 5.42 (which in normal cases will be only one), a thinning algorithm is preformed, which basically performs multiple erodes with a 3×3 rectangular kernel to the segment, until its area is reduced to a certain level. Fig. 5.37 shows

a flowchart of this algorithm, where *s* denotes the maximum number of pixels allowed in the new segment, so that the segment is thinned until the area of it is bellow *s*.

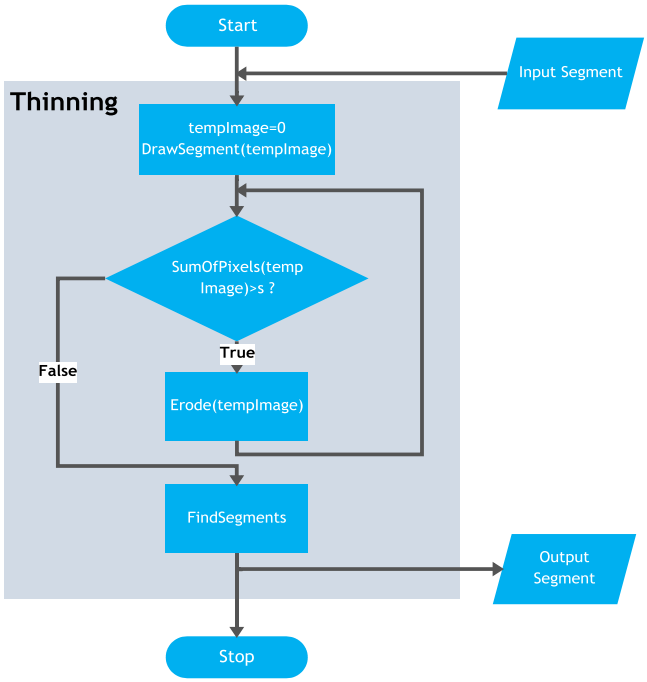


Fig. 5.37 Flowchart of the Markings Thinning Algorithm

The flowchart of Fig. 5.38 shows the steps taken to obtain the image where the Hough Transform is carried out.

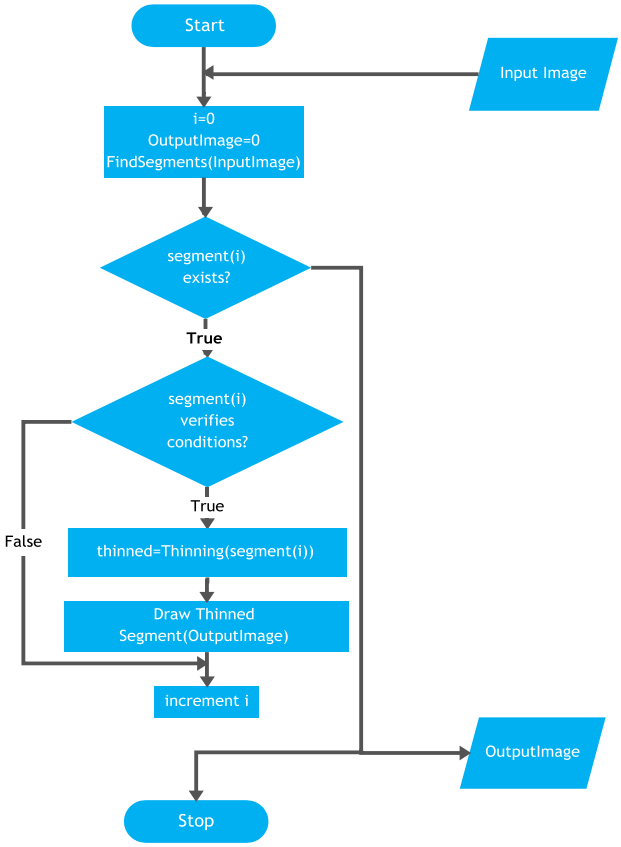


Fig. 5.38 Flowchart of the steps taken to prepare the central Markings image to the Hough Transformation

In the example of Fig. 5.39 it is shown the resultant image after this stage for an example image.

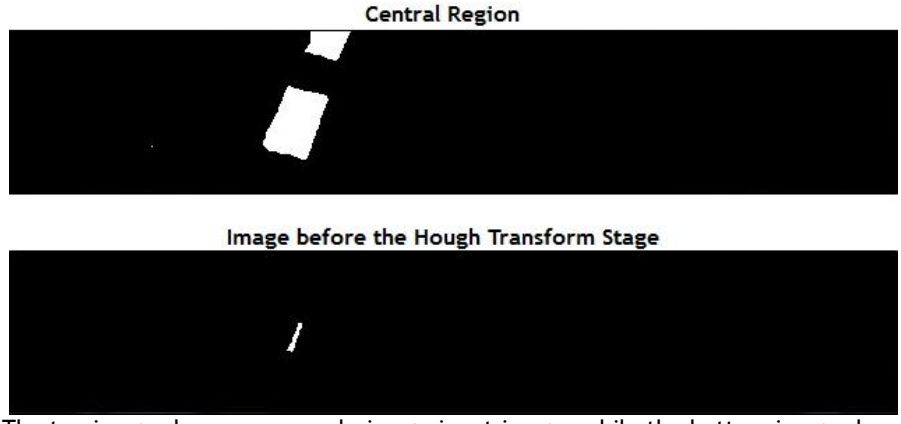


Fig. 5.39 The top image shows an example image input image, while the bottom image shows its result after the "Markings Thinning" stage

5.3.5.3 - Vector Definition

From this step on, the *thinned* image is run through the same set of procedures used in defining the left and right markings vector (shown previously in Sections 5.3.3.1 - Hough Transform, 5.3.3.2 - Lines Clustering, 5.3.3.3 - Line Selector and 5.3.3.4 - Vector Definition). There is a small refinement in the Hough Transform parameters used in this case, which is the minimum line length, due to the smaller size of this type of markings.

In the end of this process, a $G^{central}$ vector is obtained which defines the ${p_0}^{center}$ and ${p_1}^{center}$ points, completing thus, the lines matrix, shown in equation 5.37.

5.4 - Upper Signaling Panels

The detection and recognition of signs placed on the TFT display provides vital information to the high-level navigation controller of the vehicle.

The sequence of events used in this section follows a different philosophy regarding the previous section. In the earlier case, the initial preprocessing was performed, and then the rest of the system analyzes the image resulting from that task. With each new input image in the system, the entire sequence of processing and analysis tasks was performed again.

In this case, the system needs to be initialized, which only happens once, and for each incoming image, a direct search for the TFT display projection is performed. The reason why there is no such task as an initial preprocessing itself is due to the fact that the three spectral components of the image (red, green and blue) are considered in the search of the TFT. Once this region is defined, the three spectral images are discarded, and the system enters a different phase which is the determination of the received signal, based on the

template images, loaded during the initialization procedure. This procedure is summarized in the diagram of Fig. 5.40.

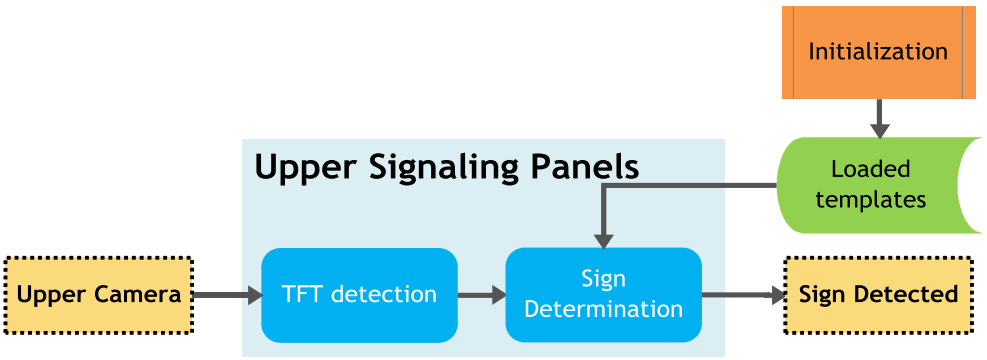


Fig. 5.40 Diagram of the Upper Signaling Panels procedure

5.4.1 - Initialization

Before any image processing task, this system is initialized. This initialization consists in loading to the application environment the template images, that will be later compared to the ones received from the TFT display.

One of the features immediately verified is that, although the five possible signs have different colors, all of them have different shapes. Taking advantage of this fact, unnecessary processing costs are saved by loading already binarized signal templates.

Fig. 5.41 illustrates the five possible signals loaded into the system, in which the dimensions are 50x50px.

Fig. 5.41 The pre-binarized template images loaded into the Upper Signaling Panels system

These images are loaded into a vector, as shown in 5.43.

$$templates = \begin{bmatrix} template\ 0 \\ template\ 1 \\ template\ 2 \\ template\ 3 \\ template\ 4 \end{bmatrix} \tag{5.43}$$

where:

$template\ 0 : "end"$

$template\ 1 : "stop"$

$template\ 2 : "straight\ ahead"$

$template\ 3 : "turn\ left"$

$template\ 4 : "park"$

5.4.2 - TFT projection detection

In situations where the camera is capturing the TFT display, the rectangular-shaped object is distorted, thus turning into a trapezoid. This trapezoid is the region of interest to analyze, and its detection is explained in this section. As said earlier, the detection of this trapezoid is performed in the 3 color planes of the image.

The algorithm starts by splitting the raw image into the three color components of the image, resulting in three matrices -$[R], [G], [B]$. An example of such a procedure is shown in Fig. 5.42.

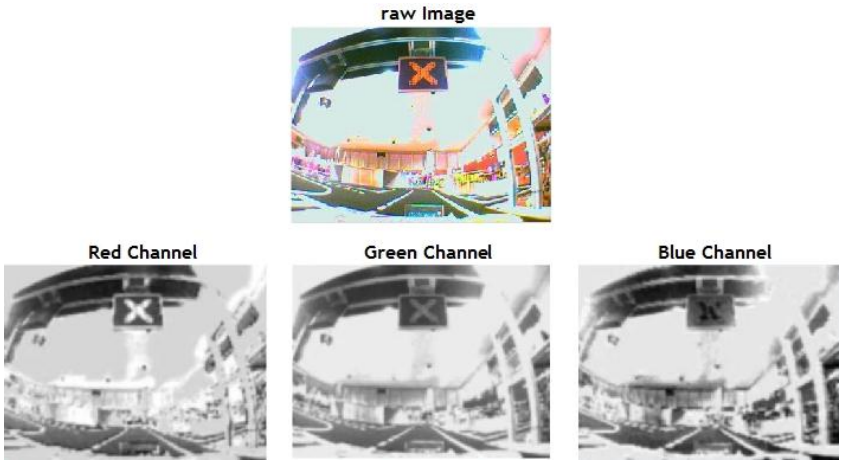


Fig. 5.42 An example of the three color components in an image

For each matrix the method that follows is applied, which aims to accurately detect the TFT display trapezoid.

For every color plane, an adaptive threshold in performed, followed by a dilatation procedure, in order to avoid eventual discontinuities. An example of such transformation is shown in Fig. 5.43.

Fig. 5.43 An example of the adaptive threshold performed to the three color components in the image

Next, a procedure of contour finding is performed on the three matrices and consequently, the contours are approximated to polygonal shapes. The output of the polygonal approximation procedure is a *n*-D vector for each approximated contour, where *n* denotes the number of sides of the polygon. Such a vector is represented in equation 5.44.

$$PolygonalApproximation_{contour\ k} = [p_o(x,y) \quad p_1(x,y) \quad \cdots \quad p_n(x,y)]^T \tag{5.44}$$

where:

$$k : contour\ index\ ,$$
$$p_i : 2D\ coordinates\ of\ the\ vertice\ i\ ,$$
$$n\ :\ number\ of\ polygon\ sides\ .$$

Then, a filtering to the returned polygonal approximation vectors is performed, based on their area, number of sides, and convexity. Note that, in Euclidean space, an object is convex if for each pair of points inside the object, each point on the line segment that joins them is also inside the object. A squared object verifies this property, and an elimination based on this propriety filters many undesired regions. OpenCV already has an integrated procedure that automatically performs this checking. The corresponding conditions are represented in 5.45.

$$Area(approximation_i) > a \ \wedge\ Sides(approximation_i) = 4 \\ \wedge\ verifies\ Convexity(approximation_i) \qquad 5.45$$

Fig. 5.44 illustrates the contours found on the image obtained in the example of Fig. 5.42, after it was filtered by the conditions in 5.45.

Fig. 5.44 The contours found in an example image, filtered by conditions in equation 5.45.

The next procedure, determines to the maximum inner angle of the four-sided polygon. This is due to the fact that in a perfect square each inner angle has a value of 90º. Filtering the polygons whose maximum angles are greater than a certain level, defines how ideal the polygon is. Since the sum of all inner angles is 360º, only three angles of these angles are necessary to define the how "ideal" the square is.

The determination of the inner angle is shown in equation 5.46, in which for three generic points (p_i, p_{i+1}, p_{i+2}) the cosine of the angle is determined.

$$\cos\varphi = f(p_i, p_{i+1}, p_{i+2}) = \frac{\Delta x_1 \cdot \Delta x_2 + \Delta y_1 \cdot \Delta y_2}{\sqrt{\left(\Delta {x_1}^2 + \Delta {y_1}^2\right) + \left(\Delta {x_2}^2 + \Delta {y_2}^2\right)}}$$

where:

$$\Delta x_1 = p_{i+1}(x) - p_i(x)$$
$$\Delta y_1 = p_{i+1}(y) - p_i(y)$$
$$\Delta x_2 = p_{i+2}(x) - p_i(x)$$
$$\Delta y_2 = p_{i+2}(y) - p_i(y) \qquad 5.46$$
$$p_i(x): the\ x - coordinate\ of\ the\ p_i\ point$$
$$p_{i+1}(x): an\ adjacent\ point\ to\ p_i$$
$$p_{i+2}(x): an\ adjacent\ point\ to\ p_i$$

This way, the inner angles of the polygon resulting from the previous step are computed, which is represented by the vector Φ in 5.47. This vector is determined given the C_i vector, which represents the $i-th$ polygonal approximation returned.

$$C_i = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \qquad \Phi = \begin{bmatrix} f(p_2, p_0, p_1) \\ f(p_3, p_1, p_2) \\ f(p_0, p_2, p_3) \end{bmatrix} \tag{5.47}$$

Therefore, only the squares that verify condition 5.48 are considered, where a is the maximum cosine deviation from the ideal value (note that $lim_{\theta \to 90º} \cos\theta = 0$). This parameter was set to 0.2, which allows a maximum inner angle of approximately $80°$.

$$\max(\Phi) < a \tag{5.48}$$

Finally the filtered contours are drawn on the input image for debugging purposes. Fig. 5.45 shows an example of the final output obtained at this stage after the filtering.

Fig. 5.45 Final output of the TFT projection detection block, where the TFT projection represented by the blue box

Note that, although in most of the situations this in the only contour found, that is not necessarily true for all the situations. Hence, other contours might arise, meeting all the requirements need to be considered as such, or no contours might be detected at all. Fig. 5.46 covers one of these situations, where two sets of contours are found. The way to overcome this situation will be discussed in the following section.

Fig. 5.46 Two contours found in the detection of the TFT projection

This procedure is summarized in the diagram of Fig. 5.47.

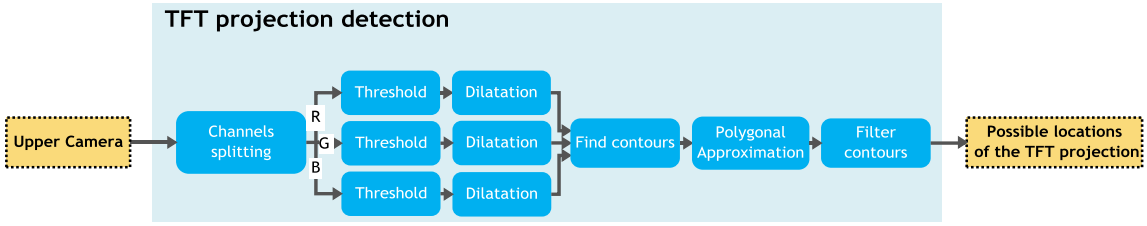


Fig. 5.47 Diagram of the procedure followed to determine the position of the TFT projection

5.4.3 - Sign Determination

Once determined the location of four points defining the location of trapezoids that outline the possible location of the display, the corresponding image is isolated, thus creating an image solely with the contents of these contours. This image is then compared with pre-loaded templates in order to determine which signal is present. Since this procedure is performed to all the possible locations of the TFT projection, they are all compared with the templates. The result of these matches is then selected to determine which region corresponds to the actual display, so that the present signal is returned.

A diagram showing these steps is shown in Fig. 5.48.

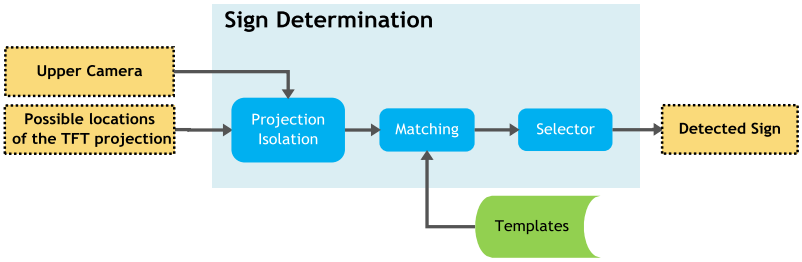


Fig. 5.48 Diagram of the procedure followed to determine the sign projected in the TFT display

5.4.3.1 - Projection Isolation

The isolation of each region corresponding to the position of the display is made through perspective transformation. However, a treatment of the incoming data from the previous step should be performed. Note that this procedure is performed for all contours arising after the stages of filtering. A block diagram of this step is represented in Fig. 5.49.

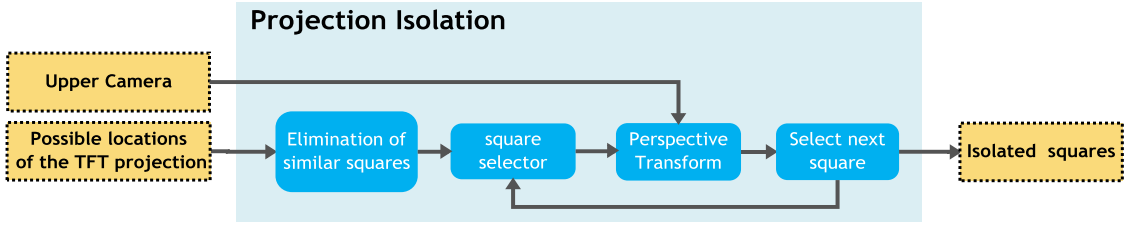


Fig. 5.49 Block diagram of the tasks followed in the Projection Isolation step

Elimination of similar squares

It turns out that at the end of the previous procedure, some trapezoids arose virtually with the same size and placed in the same positions. This is due to the search in three color channels and also because the regions are not perfectly rectangular, which allows various quadrangular approximations. Therefore, an elimination of squares whose centers are alike is performed. For this purpose, when performing the computation of the C_i vector (equation 5.47), which stores the four corners of the square, a new element is actually appended to it, so that the coordinates of the center of mass are known. The previous step returns thus, an N-by-5 matrix, where N denotes the number of squared regions. Each row of it represents the C_i vector, with the added element of the center of mass.

This procedure compares all the centers of mass of the contours found, and if their difference is below a pre-set value, one of the trapezoids is marked to be eliminated. The condition is shown in 5.49.

$$\frac{c_j(x) - c_i(x)}{c_i(x)} < \delta \quad \wedge \quad \frac{c_j(y) - c_i(y)}{c_i(y)} < \delta \tag{5.49}$$

where:

$$c_x(j) \ : \ the\ x-coordinate\ of\ the\ center\ of\ mass\ of\ contour\ j \ ,$$
$$i \in [0, n] \ ,$$
$$j \in [i+1, n] \ ,$$
$$n \ : \ total\ squares\ contours \ .$$

Perspective Transform

The region defined by the four points is then adjusted to match the dimensions of the pre-loaded sign templates. This adjustment is accomplished through a perspective transform that maps the inner region defined by the respective contour, into a perfectly rectangular-shaped region. The perspective transform (also called *homographies*) uses a 3-by-3 matrix, that computes the way in which a plane in three dimensions is perceived by a particular observer (in this particular case, the camera), which is not looking straight on that plane. Given two sets of 4 points, the source and destination points, this matrix is computed automatically in OpenCv, through a list of correspondences. The vertices of the region to be transformed fill a a source vector, while the destination vector b corresponds to the corresponding points in the destination image. The location of the inner points in the resulting image, are thus corrected by linear interpolation to fit the dimensions of the destination image. Such transformation in represented mathematically in homogeneous coordinates, as shown in equation 5.50.

$$p_d{}' = H_{sd} \cdot p_s \quad \rightarrow \quad p_d = {p_d{}'}/{w'} = \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{5.50}$$

where:

$$p_s = \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} \ (source\ points) \ ,$$

$$H_{sd} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \ (transformation\ matrix)\ ,$$

$$p_d{}' = \begin{bmatrix} x_d w' \\ y_d w' \\ w' \end{bmatrix} \ (destination\ points)\ .$$

To accomplish this transformation, it is necessary to know the geometric relationship between the vertices of the rectangle, by determining the top-left, upper-right, bottom-left and bottom right vertices. Since the method of determining the polygonal approximation implemented in OpenCV does not return a vector with a specific ordering criteria, the vector was reordered in accordance with the geometric relation between the vertices of the trapezoid. The ordering notation used was that of Fig. 5.50.

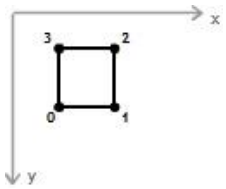


Fig. 5.50 Geometric Notation used in the re-ordering of the vector returned by the polygonal approximation.

Mathematically the re-ordered vector is reflected by equations 5.51,5.52,5.53 and 5.54.

$$p_{o\_corrected} = \{p_o, p_1, p_2, p_3\} : \\ (\,|p_i(x) - min_x| < |p_i(x) - max_x|) \ \wedge \ (|p_i(y) - min_y| < |p_i(y) - max_y|) \qquad 5.51$$

$$p_{1\_corrected} = \{p_o, p_1, p_2, p_3\} : \\ (\,|p_i(x) - min_x| > |p_i(x) - max_x|) \ \wedge \ (\,|p_i(y) - min_y| < |p_i(y) - max_y|) \qquad 5.52$$

$$p_{2\_corrected} = \{p_o, p_1, p_2, p_3\} : \\ (\,|p_i(x) - min_x| > |p_i(x) - max_x|) \ \wedge \ (\,|p_i(y) - min_y| > |p_i(y) - max_y|) \qquad 5.53$$

$$p_{3\_corrected} = \{p_o, p_1, p_2, p_3\} : \\ (\,|p_i(x) - min_x| < |p_i(x) - max_x|) \ \wedge \ (|p_i(y) - min_y| > |p_i(y) - max_y|) \qquad 5.54$$

where:

$$min_x = min\{p_o(x), p_1(x), p_2(x), p_3(x)\}\ ,$$
$$min_y = min\{p_o(y), p_1(y), p_2(y), p_3(y)\}\ ,$$
$$max_x = max\{p_o(x), p_1(x), p_2(x), p_3(x)\}\ ,$$
$$max_y = max\{p_o(y), p_1(y), p_2(y), p_3(y)\}\ ,$$

$$p_{i\_corrected} : \ the\ corrected\ value\ of\ p_i\ ,$$
$$p_i(x) : \ the\ x - coordinate\ of\ p_i\ ,$$
$$p_i(y) : \ the\ y - coordinate\ of\ p_i\ ,$$
$$i \in [0,3]\ .$$

At this stage it is possible to perform the geometric transformation of the region, to match the size of the templates. Since the size of the templates is 50x50, the destination vector b used to compute the transformation matrix was set in accordance with the notation, as well as the source vector a, as shown in 5.55 and 5.56.

$$b = \begin{bmatrix} (0,0) \\ (50,0) \\ (50,50) \\ (0,50) \end{bmatrix} \quad 5.55$$

$$a = \begin{bmatrix} p_{0\_corrected} \\ p_{1\_corrected} \\ p_{2\_corrected} \\ p_{3\_corrected} \end{bmatrix} \quad 5.56$$

After applying this transformation, it is then obtained a new image with the respective region isolated. Fig. 5.51 shows an example of this set of operations applied to a raw image.

Fig. 5.51 An example of an isolated image contemplating the region of the TFT display

5.4.3.2 - Matching

At this stage, this sub-system created an image of fixed dimensions, which corresponds to the image displayed in the TFT display. It is therefore determined which is the sign being displayed in the TFT display. For this purpose, the technique of template matching is used by comparing a binary image corresponding to the isolated region with a set of templates. It was chosen to binarize the image, as this way the computation of the correlation factor is greatly simplified (less computationally expensive). It has also a more significant and accurate value, due to the fact that its color is not a decisive parameter, but its shape.

The preparation for the template matching operation starts by converting the isolated image to grayscale, followed by simple blur smoothing and erosion filters. A dilation operation was then carried out, so that the resulting image preserves as much as possible its original shape. Fig. 5.52 shows an example of the resulting image of such set of operations.

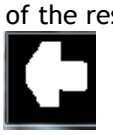


Fig. 5.52 An example of a binarized isolated image for further template matching procedure

The template matching method in then applied between the isolated image and the binarized template through the square difference matching method. The correlation value for

each template is stored in a 5-D vector, as shown in equation 5.57. Note that the index i indicates the value of the correlation between the template image i, from the $templates$ vector, and the image found.

$$corr' = \begin{bmatrix} corr_0 \\ corr_1 \\ corr_2 \\ corr_3 \\ corr_4 \end{bmatrix} \quad 5.57$$

Note that in the square difference matching method, the best match has the value 0, so the values in the vector are re-adjusted to the range of 0 to 100%, through the operation shown in equation 5.58.

$$corr = 100 - (100 \cdot corr') \quad 5.58$$

Finally, it is returned the position in the vector where the maximum element is placed, where this index indicates the number of signal present, consistent with the $templates$ vector.

5.4.3.3 - Matching Selector

The matching is performed for all contours found in the original image, resulting in a final vector of correlation factors with the various templates. It turns out that if the matching is being performed in a region that does not correspond to the actual display, the correlation factor with any of the templates will be small (ideally zero).

Therefore, it is determined the maximum value of elements of all vectors returned. The vector of the correct region corresponds then to the maximum value of these maximums. The region corresponding to the TFT corresponds then to the value j shown in equation 5.62.

$$CorrectRegion = j \ : max\{max_0, max_j, \cdots, max_k\} \quad 5.59$$

where:

$$max_i = max\{corr_0^i, corr_1^i, corr_2^i, corr_3^i, corr_4^i\} \ ,$$

$$corr^i \ : \ resultant\ corr\ vector\ for\ region\ i \ ,$$

$$k \ : \ number\ of\ resultant\ regions \ .$$

Finally, it is returned the position in the vector $corr$ of region j, where the maximum element is placed, where this index indicates the signal present, consistent with the $templates$ vector, as shown in 5.63.

$$Signal = s \ : max\{corr_s^j\} \quad 5.60$$

5.5 - Crosswalk

The crosswalk is defined as a set of seven equally spaced white rectangles aligned vertically, and it defines the beginning of the track. This way, the rectangles are parallel to each other as well as to the road limits. The detection of the crosswalk provides several

indicators of extreme importance for the navigation system. Therefore, this procedure determines at all temporal instants if the crosswalk is present in the fused images captured by the cameras.

The detection of such feature follows a similar procedure to that used in the detection of the upper signaling panels, as the algorithm starts by finding contours in the three color planes, where its polygonal approximation is given by four sides. As opposed to the TFT display, the markings of the crosswalk are disposed horizontally, which increases the effect of perspective, especially due to the fact that it is desired to anticipate the detection of it as much as possible. In greater distances, the projection of the physical rectangles becomes significantly distorted. Therefore, the resulting contours are not restricted based on their inner angles, and the searching is performed throughout the whole raw image.

5.5.1 - Contours Searching and Filtering

The low level processing performed before searching for contours aims to preserve the shape and the physical separation between the rectangles. This processing level consists in performing an adaptive threshold, with unitary mean and a relative high offset value. It is followed by a closing operation, to preserve the convexity of the shape, preceded by erosion in order to reduce some noisy particles that might be in the vicinity of these segments that eventually allow two shapes to merge.

Its resulting images, although being noisy (especially due to the high offset for the adaptive threshold), fulfill these requirements, as shown in Fig. 5.53 for the three color channels.

Fig. 5.53 An example of same image binarized in three color channels.

It is then conducted the search for contours itself, followed by a polygonal approximation on those contours. This search is narrowed to: convex regions, polygon number of sides of 4 and to an area range. At the end of the search, carried out in the three images, it is obtained a matrix N-by-6, as shown in equation 5.61. Note that, as in the detection of the TFT display, the center of mass of each region is appended to the matrix. In this situation its area is also computed, which is required for the subsequent steps.

$$C = \begin{bmatrix} p_0^0 & p_1^0 & p_2^0 & p_3^0 & A^0 & c^0 \\ p_0^1 & p_1^1 & p_2^1 & p_3^1 & A^1 & c^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_0^N & p_1^N & p_2^N & p_3^N & A^N & c^N \end{bmatrix} \tag{5.61}$$

As previously verified, several segments for the “same” location of the contour are retrieved. The similar contours are accordingly discarded.

These tasks are summarized in the diagram of Fig. 5.54.

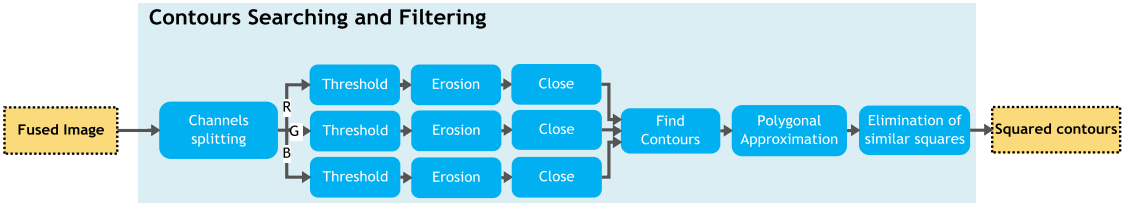


Fig. 5.54 Diagram of the tasks performed in the Contours Searching and Filtering block

5.5.2 - Markings Clustering

The resulting regions are then submitted to a clustering process based on two criteria: the location of the center of mass and the area.

The projection of the rectangles in the picture follows a well-defined orientation. In other words, the y coordinate of the center of mass of the rectangles is approximately the same. However, due to the location of the cameras, the resulting image from image fusion step creates a symmetry effect by the horizontal half of the image. It follows that the orientation of the rectangles is different in the two halves of the image, as shown in Fig. 5.55, where the vehicle is placed parallel to the crosswalk. The orientation is also not the same when the vehicle is not parallel to the crosswalk.

Fig. 5.55 An example of an image showing the crosswalk and the symmetry effect

However this discrepancy is neglected, since the displacements are not very significant compared to the image dimensions. Therefore, a grouping by y-coordinate of the centers of mass is performed, as shown in condition 5.62. The δ factor defines the relative deviation with respect to the actual group value, and was set empirically to a value of 0.1.

$$C_i \in \{X-k\} \leftrightarrow c_{\{X-k\}}(y)\cdot(1-\delta) < c_i(y) < c_{\{X-k\}}(y)\cdot(1+\delta) \tag{5.62}$$

The process of creating groups is analogous to that used in section 5.3.3.2 - Lines Clustering, returning a matrix J with the mean value of y-coordinate of the center of mass and the number of regions appended to it, as shown in equation 5.63.

$$J = \begin{bmatrix} \overline{c_{\{X\}}(y)} & ctr_{\{X\}} \\ \overline{c_{\{X-1\}}(y)} & ctr_{\{X-1\}} \\ \vdots & \vdots \\ \overline{c_{\{X-N\}}(y)} & ctr_{\{X-1\}} \end{bmatrix} \tag{5.63}$$

Another property of these regions is sharing a similar area. In practice, the projection in the image plane also results in regions with different areas, but that effect is neglected. Consequently, another matrix is obtained, containing the mean area values for each group,

called A, as shown in 5.64. The analogous conditions to fit a certain contour in an existing set are defined in 5.65.

$$A = \begin{bmatrix} \overline{a_{\{X\}}} & ctr_{\{X\}} \\ \overline{a_{\{X-1\}}} & ctr_{\{X-1\}} \\ \vdots & \vdots \\ \overline{a_{\{X-N\}}} & ctr_{\{X-1\}} \end{bmatrix} \quad 5.64$$

$$C_i \in \{X-k\} \leftrightarrow \overline{a_{\{X-k\}}} \cdot (1-\delta) < A^i < \overline{a_{\{X-k\}}} \cdot (1+\delta) \quad 5.65$$

5.5.3 - Cluster Selection

The attainment of the sets that correspond to correct markings is made through a philosophy similar to those used before. The mean y-coordinate value, for the maximum ctr is computed, and the same is done for the A matrix. These two values represent the greatest set of contours whose areas and orientations are similar, and they are obtained as shown in equations 5.66 and 5.67.

$$a = \overline{a_{\{X-i\}}} \ : \ i = \max(A_{ctr\{x\}}, A_{ctr\{x-1\}}, \cdots, A_{ctr\{x-N\}}) \quad 5.66$$

$$b = \overline{c_{\{X-j\}}(y)} \ : \ j = \max(J_{ctr\{x\}}, J_{ctr\{x-1\}}, \cdots, J_{ctr\{x-M\}}) \quad 5.67$$

where:

$$a : the\ mean\ area\ value\ of\ the\ set\ with\ the\ greatest\ number\ of\ appended\ contours \ ,$$

$$b : the\ mean\ y-coordinate\ value\ of\ the\ set\ with\ the\ greatest\ number\ of\ appended\ contours \ ,$$

$$A_{ctr\{x-k\}} \ : \ the\ ctr\ value\ of\ the\ \{X-k\}\ set\ in\ the\ A\ matrix \ ,$$

$$J_{ctr\{x-k\}} \ : \ the\ ctr\ value\ of\ the\ \{X-k\}\ set\ in\ the\ J\ matrix \ ,$$

$$N \ : \ number\ of\ groupings\ by\ Area\ value \ ,$$

$$M \ : \ number\ of\ groupings\ by\ y-coordinate\ value \ .$$

The intersection of these two sets reflects the group most likely to contemplate the rectangular-shaped objects that represent the crosswalk. That way, the matrix C (equation 5.61), is constrained to contours in which the A and c values of this matrix are similar to a and b values, shown in equations 5.66 and 5.67. In order to do so, it is performed the inverse process of grouping. By comparing a and b values with $(A^0, \dots, A^n)$ and $(c^0, \dots, c^n)$ through conditions 5.62 and 5.65, it is determined which rows of the initial matrix C are discarded.

At this point, the remaining rows are counted, and it is assumed that the crosswalk is present if at least 5 out 7 markings are detected. This sub-system returns then a flag, indicating whether the crosswalk is present or not.

5.6 - Chapter Conclusions

This chapter details the computer vision approach which encompasses the most important sensory information of the system. The track limits are represented by three vectors denoting the three set of markings captured by the two cameras. Based on this information, it is possible to compute the lateral distance of the vehicle on a given lane. This is the main localization feature used by the navigation routine. Along with this, the upper signaling panels block provides ordering information, so that the navigation process satisfies the given command. The computer vision subsystem also detects the crosswalk, comprising an important track landmark. The interpretation of these outputs is detailed in the next chapter.

The detection of road limits returns three vectors reflecting the tendency of the markings on the closest points to the vehicle. The result of this approach is similar to that followed in the ATLAS project, since line markings are effectively determined. However, it does not come from the expensive operation of perspective transformation needed to generate the top view, nor is it required an accurate image fusion which is quite expensive as well. The method itself is completely different since a grouping of similar lines is performed and a voting procedure is carried out, rather than using a fixed statistical criterion, which could compromise the use of it in other environments. The vision scheme behind the ROTA project, detected solely one of the lines, in order to maintain a fixed distance to it. Although effective, this method is far too competition-oriented and this was not the philosophy behind this work. The VERSA robot was thought to the rules of 2005 edition, where only one track existed. Thus, although the implemented method in this thesis is somehow also thought to maintain the robot between two lines, the required complexity is far superior to that implemented in VERSA. This way, the VERSA method would not be practicable.

Regarding the crosswalk detection, it was based on searching for a well-defined pattern of rectangles, as opposed to the method used in ATLAS, where the input image is systematically searched for a template image, which was to be rotated solidary with the orientation of the robot. The implemented method was thought to be significantly less computationally expensive. The ROTA project did not detect the crosswalk through the vision system. Instead it was based on odometry and track mapping, which proved to be effective, although not a general solution. The detection of the crosswalk in the VERSA project is quite simple and effective. However, as the track consisted of only one lane, the width of the crosswalk was consequently smaller. Thus it was only one camera was needed to captured the entire track width, allowing an easy and “clean” detection of two "perfect" and horizontal-aligned lines. This solution is although rather closed with respect to the general approach sought in this thesis.

Finally, the detection of the upper signaling panels consisted in detecting the respective region of the TFT, proceed by a perspective transformation of its content, adjusted to the dimensions of the signal templates. Thus, the correlation is determined only once, reducing the computational costs and increasing the robustness to various panel orientations. This approach, avoids the transformation of the image color space to HSV and subsequent color segmentation done in the ATLAS project. It also avoids the time-consuming task of searching for various templates along the whole image as in ROTA, which is not scale and rotation invariant. The VERSA project is also not scale and rotation invariant.

Chapter 6

Navigation System

6.1 - Vehicle Dynamics and Control

The vehicle used in the context of this work is definable by the Ackermann Steering Principle. It defines the geometry that is applied to all vehicles, enabling a correct turning angle of the steering wheels to be generated when negotiating a corner or a curve. Therefore, it solves the problem of wheels on the inside and outside of a turn, needing to trace circles of different radii. For that purpose an arrangement of linkages is obtained, that applies different angles to the turning wheels. This arrangement ensures that at any angle of steering, the centre point of all of the circles traced by all wheels will lie at a common point. Hence both wheels turn on arcs parallel to one other, as depicted in Fig. 6.1. The common point of intersection is denominated *O* in this picture, so in order to curve the vehicle applies different angles to the wheels, tracing two circles with *r1* and *r2* radius.

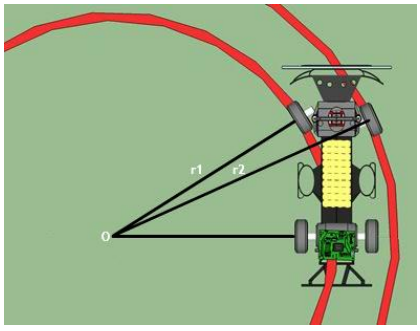


Fig. 6.1 Two arcs defined when the vehicle is performing a curve

As for the dynamics of the system, the plant model has not been determined, because given its complexity and non-linearity, obtaining a model is extremely difficult. Thus, the dynamics of the system were approximated to a linear and the first order system

6.2 - World Perception

World perception delivers key aspects to the planning system, so that it can take correct actions. Its main goal is to determinate the position of the vehicle on the track. In the context of this thesis, it is seek to obtain a solution as generic as possible, enabling the vehicle to navigate in any kind of track. Nevertheless, some of the key features of this specific track must be considered.

This phase also aims at ensuring the proper transfer of data from the physical sensors placed on the vehicle, and a correct acquisition of the vision system outputs. Thus a well-defined architecture of data transfer and manipulation is needed.

To that end, three main classes were created, operating on different levels of the navigation system. These are classes are:

- *CCAR*, stores data from vehicle sensors and sends commands to it. Trajectory planning is performed at this level.
- *CTRACK*, performs all the filtering of the values obtained by vision system, as well as the computation of measures. The location of the vehicle itself is performed at this level.
- *CUDP*, handles the data exchange between the decision layer and the control layer, through UDP packets.

In the scope of world perception, the data flow between these classes is that shown in Fig. 6.2.

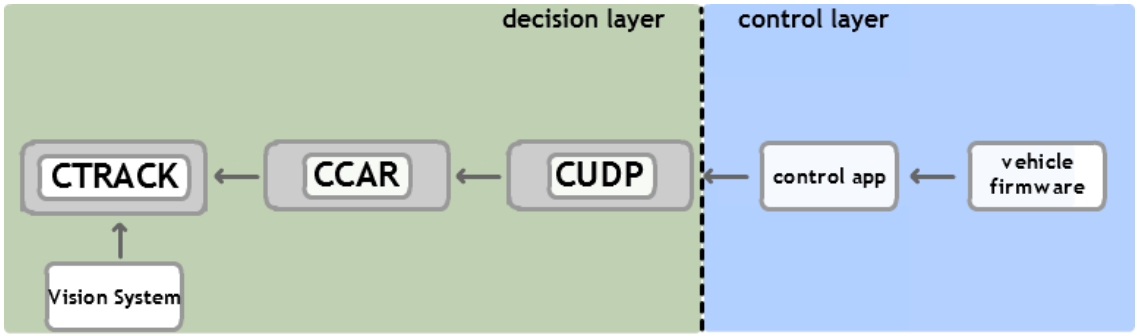


Fig. 6.2 Direction of data flow within the various components of the system used in world perception

6.2.1 - Data Acquisition and processing

6.2.1.1 - Vision system outputs – interpretation and filtering

Lateral Distance determination

The output provided by the sub-system of track limits consists in the location of two points for a set of three lines. These lines define the tangent at the bottom of the image of the curves that define the road limits, as well as the central markings. It is also returned if

these curves are available. This stage aims to determine the relative distance of the vehicle relative to these track limits.

The approach followed does not seek to obtain accurate metric measurements, but qualitative measures. So it was set a distance notation, defined in percentage terms, as shown in Fig. 6.3. Therefore, when the car is placed on the left lane (red region), it is assumed that the relative distance is negative, and vice-versa for the right lane (yellow region). This way, the navigation is invariant to different tracks, as long as the markings are still visible.

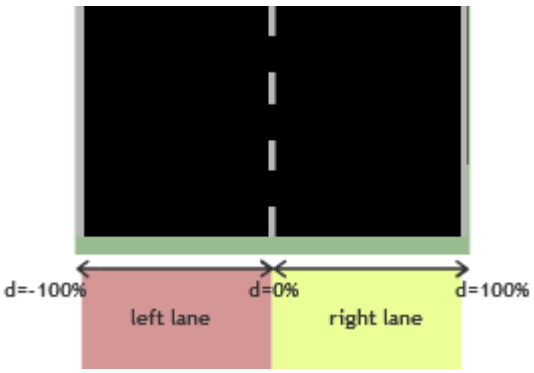


Fig. 6.3 The notation used to obtain the relative lateral distance

Assuming that cameras are placed exactly at the center of the vertical orientation of the vehicle, then the horizontal half of the image towards the central markings indicates in which track lane the vehicle lays on. Fig. 6.4 illustrates two situations in which the vehicle is on the right (top image) and left lane (lane image) of the track. As it can be seen, when the vehicle is in the right lane, the horizontal half is placed between the central and right markings, green and blue lines, respectively. Analogously, this arrangement occurs when the vehicle is placed on the left lane.

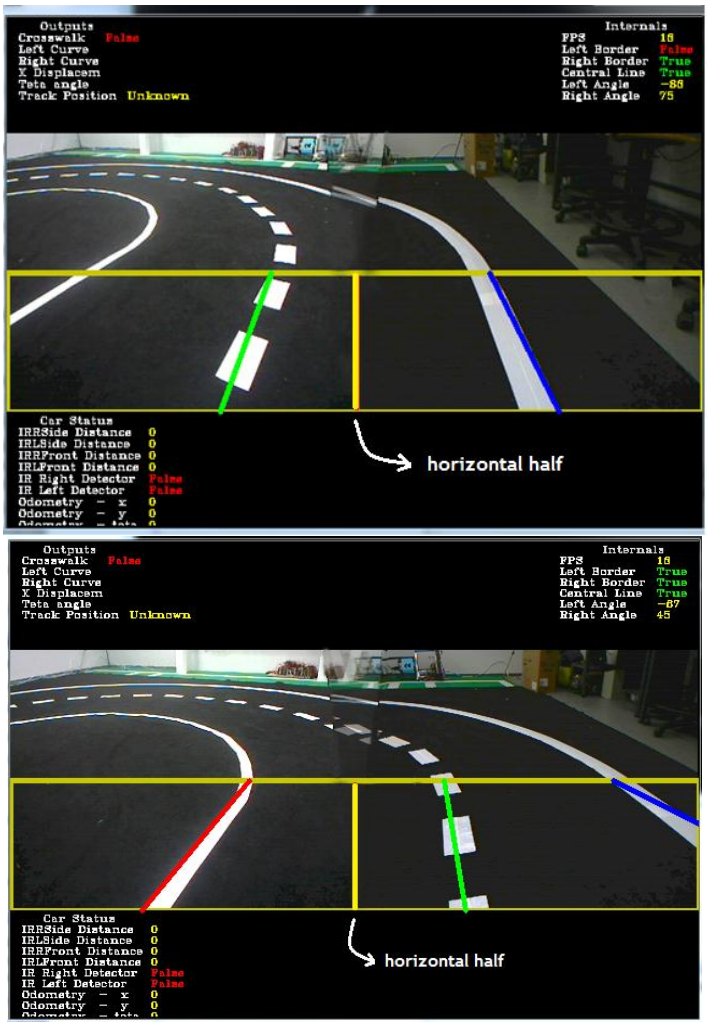


Fig. 6.4 Arrangement of the horizontal half of the image with respect to road markings, in situations where the vehicle is on the right (top image) and left (bottom image) lanes.

Following the same philosophy, the relative distance is likely to be obtained by considering the following: if the vehicle is on the right lane (right and center lines visible), then the ratio between the distance of half the image to the central markings and the width of the lane indicates the relative lateral localization. The width of the lane is considered to be the distance between the central and right markings. Note that this measurement is not accurate, as this approach assumes that as the vehicle moves laterally, the relative lateral distance changes proportionally. It is easily understandable, considering the following situation: when the vehicle is placed in the middle of the right lane (d=50%), and moves to the center, the width of the lane increases at a higher ratio than the distance to the center, due to the perspective effect. This first situation is depicted in Fig. 6.5-a), where the parameter *b* is considered to be doubled than *a*, so the vehicle is located in the middle of the right lane, which is correctly assumed. Then, the vehicle moves closer to the central line, and the image captured is the on illustrated on Fig. 6.5-b), where *a* halved, and *b* doubled. In this situation the ratio between distances indicates that the vehicle is placed at a d=12,5%, which is not correct.

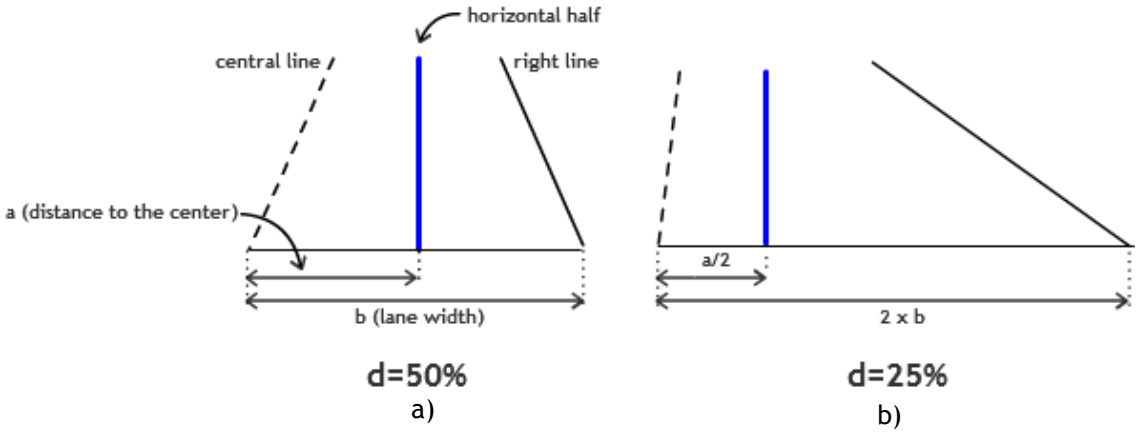


Fig. 6.5 Perspective effect in the attainment of lateral distances

However, this effect is neglected, since it is intended that the vehicle is placed most of the times in the middle of two lanes, and if eventually it is not, the discrepancy between the real *d* value and its determination is not significantly large. Therefore, *d* is obtained as shown in equation 6.1.

$$d(\%) = \begin{cases} \frac{a-c}{r-c} \times 100, & if \ \ a > c \\ \frac{a-c}{c-l} \times 100, & if \ \ a < c \end{cases} \qquad 6.1$$

where:

$$a = \frac{width_{image}}{2} ,$$

$$c \rightarrow C(x) = height_{image} ,$$

$$r \rightarrow R(x) = height_{image} ,$$

$$l \rightarrow L(x) = height_{image} .$$

The a, c and l values are directly obtained from the p matrix obtained in the detection of the road limits. Therefore equation 6.1 can be rearranged, considering the points returned from this matrix and the flags indicating if the lines are visible, as shown in equation 6.2.

$$d(\%) = \begin{cases} \frac{a - {p_1}^{center}}{{p_1}^{right} - {p_1}^{center}} \times 100, & if \ \ a > {p_1}^{center} \ \wedge \ RL = true \ \wedge \ CL = true \\ \frac{a - {p_1}^{center}}{{p_1}^{center} - {p_1}^{left}} \times 100, & if \ \ a < {p_1}^{center} \ \wedge \ LL = true \ \wedge \ CL = true \end{cases} \qquad 6.2$$

where:

$$RL \ : \ the \ flag \ indicating \ the \ presence \ of \ the \ right \ line$$

$$CL \ : \ the \ flag \ indicating \ the \ presence \ of \ central \ markings$$

$$LL \ : \ the \ flag \ indicating \ the \ presence \ of \ the \ left \ line$$

Filtering

The outputs of the vision system refer always to the frames received at the present time instant. Thus, it may happen that the values determined in certain time instant, are not the correct ones, preventing certain key procedures to behave correctly. For example, the

incorrect determination that some particular line of the track is not available prevents the proper insulation of the central region. In order to bridge this problem, the outputs are filtered through a low pass filter.

The filter computes the mean value of the variable over the last *N* samples received. In the case of binary variables, it is incremented a counter (up to *N*) whenever a sample is received with the *TRUE* logical value, and decremented (up to 0) otherwise. It is considered that the variable has a *TRUE* logical value if the counter value is greater than *N/2*. The binary values considered are: the $RL\,, CL\ and\ LL$ flags, and the presence of the crosswalk, denoted as CP.

As for the indirect output of lateral distance ($d(\%)$), its filtering procedure consists in storing the last N samples received in a queue data structure, and when filled up to N the mean value is computed. For every incoming value, the queue is popped, and the new value is pushed into the front of it. Its resultant value is subjected to a rate of change limiter, to improve the smoothness of the measurements. For that end, it compares the actual value with the previous one, and if their differences are above a certain threshold, only a maximum change of χ is allowed, as shown in equation 6.3, where $d_{ROC\ limiter}(\%)$ denotes the resultant value of d after the rate of change limiter filter.

$$d_{ROC\ limiter}(\%) = \begin{cases} \bar{d}_{N-1} + \chi\ , & \bar{d}_N - \bar{d}_{N-1} > \chi \\ \bar{d}_{N-1} - \chi\ , & \bar{d}_{N-1} - \bar{d}_N > \chi \end{cases} \qquad 6.3$$

where:

$\bar{d}_{N-1}$: $the\ value\ of\ \bar{d}\ for\ the\ N-1\ sample$

$\bar{d}_N$: $the\ present\ mean\ value\ of\ d$

χ : $the\ maximum\ change\ allowed$

The implementation of the filtering process and storage of the resulting values is in charge of the class CTRACK. At each main cycle of the application, the class updates the values as well as other issues to be discussed ahead.

6.2.1.2 - Sensors

Unlike the data coming by vision system, the sensors located on the vehicle are not subject to any signal filtering, as this task is already done in the lowest layers (either by the firmware of the vehicle, or by the control application).

As previously stated, the data coming from the control layer is transported to the decision layer through a protocol based on UDP sockets. The management of the data exchange on UDP is in charge of the singleton class, CUDP. A diagram of that class is shown in Fig. 6.6.

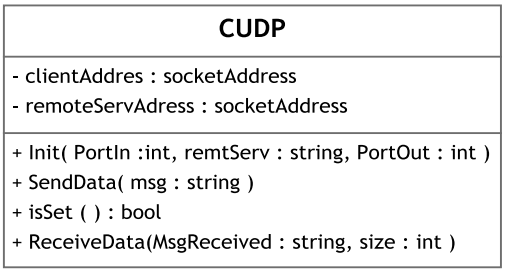


Fig. 6.6 An UML Class Diagram for the CUDP class

Thus, another class, CCAR makes requests to control layer through the CUDP class. The received data fills in a structure that defines the internal state of the vehicle. The sequence of actions performed when a refreshment of data is accomplished is as described in the scheme of Fig. 6.7.

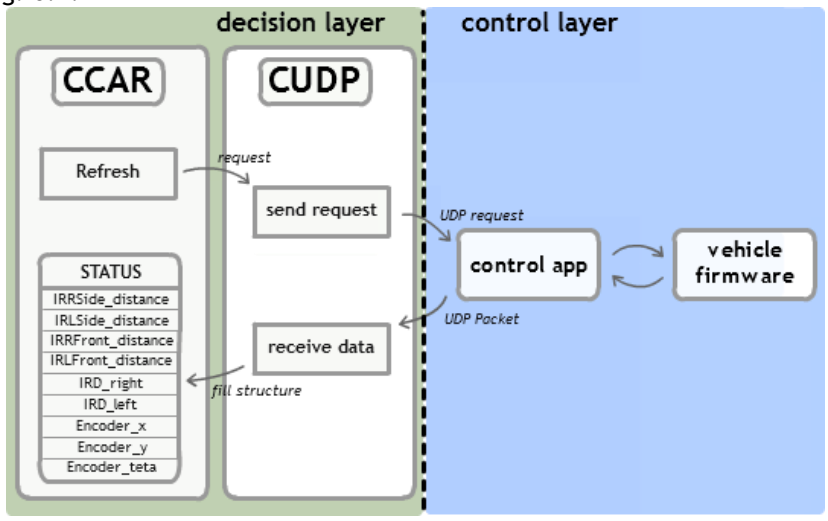


Fig. 6.7 Set of operations performed to update the vehicle state

6.2.2 - Localization on the Track

Through the information received by the sensory part of the system, it is possible to infer the relative position of the vehicle on the track. The approach followed creates a topological representation of the world, and follows two main considerations. The first is that the shape of the track is symmetrical on the crosswalk, i.e., once the vehicle crosses it, the track layout is the same regardless the direction followed. Consequently the approach followed to one side of the track is equivalent to its complementary side. The second consideration is that the location on the track is sectorial, i.e., it is not determined the precise location of the vehicle, but the sector the vehicle is travelling on. This is due to the fact that the navigation system was designed as much as possible for general navigation purposes, regardless of the track characteristics.

Accordingly, the track was split in 4 different sectors, as shown in Fig. 6.8.

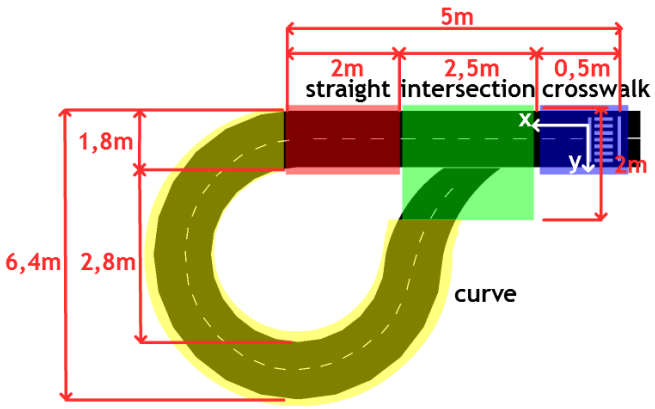


Fig. 6.8 Track sectors

The recognition of the current track sector is based on the metric characteristics of the track, depicted in Fig. 6.8. For that purpose the odometry system is used for this determination. Supposing the vehicle is located in the crosswalk, parallel to the track, and is moving forward into the intersection sector, the reference coordinate frame is that of Fig. 6.8, represented by the white x and y axis.

A state machine was implemented defining the transition conditions in the sectors based on the track metrics. This method was chosen aiming to reduce the accumulative odometry errors, increasing the performance of the estimator. This way, the odometry system is reset every time the vehicle enters a particular sector. This state machine is integrated in the class CTRACK, and the state machine evolves during the main cycle of the decision layer. The state machine diagram is shown in Fig. 6.9.

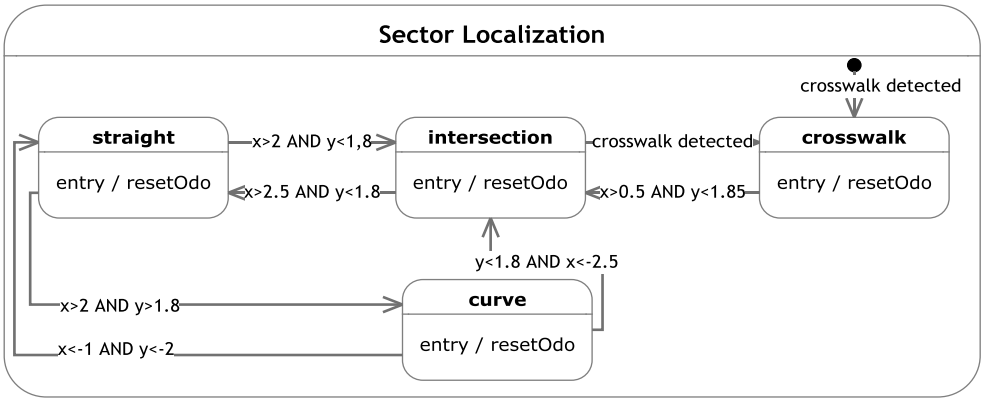


Fig. 6.9 Sector Localization State Machine

Note that this state machine is updated following the filtering process of the vision outputs.

6.3 - Planning

The planning phase determines what action the vehicle takes facing the world conditions. Therefore, it aims to determine the value imposed on the two variables on which the system operates: the speed and direction of the wheels.

The planning system is implemented in the CCAR class, which computes the reference values and sends its result directly to the CUDP class, which in turn sends the data to the control layer. To this end, the system is based on the information contained in the world perception stage, implemented in the CTRACK class. The direction of data flow is that as depicted in Fig. 6.10.

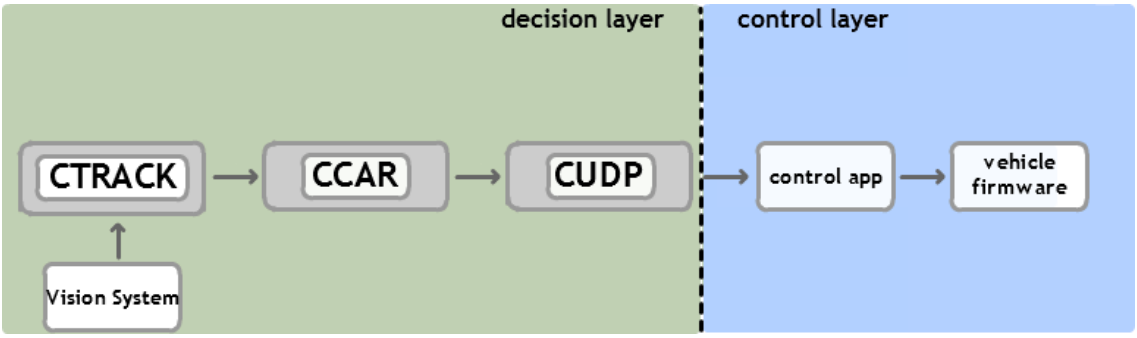


Fig. 6.10 Direction of data flow within the various components of the system used in the planning stage

6.3.1 - Steering Control

Since it is intended a solution invariant with the track, the steering control system developed aims to be invariant with the current section of the vehicle on the track. Thus, the approach followed is lane-oriented. This way, regardless of the current track sector, the control aims to keep all points of contact of the vehicle with the ground within the boundaries of a given lane.

The controller should, therefore act in the steering angle, based on current and desired lateral distances. This is therefore a closed loop controller, which acts on the plant - the vehicle, receiving feedback through the sensory part of the system. To this end, a controller was designed based on a proportional-derivative controller, which controls the steering angle of the vehicle based on the resultant error between the desired lateral position (d^*) and the actual lateral position (d). After computing the error (Δe), this result is subject to a rate-limiter in order to avoid abrupt changes, possibly arising from an erroneous evaluation of the sensory part, that would result in noise when computing the derivative. This corrected error (Δe^*), is then introduced to a PD controller, which performs the key task of the steering controller. An angle value is obtained (θ^*), and if outside a certain range, is re-adjusted through a saturation block, resulting a reference value of θ. The block diagram of this controller is depicted in Fig. 6.11.

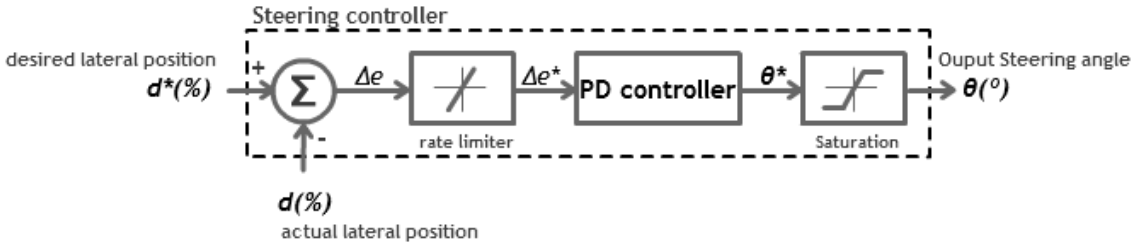


Fig. 6.11 Block diagram of the steering controller implemented

Due to the high complexity of the system, and its strong non-linear characteristics, the determination of the dynamic model of the entire system is very difficult. Thus, the tuning of this controller was done empirically and gradually. This procedure started by calibrating only the PD gains, ignoring the other blocks. The features that are intended to result from this PD control are stable steering with minimum overshoot, maintaining a low steady-state error.

The expression of an ideal PID controller in its continuous form is given by equation 6.4, where T_i, and T_d, denotes the integral and derivative time; K_p the proportional gain, and e the error. Discarding the integral term due to the nature of the process and discretizing the controller yields equation 6.5 that was computed in the system.

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{d}{dt}e(t)\right) \qquad 6.4$$

$$u[k] = K_p e[k] + K_d(e[k] - e[k-1]) \qquad 6.5$$

where:

$$K_d = \frac{K_p T_d}{T}$$

$$T \;:\; sampling\ time$$

The rate-limiter was computed in a similar way to that shown in equation 6.3. As to the saturation block its output is that as shown in equation 6.6.

$$\theta = \begin{cases} \alpha\,, & \theta^* > \alpha \\ -\alpha\,, & \theta^* < -\alpha \\ \theta^*\,, & otherwise \end{cases} \qquad 6.6$$

6.3.1.1 - Controller parameters

Although it is intended to have a generic controller, it was experimentally proven that the system robustness is improved if different parameters are assigned depending on the action to be accomplished. Thus three different situations were considered: straight, curve and lane changing. For that purpose three similar functions were implemented, which differ mainly in the controller parameters.

Straight

The straight situation, where the vehicle moves in the middle of one lane and parallel to it, is the most comfortable situation for the correct effectiveness of the controller. This is

due to the fact that this is the situation where the observation of the d variable is best attained, with greater precision and smoothness. Therefore, the k_p value of the PD controller is slightly larger than in other situations, reducing the rise time. It was taken into account that such an increase would not introduce a large overshoot. Regarding the error change of rate, it was enabled a greater margin with respect to other situations, due to the higher degree of confidence of the observed variable.

Curve

In the situation where the vehicle is curving a more sensitive control is needed. The estimation of the d variable has a slightly lower degree of confidence than the previous situation. Therefore, in order to reduce abrupt changes in the steering angle, the K_p value is decreased, as well as the maximum rate of change of the error and the maximum output value.

Lane changing

In the special situation of lane changing, it is intended that the change takes place as smoothly as possible. Therefore, the maximum output allowed is decreased sharply. This value was set so that the vehicle can still make the change in the worst case, which is during the curve. The saturation parameter was also reduced drastically, as well as the maximum rate of change. As for the K_d parameter, this was increased slightly to increase the sensitivity of the system when it approaches the set point.

When performing lane changing, it is important for the higher level controller to identify when the lane changing is completed. For that purpose the implemented function returns a flag with that notice. It is considered that the lane changing is completed when current error (Δe) is below $\pm$ 5º.

Table 6.1 summarizes the controller parameters, where K_p and K_d denote respectively the proportional and derivative gains, α the saturation threshold and χ the absolute maximum rate of change allowed.

Table 6.1 – Steering controller parameters

Situation	K_p	K_d	α(º)	χ($^{\circ-1}$)
Straight	1,7	0,01	30	8
Curve	1,2	0,01	25	5
Change lane	0,7	0,05	15	3

The diagram of Fig. 6.12 outlines the functions implemented in the CCAR class, to perform the steering control on the vehicle.

CCAR
- previous_error : int - STATUS : state
+ MaintainLane_STRAIGHT(setpoint : int) + MaintainLane_CURVE(setpoint : int) + ChangeLane(setpoint :int) : bool + SetSteering(angle : int)

Fig. 6.12 The attributes and functions of the CCAR class in order to implement steering control

6.3.1.2 - Backup controller

In order to avoid situations where the controller or the measuring system fails, causing the vehicle to follow a trajectory that sends it outside the track, a backup controller was implemented. The occurrence of such situation is anticipated by the infrared sensors that detect white markings. Fig. 6.13 illustrates an example where the right sensor detects the right lateral limit.

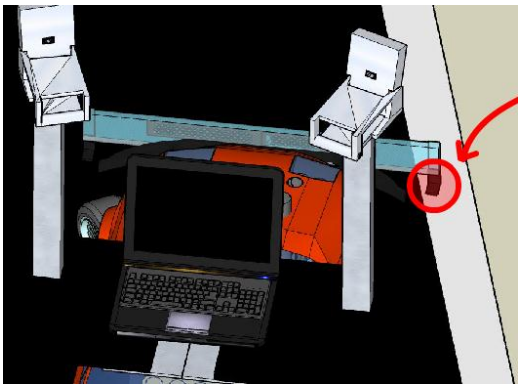


Fig. 6.13 The right limit being detected the infrared sensor

This controller is placed before the main controller in order to ascertain whether such a situation occurs. In normal situations, it allows the main controller to operate. In abnormal situations, the main controller is not actuated, and while this anomaly is verified, the steering angle of the wheels is determined by this controller. This setup in depicted in Fig. 6.14

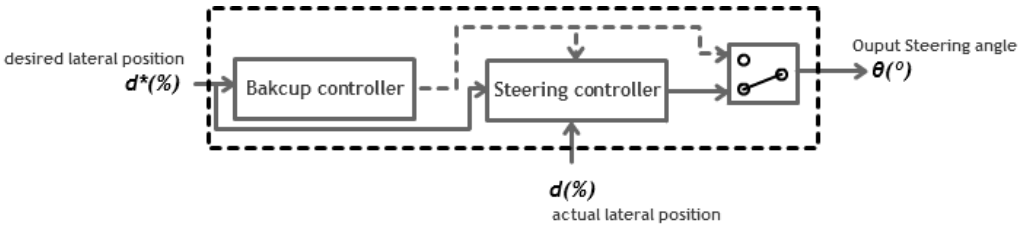


Fig. 6.14 The backup controller used in steering control

The controller is operated only in situations where the vehicle is on a curve or in a straight. Lane changing is not considered at this level, since when it is performing lane

changing, the sensors might detect the center line, and make a wrong interpretation. A flag indicating that a sensor has detected a line, lies in the *STATUS* structure, of the *CCAR* class, called *IRD_right* or *IRD_left* (see Fig. 6.7).

Thus, in abnormal situations, the behavior of this controller is as shown in equation 6.7. Note that the angle of 40º reflects the maximum angle allowed physically.

$$\theta = \begin{cases} -40\ , & d^* < 0\ \wedge\ IRD\_right \\ +40\ , & d^* > 0\ \wedge\ IRD\_left \\ (normal\ control)\ , & otherwise \end{cases} \qquad 6.7$$

6.3.2 - Speed Control

Along the navigation, due to different circumstances, it is necessary to make an adjustment in the current speed. This need comes from the fact that when performing complex tasks (such as performing a curve), robust control is more easily accomplished at a lower speed.

Thus, there is an assignment of various speed values to different sectors / situations on the track. The considered speed levels for the different sectors or situations were as follows: straight, intersection, crosswalk, curve and obstacle.

So that these values are easily changed, the values are stored in an *ini* file, which also includes other settings. This file is read at application startup, and the settings are loaded onto the environment. Fig. 6.16 shows the contents of the file when it comes to speed settings.

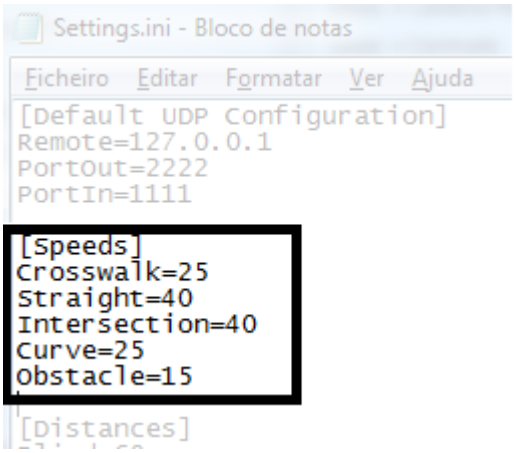


Fig. 6.15 The settings file with the assigned speeds for different situations

6.3.3 - Sequence of actions

Given the specificities of the competition, there is a sequence of a well-defined set of actions to take, in accordance with the information provided by the upper signaling panels. In the context of this thesis, two main actions were considered: go straight ahead on the intersection and turn left on the intersection. Along with these main actions, obstacle avoidance is also performed, if necessary.

A trial starts with the vehicle laid on the right lane, on the first horizontal line of the crosswalk, as illustrated in Fig. 6.16. Then the system is initiated and starts the search / detection of the signal in the TFT display. When the signal is detected, the vehicle makes a sequence of actions ordered by that signal.

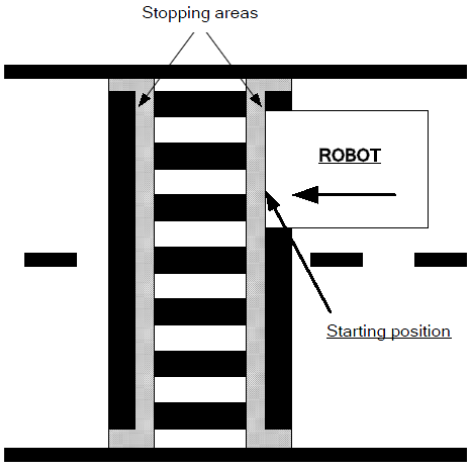


Fig. 6.16 The starting position of the vehicle [21]

A signal appears on the screen long before the vehicle is on the crosswalk itself. Hence, the system anticipates the detection of the signal, while the task being performed is still running. Therefore, once determined the signal, the system turns off the upper camera, and the search is disabled. Search is resumed when the vehicle is approaching the start of the track. Once the signal display is defined, the code of it is stored, and the sequence of the previous order is terminated when the vehicle is back in the starting position.

The high level state machine is depicted in Fig. 6.17.

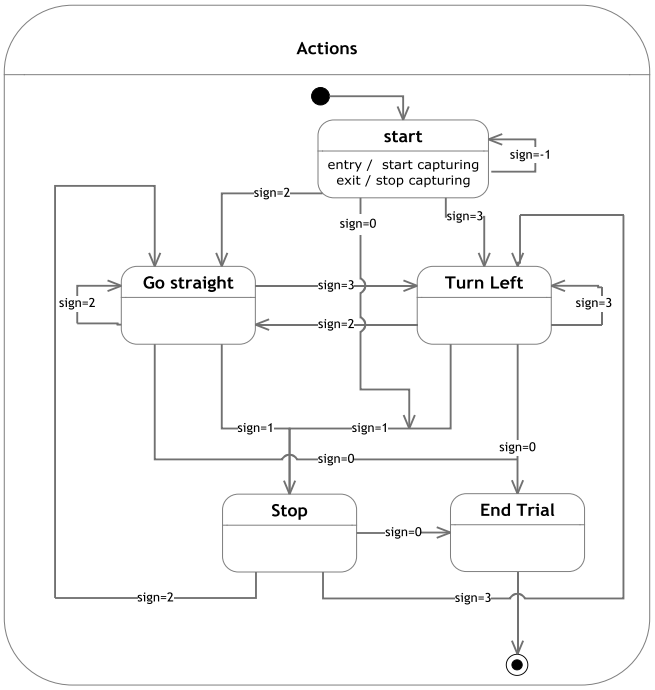


Fig. 6.17 UML representation of the state machine that implements the sequence of actions to be performed during a trial

Straight ahead on the intersection

The image of Fig. 6.18 illustrates the trajectory considered for this action.

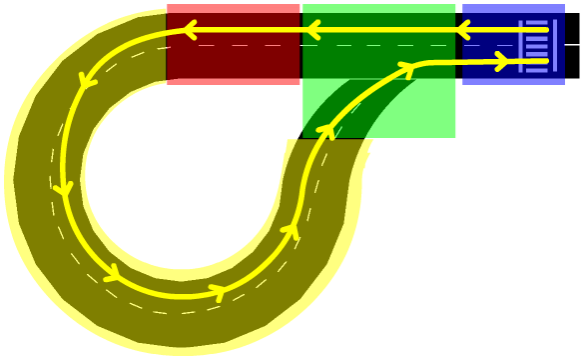


Fig. 6.18 Trajectory used on the Straight ahead order

Assuming that the vehicle always starts in a position perfectly parallel to the limits of the road, the sequence of actions begins ordering a steering angle of 0 °, until the area of the crosswalk is overcome. At this point, the steering control system with the straight variant is used.

In order to reduce the time it takes the vehicle to make the curve, it is always assumed that the curve is travelled by the inner radius, covering thus a shorter distance. To this end, two lane changes must be performed: the first takes place when the vehicle meets the curve, making a shift to the left lane. Then when the curve is being finalized, and the intersection arises, the vehicle switches to the right lane in order to fulfill the criterion of finishing always in the same position. At this point, two parallel set of actions are performed: navigation, and detection of the TFT display.

When the vehicle approaches the intersection, the middle line will inevitably fail to be visible (in the field of view of the cameras). In this situation, it is no longer possible to determine a lateral distance. As the vehicle is making a lane change to right, it is considered that the steering angle remains the same until the central line if visible again. Note that there is virtually no risk of the vehicle to leave inadvertently the track, due to limitations in the “lane change" parameters of the steering control. Also this situation in which the vehicle navigates without any reference point is very short is time. Once the central markings are visible again, the vehicle resumes the navigation, until it reaches the starting position. The starting position is detected through the infrared sensors, as shown in Fig. 6.19.

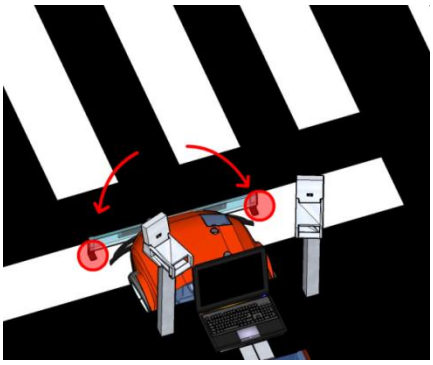


Fig. 6.19 Infrared sensors detecting the starting position

At the same time, as the vehicle performs these latter steps, the system starts the search for the TFT display. This task is finished only when it is determined which is the current sign. Eventually, if it is not possible to determine the signal before reaching the starting position, the vehicle stops and the system waits until a signal is obtained. Once obtained, the signal code is returned, to start another set of actions.

The state machine that implements this set of actions is shown in Fig. 6.20.

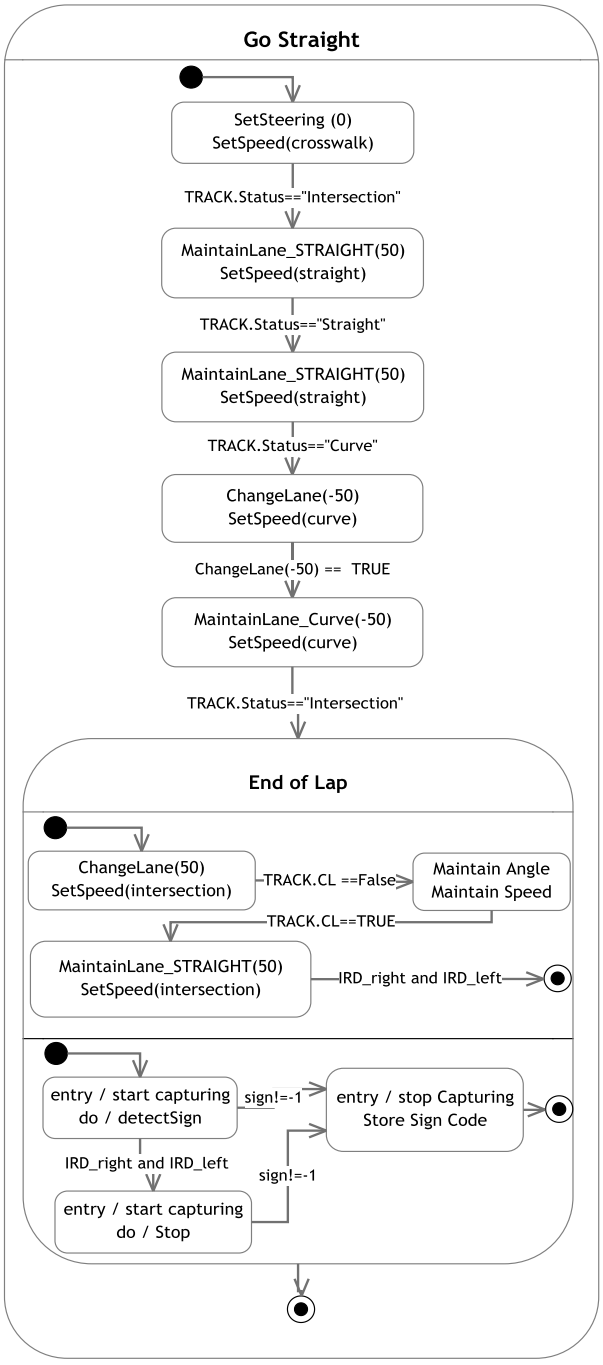


Fig. 6.20 The state machine implementation of the Go Straight order

6.3.3.1 - Turn left on the intersection

The proposed trajectory of the vehicle in the situation where it was ordered a left turn at the intersection, is represented in Fig. 6.21. As shown, the trajectory starts in the right lane of the track, then the smaller radius curve is travelled and it finishes in the right lane, back to the starting point.

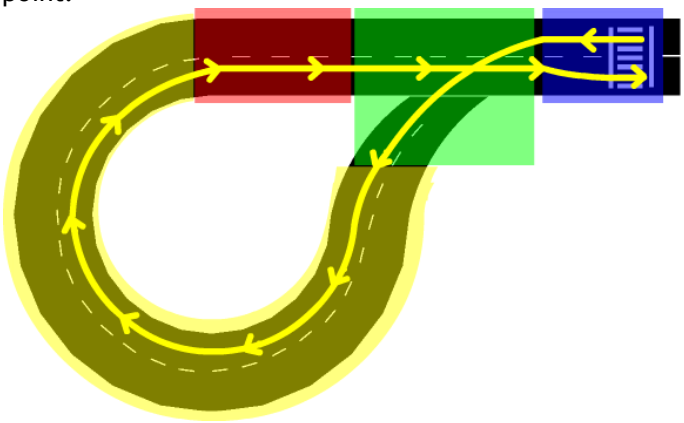


Fig. 6.21 The ideal trajectory on the Turn Left order

The approach followed in the implementation of this procedure was similar to that described above. The vehicle trajectory starts with a zero steering angle, until the crosswalk area is overcome. Then it starts curving with a fixed value of -15°, until the vehicle has reached the curve sector. At this point, slight deviations from the trajectory are adjusted, through the change lane function, so that it is done smoothly. Then the rest of the curving is normally carried out.

Once at the straight sector the vehicle leans slightly to the right (20%). This is because the sector that follows, intersection, the right line is no longer visible. So, by leaning closer to the central line, the chance of visualizing the left line is increased. Therefore, it is possible to estimate a relative lateral position. If the left line is still not detectable in the intersection sector, then the same steering angle is maintained until the right line is found and the steering control takes over again. At this stage, the search and detection of the TFT is carried out in a parallel task.

This task finishes then in an analogous way to that shown in the previous section. The implementation of this state machine is shown in Fig. 6.22.

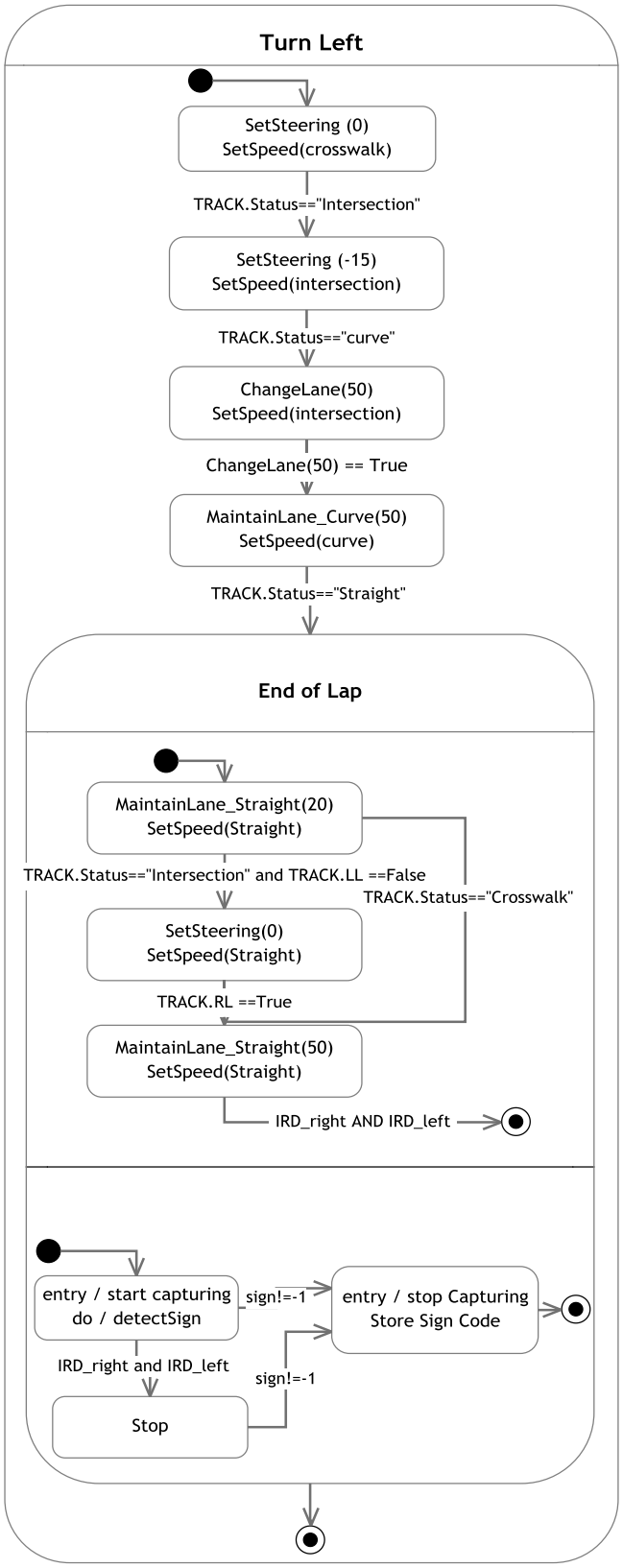


Fig. 6.22 The state machine implementation for the Turn left order

6.3.3.2 - Stop

For illustrative purposes, Fig. 6.23 shows the state machine implementation of the Stop action. It solely orders the vehicle to stop, while waiting for a signal to be available.

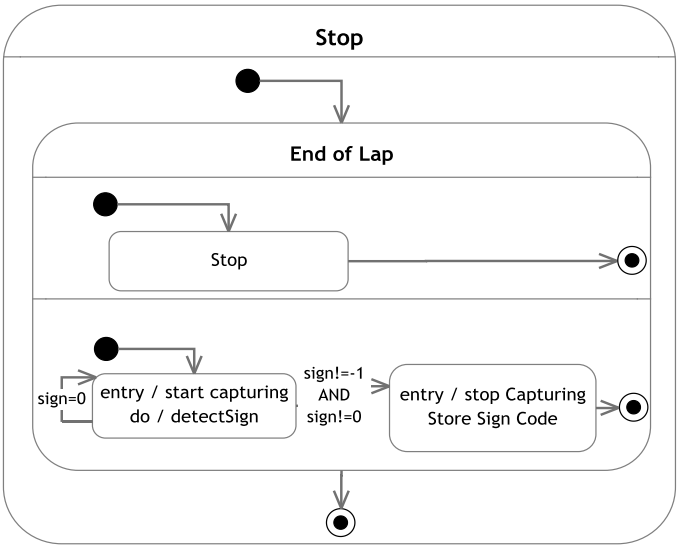


Fig. 6.23 The state machine implementation for the Stop action

6.3.3.3 - Obstacle Avoidance

The determination of the existence of obstacles, and further avoidance is implemented in the designed system, through the data returned by the physical sensors. An obstacle is considered as an object with known volume that occupies a set of points that were pre-destined to be part of the normal trajectory.Fig. 6.24 illustrates such a situation.

In the main program cycle, the CCAR class, besides an update of the state variables of the vehicle, also carries an analysis of the values coming from the infrared sensors that detect distances.

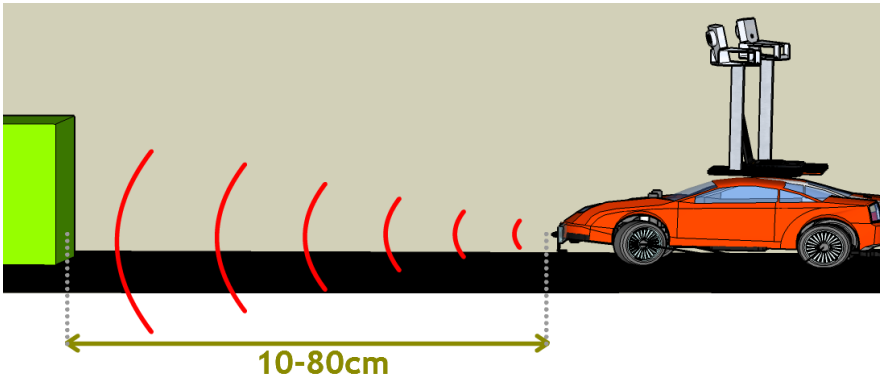


Fig. 6.24 The sensors detecting an obstacle

The sensors return distance values between 10 and 80 cm. Therefore, it is considered that an obstacle indeed exists, if during a certain number of samples obtained, the detected distance is in this range of values. The placement of the obstacle is assumed to be within the space defined by a lane. In case an obstacle exists, one of the sensors (left, right or possibly

both) will detect the presence of the obstacle in advance. The obstacle consists in rectangular box with a 60cm square base, with a minimum height of 20cm.

As performed for the rest of the sensors, these sensors are also subjected to a filter that aims to detect accurately if an obstacle is present, preventing false obstacles to be detected. The filter consists in incrementing a counter each time a value is returned from any sensors that is within its operational limits. After N samples received, if the counter value is greater than $^{N}/_{2}$then it is considered that an obstacle indeed exists.

Once the obstacle is detected, the current trajectory of the vehicle is diverted, and when the obstacle no longer obstructs the path, the vehicle returns to its previous trajectory. This procedure is illustrated in Fig. 6.25.

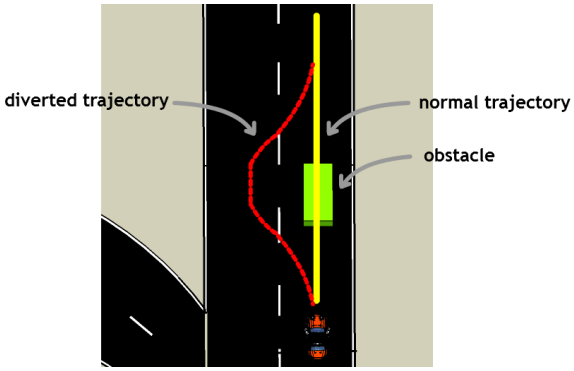


Fig. 6.25 The followed procedure when an obstacle is detected

To this end, alongside with the actions described above, another state machine performs this checking procedure. If an obstacle is detected, the speed is decreased, the current values of the odemetry system are stored and the vehicle changes to the opposite lane. When the lane change is completed, the vehicle maintains itself on the current lane until it has traveled a sufficient distance that allows it to resume the planned trajectory. This distance is considered as the distance travelled from the moment the vehicle detected the object, to the current point in space. Its value is obtained by computing the modulus of the vector joining these two points, and if above a threshold value, the vehicle resumes its normal trajectory. Fig. 6.26 shows a situation where the computed distance is above a threshold value, denoted by return point. This threshold value was defined as 60 cm (length of the squared base of the obstacle) + 100 cm.

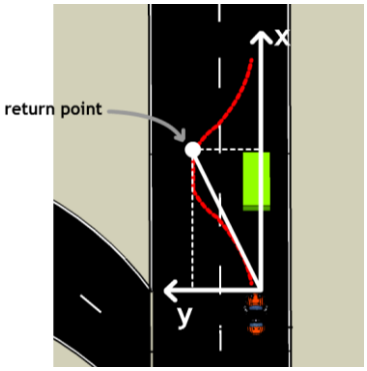


Fig. 6.26 The return point, where the vehicle resumes the normal trajectory

The state machine implementation of this system is illustrated in Fig. 6.27.

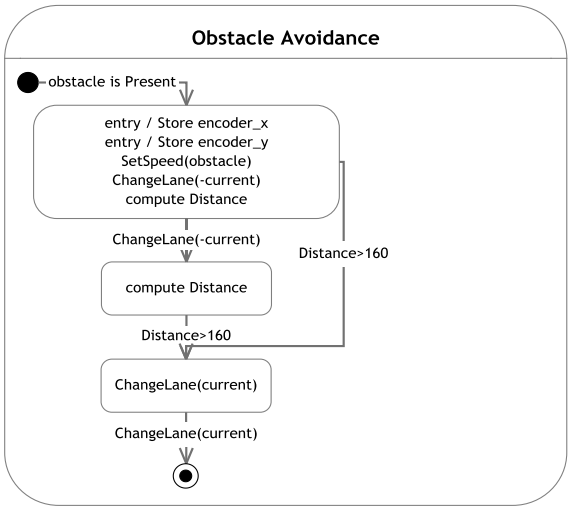


Fig. 6.27 The state machine implementation of the obstacle avoidance system

6.4 - Chapter conclusions

This chapter discussed the methodology used for navigation. It begins by showing how the interpretation of sensory data is performed, to determine appropriate and relevant measures needed for high-level control. Thereafter, the method of locating the vehicle on the track is presented. Then, the steering controller developed is presented, as well as the speed controller. The chapter concludes by demonstrating the sequence of steps to be undertaken in the context of the competition. This last section includes the methodology used for obstacle avoidance, which makes use of two low-cost infrared sensors to determine the distance.

Chapter 7

Experimental Results and Discussion

In order to measure somehow the quality of the developed system, an accuracy analysis of measurements was performed, as well as the system response to different world conditions, when it is desired to reach a certain state.

For this purpose two sets of tests were held: computer vision sub-system tests (measurements) and navigation tests (system response).

7.1 - Computer Vision Subsystem

7.1.1 - Road Limits

Regarding the detection of the road limits, samples were taken with the vehicle going through all the parts of the track, examining the effectiveness of the method for the various situations. In this context, no filtering processes were applied at all. This analysis was made in a subjective way through visual inspection, setting a false detection when the resultant vector defines a line clearly defective, which would seriously compromise the navigation system if no filtering was applied.

It should be noted that, the configuration parameters were the same for both cameras (contrast, illumination and saturation), so that no disparity exists between both parts of the fused image.

Table 7.1 shows the obtained results for each section of the track, showing the number of frames where the respective line was clearly detected, and its percentage of the total number of frames captured. One can notice that when approaching the intersection, the left or right line fails to be detected, since the respective line disappears from the image as the vehicle moves forward, therefore the analysis did not contemplate the intersection sector. Another fact demonstrated was that, as the vehicle approaches the crosswalk, it is harder for the system to properly detect the central line. This is due to the fact that the central line loses its continuity on the crosswalk, and in the immediate surroundings, the presence of the crosswalk affects the determination of the central line. Nevertheless, the right and left lines are still properly detected, hence not affecting the navigation.

Table 7.1 – Obtained results for the road limits block

Situation		Left	Central	Right	Total Frames
Crosswalk	Success rate	821 **100%**	648 **78.9%**	812 **99.9%**	821
straight	Success %	1701 98.7%	1670 96.9%	1690 98.1%	1723
Curve	Left curve Success %	2660 95.0%	2650 94.8%	2690 96.1%	2800

7.1.2 - Crosswalk

The validation of the crosswalk detection algorithm consisted in placing the vehicle close to the crosswalk in order to verify the effectiveness of the method at various distances and angles, as illustrated in Fig. 7.1. It was chosen to place the vehicle on the right lane, since this situation is more unfavorable when compared to the center, where the visible area is greater.

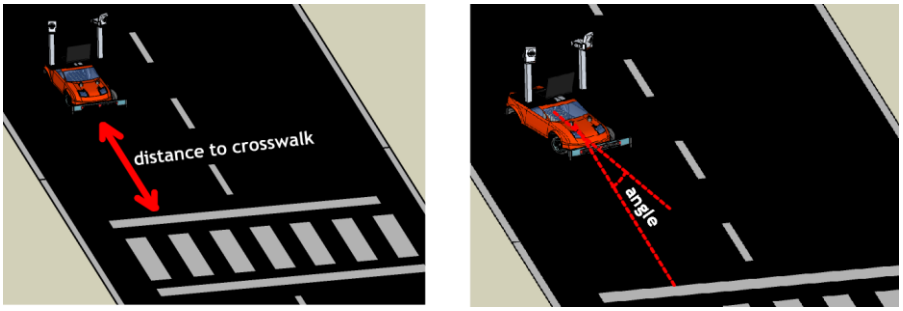


Fig. 7.1 Representation of the validation tests performed during the validation of the crosswalk detection algorithm.

The experimental setup started with running the algorithm with the car placed at the minimum distance where the crosswalk is visible, testing three angle scenarios: 0 degrees (vehicle placed perpendicular to the crosswalk), 25 degrees and 45 degrees. This procedure was repeated every 10 cm until the crosswalk was no longer detected. The collected data summarized in Table 1.

Table 7.2 - Crosswalk detection validation

distance		Angle 0º	25º	45º
	10 cm	48	196	49
	Success rate	24%	98%	24.50%
	20 cm	200	23	56
	Success rate	100%	11.50%	28%
	30 cm	200	111	58
	Success rate	100%	55.50%	28%
	40 cm	200	175	200
	Success rate	100%	87.50%	100%
	50 cm	200	199	200
	Success rate	100%	99.50%	100%
	60 cm	200	182	195
	Success rate	100%	91%	97.50%
	70 cm	200	200	186
	Success rate	100%	100%	93%
	80 cm	200	200	36
	Success rate	100%	100%	18%
	90 cm	200	58	12
	Success rate	100%	29%	6%
	100 cm	34	0	0
	Success rate	17%	0%	0%

With these tests it was concluded that the detection of the presence of the crosswalk is successful if the captured area from the cameras encompass totally the crosswalk, or most part of it. In fact, reducing the number of significant trapezoids, or reducing the similarity factor for a candidate trapezoid with the groups, results in detecting the crosswalk in a greater number of situations, including higher angles. However, this would come with the cost of detecting false crosswalks as the vehicle moves throughout the track.

At the shortest distance and at 0º, the detection is not successful in most of the samples, as the positions of the cameras do not allow the full view of the crosswalk. Increasing the angle to 15º, a larger area of the crosswalk is captured, leading to a successful detection. However, at 45º the detection is not successful, due to the lack of information captured by the cameras. Although the detection can be made successfully at 10cm with an angle of 25º, this does not happen at 20cm, due to the lack of provided information, and the same goes, as

expected to 45 °. However at 0 °, the crosswalk is entirely viewable in its length, thus being detected. This pattern is seen at short distances, and from 40 cm, the crosswalk is clearly detected in any situations, up to 70 cm. Above this value the same pattern is found for smaller distances, and the detection starts failing to be successful as the angle increases.

7.1.3 - Upper Signaling panels

The tests conducted to validate this algorithm were similar to those shown in the previous section. It was concluded that the algorithm determined the true sign in 100% of the cases found. The algorithm proved to work over a distance of 120cm, provided that the vehicle is aligned with a maximum angle of 40 ° angle with the panel. For longer distances and angles, the algorithm fails to detect the region corresponding to the TFT display.

7.2 - Car Navigation

The vehicle navigation tests were performed in the robotic soccer field of the department of electrical and computer engineering at FEUP. Therefore, a track was placed on the field, in which the curve had a smaller radius of curvature, but apart from that, with the exact dimensions required by the competition.

In order to obtain the position of the vehicle in the track, the Ubisense RTLS (Real-Time Location Systems) [53] system was used. This system is based on tags that transmit a UWB (Ultra Wideband) signal. The transmitted signal is received by sensors, determining a set of possible points for the location of the tag. By using multiple sensors, a triangulation technique is used to solve for the position of the tag in world coordinates.

This way, to achieve the most accurate position of the vehicle in the track, four tags were placed in such a way that their geometric center represents the center of mass of the vehicle. This arrangement is shown in Fig. 7.2-c). Regarding the sensors, these where placed in the four corners of the field, so that the covered area is maximum (see Fig. 7.2-a)). The sensors and tags are showed in detail in Fig. 7.2-b) and Fig. 7.2-d) respectively.

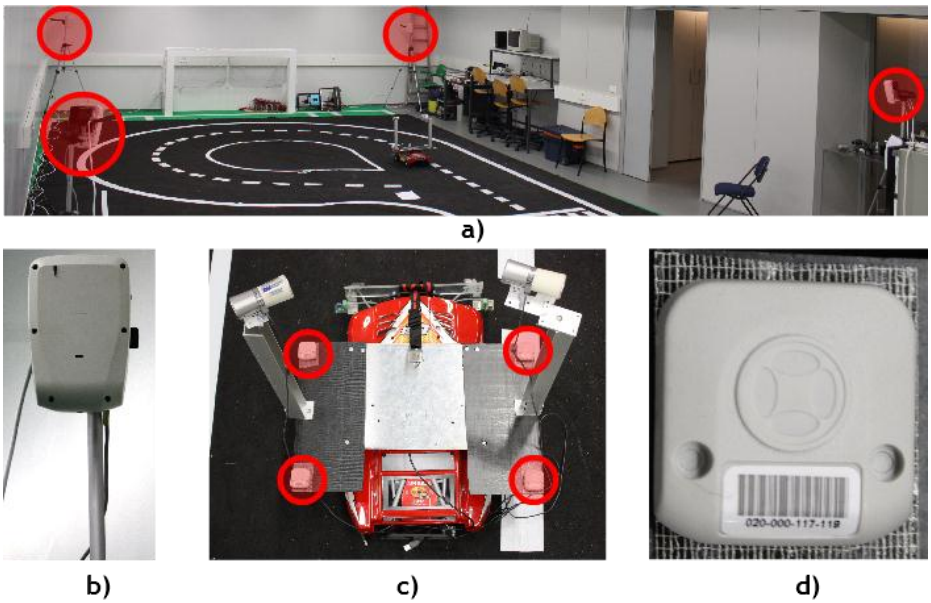


Fig. 7.2 The Ubisense setup used to validate the designed platform.

The output of the system is a data set where each package represents information related with a specific tag, namely: tag ID, timestamp and its 3D coordinates.

A track mapping was firstly performed, in order to reduce the calibration errors during the system setup and the human-introduced errors while the physical markings were placed on the track. For this purpose, the car was scrolled through the markings manually. The vehicle was driven remotely via another PC, through the implemented protocol over the UDP sockets. This way, the introduced disturbances in the Ubisense setup were minimal, as the presence of human bodies in the experiment affects the results. The returned data packets do not meet a pre-defined order with respect to the tag IDs and the sampling period oscillates around the mean value of 15ms. This way, the center of mass is computed whenever a set of four different tags are received. These resultant means are subject to a non-linear median filter within a window of 300 samples. These filtered samples were then plotted, resulting in three curves, representing the outer, the central and the inner markings.

In the context of the tests themselves, a trajectory was defined without any lane changes, or intersection negotiations, so one can infer about the actual quality of following a predefined path. Thus, the defined route was based on following the right lane along the straight and consequently the curve, stopping when the intersection was reached. This trajectory is defined thus, as the mean distance between the outer, and the central markings curve throughout the track. Based on the Ubisense data, an “ideal” trajectory was then defined. This curve is represented in the plot of Fig. 7.3 where the blue curve represents the outer markings, the red curve the central markings, the green curve the inner markings, and the black curve represents the computation of the ideal trajectory.

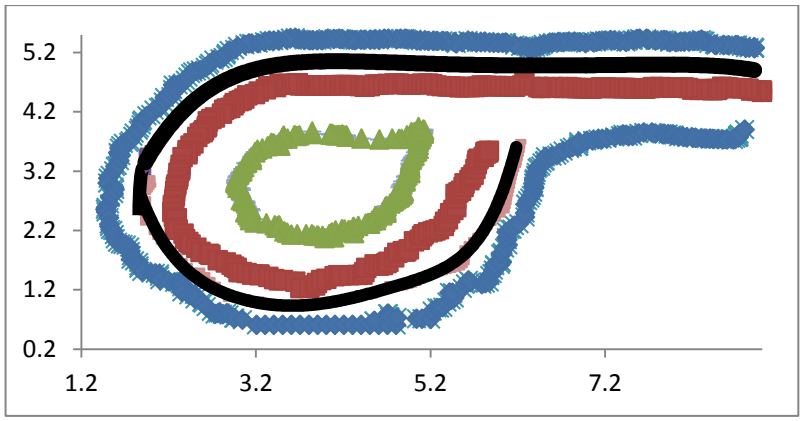


Fig. 7.3 The track mapping returned by the Ubisense system, and the resultant ideal trajectory

System performance is dependent on many variables: initial position, rate limiters, saturation thresholds, PD controller gains, desired speed and sampling time as well as environment conditions, such as lighting. However, the combination of all the different parameters is impractical for testing purposes. Thus it was chosen to vary the parameter that clearly has the most influence on the ability of the vehicle to reach the desired state while keeping the other parameters constant. This parameter was found to be the number of samples considered in the process of filtering the lateral distance (d). This variation introduces on the one hand a smoother trajectory as the accumulated number of samples increases, but on the other hand, introduces a larger delay of system response, which may be critical especially in the curve.

The sampling rate (frequency of execution of the main cycle) value was set to 10Hz, and the number of samples considered in the distance filtering were 2, 3 and 5. The limit of five samples was chosen as above this value, the error increased drastically.

For each filtering scenario, four different speed profiles were considered to the straight and curve sections. For every situation five tests were performed, to ensure statistical consistency.

The Ubisense system has an accuracy of about 50cm, and the samples are largely corrupted by noise. Some parts of the track show systematically corrupted samples, especially in the lower left corner, highlighted in Fig. 7.4. This effect may be explained by the proximity to a sensor in this area of the track.

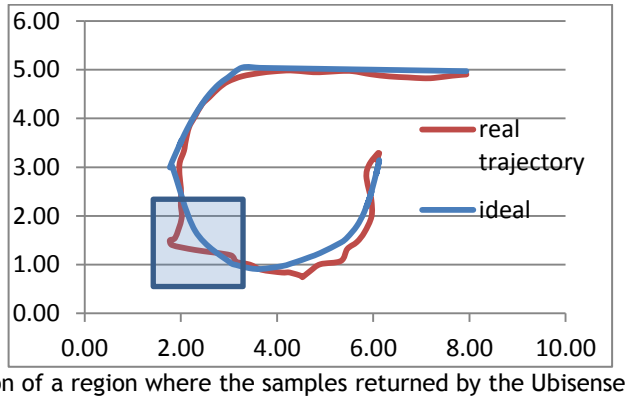


Fig. 7.4 Location of a region where the samples returned by the Ubisense where systematically corrupted by noise.

Nevertheless, the Ubisense system, returns a good approximation of the vehicle behavior, so that an analysis can be performed with some degree of confidence, thus enabling the determination of the best operation scenario according to the test parameters.

Table 7.3 shows the resultant standard deviation of the error for each scenario, which corresponds to the average value after 5 tests. Appendix A shows the trajectory followed for each test situation, as well as their statistical analysis.

Table 7.3 Different testing scenarios for the car navigation and the resultant standard deviation

Samples	Speed (cm/s)		Standard Deviation (cm)
	Straight	Curve	
2	25	15	16.50
	35	25	14.20
	50	35	11.50
	50	50	13.60
3	25	15	9.38
	35	25	4.02
	50	35	7.15
	50	50	16.57
5	25	15	10.6
	35	25	12.44
	50	35	5.93
	50	50	6.44

Based on the results obtained, it was found that filter performance is quite variable with the chosen speed, especially noticeable in the curved region.

For the purposes of participating in the autonomous driving competition, the most interesting situation to analyze is where higher speeds are attained. It was found that the best scenario is obtained in the situation in which 5 samples are considered in the sliding average filter, because although the settling time increases slightly, a smoother path is obtained due to an increased robustness to noisy measurements. With a smaller number of samples, despite the decreasing of response times, there are major fluctuations in the trajectory, increasing the overall discrepancy. For this reason at lower speeds, a smaller number of samples might be considered, since as the vehicle moves slower, the response time when facing a discrepancy in its ideal trajectory path is greater with a larger number of samples. These set of tests also demonstrated the highly nonlinear characteristic of the system.

Throughout the tests, a vehicle data log was created, where the measurements and the outputs of speed and steering were stored for each control cycle. In figure Fig. 7.5 three illustrative cases are shown, where the blue graph represents the measured distance over time, and the red graph the output of the filter. As seen, as the number of samples increases, the variation is smoother and the "peaks" are attenuated.

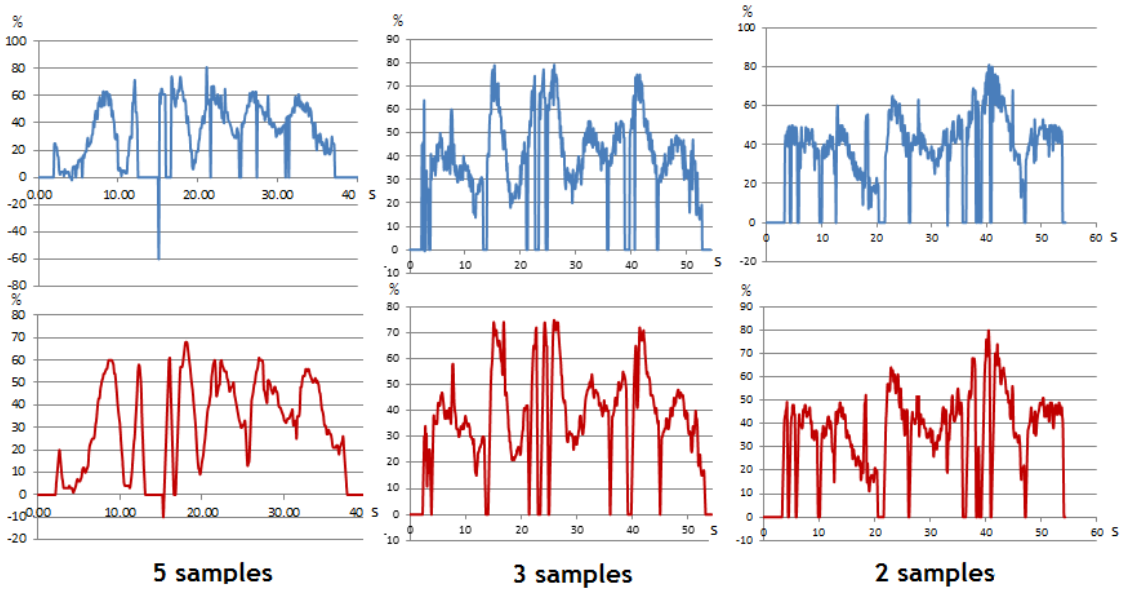


Fig. 7.5 The distance filter applied in three different scenarios. The top images show the determined raw data. The bottom images show the respective data with the sliding average filter applied

Another interesting aspect to analyze is the behavior of the steering controller with respect to the inputted lateral distance. Fig. 7.6 shows the steering controller response (bottom chart) to the inputs shown in the upper chart. This plot also highlights the special situation where the backup controller takes place, ordering the extreme angle of -40°.

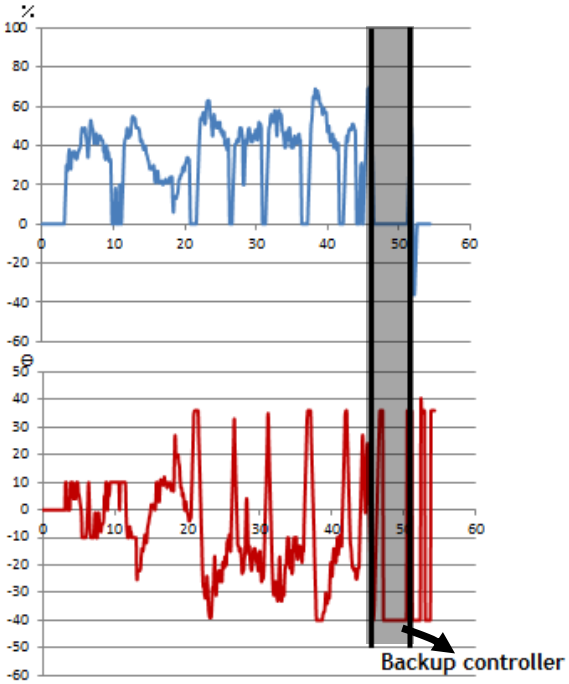


Fig. 7.6 The steering controller response (bottom) to the filtered lateral distance input(upper)

The vehicle log also included the odometry state variables along the track and it was found a large discrepancy between the position returned by this system compared to the "real" Ubisense position. Note that throughout this logging, no resets were performed to the odometry.

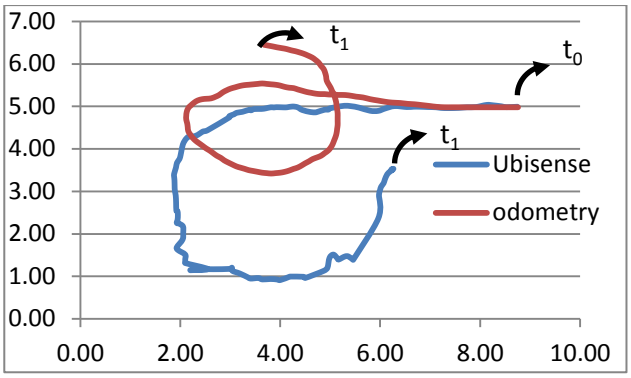


Fig. 7.7 The odometry system compared to the "real" position of the vehicle returned by Ubisense

This effect, although not considered in the context of this thesis, but in [1], can be explained by the fact that the current angle of the wheels is not estimated or determined. The implemented odometry system assumes that the steering system has unitary gain and no delay, thus assuming that the current steering angle is that just ordered. In fact, as the speed of the vehicle decreases, it is harder for servo motor to turn the wheels, hence increasing the delay time. Neglecting this effect may cause massive errors, which is exemplified in the chart of Fig. 7.7, where initially the odometry system follows a similar path to the correct position of the vehicle, but the cumulative error deviates it from the correct path progressively, resulting in the end in a massive displacement with respect to the true position.

It is also important to analyze the evolution of the distance variable, with respect to the vehicle observation and Ubisense. For this purpose, the values returned from Ubisense were re-scaled to [-100%,100%] and its comparison is illustrated for an exemplary case in Fig. 7.8. Note that the distance values observed by the vehicle were not filtered. As can be seen, both plots have a similar shape and trends, although there are some obvious points of discontinuity. The nature of these discrepancies was attributed to the error on the one hand resultant to a poor vehicle evaluation, but also and especially to the Ubisense system.

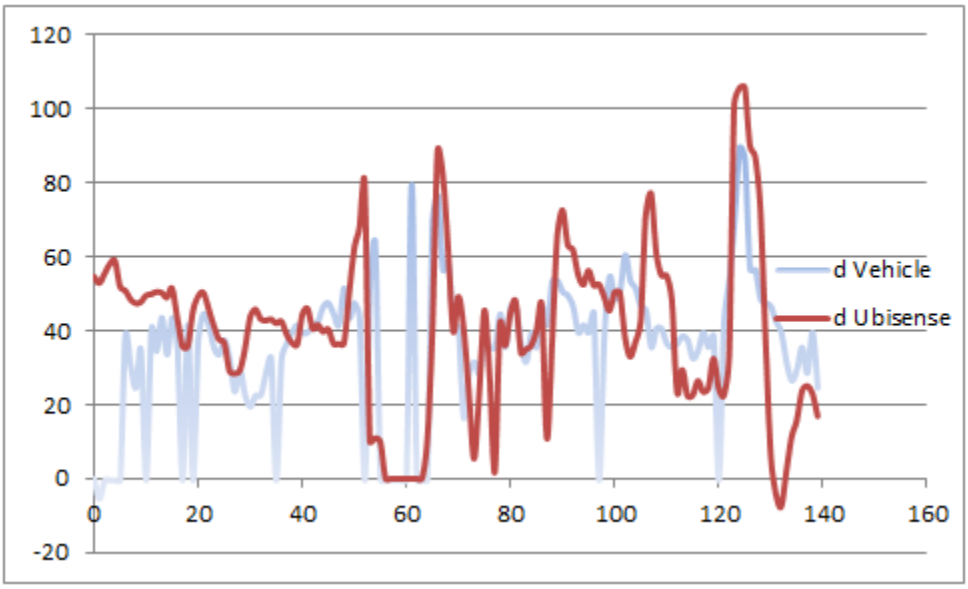


Fig. 7.8 The real lateral distance compared to that computed by the vehicle controller

7.3 - Performance measurement

The scope of the competition is to achieve the pre-defined route in the shortest time possible. Thus, the average time for the various trials was registered. Since the platform used in the tests covers only one half of the real track, and the radius of curvature is smaller, (completing a curve distance of about 7 meters, instead of about 13 meters on the real track), an estimated total lap time was calculated. These results are shown in Table 7.4, only for illustrative purposes.

Table 7.4 – Performance analysis of the vehicle using various speeds

Speed (cm/s)		Time	Estimated Lap
Straight	Curve	(s)	time (s)
25	15	90.6	234
35	25	63.0	151
50	35	40.8	90
50	50	38.7	79

It should be noted that, for the purposes of participation in the competition, the gearing system in the vehicle should be changed, so that it could reach greater speeds. Although not experimentally tested, the vehicle was designed so that higher speeds can be obtained, only by adjusting some of the control parameters.

7.4 - Chapter conclusions

This chapter discussed the methods used to validate the developed platform. Therefore, taking into account the modular architecture of the system, the tests and respective results are shown, for the various modules of the developed computer vision system. These tests reveal the robustness of the method against various scenarios, so that one can infer about the region of operation of a certain module, given the test conditions.

Next the navigation tests were presented, with special emphasis on trajectory. The tests faced various speed profiles against different control parameters. Although the tests are not accurate enough due to considerable errors in the test platform, these tests are an indicator of its robustness.

Chapter 8

Conclusions and Further Work

8.1 - Conclusions

The main objective set for this work was to develop a platform that would allow a vehicle to participate in the autonomous driving competition held at the National Festival of Robotics, using a previously adapted vehicle with some embedded low-level control.

Thus, a requirement analysis was conducted, concluding that the main sensory component of the vehicle should be computer vision. In this sense, the problem was split into parts. The main part consisted in developing an algorithm able to determine the lateral boundaries of the track, based on the images coming from two cameras directed to the track plane. This algorithm proved to be robust and reliable, correctly determining three lines corresponding to the lane markings placed on the track. Subsequently, it was developed a method to determine secondary track characteristics, including the detection and determination of traffic lights and the key point of track location, the crosswalk. These methods were developed in the C++ language, through the routines implemented in the portable, free and noncommercial OpenCV library.

With the information returned by the vision based system, a relative lateral position of the vehicle on the track is determined, which integrated with a odometry system allows a sectorial location of the vehicle on the track. This way, task planning is accomplished providing an effective method for the vehicle navigation throughout the track.

During the project development, it was also taken into account the portability between various operating systems, and the proposed system is able to run both in the Linux environment as in Windows.

For the purpose of results analysis, it should be noted the important role of the Ubisense system which allowed, with relative accuracy, the validation of the developed platform by tracking the vehicle with a pre-defined trajectory. It should be also highlighted that the test logs (around 60 laps) could be used as a data bank (available in [54]) for the study areas of learning algorithms and artificial intelligence. However, even with some filtering, the results show that the data provided by the Ubisense system includes a considerable error with respect to the real position of the vehicle, which, along with its high sensibility to external elements (such as persons), may invalidate its migration from the academic environment to real vehicle tracking situations.

Finally it can be stated that, despite the project validation method is not very enlightening, due to the unknown error correlation between the Ubisense outputs and the internal state of the vehicle, an adaptable, scalable and efficient autonomous driving platform was obtained, meeting very positive outcomes.

8.2 - Future work

The remaining aspects in order to fully prepare the vehicle to the competition, include the parking maneuver, the detection of signals along the track and the working zone navigation. In fact, some steps were taken in this direction during the project, particularly in detecting the working zone, but due to its non-completion, this aspect was not documented in this thesis. The stages include a transformation of the RGB image into the HSV color space for subsequent segmentation, in order to isolate orange elements in the raw image. The actual detection of the cones can then be done through template matching methods, or by determining the central moments of the objects. Thus it is possible to delimit lateral borders for the robot navigation. Regarding the detection of signals along the track, this can be accomplished via the same method developed for the detection of the upper signaling panels, where the algorithm is run concurrently or sequentially with the main control algorithm, eventually with a lower periodicity. As for the parking maneuver, this can be done through the trapezoids' searching method, narrowing the search to a certain area range, or considering only the trapezoids that contain the "P" drawn sign, in order to determine the free parking spot. The navigation can then be performed similarly to the developed controller used in the lane following procedure.

Regarding the calibration of the steering controller parameters, higher accuracy can be accomplished via a supervised learning method, where the inputs to the training algorithm are the positions returned by the Ubisense system or the computed error with respect to an "ideal" trajectory. It should also be noted that using a computer with faster computing speeds, the frequency of execution of the main cycle may be increased above the 10FPS achieved with the 2.00GHz Core 2 Duo processor used in this work.

From a structural standpoint, the drive system can also be modified with a motor able to reach higher speeds.

The obstacles detection should also be improved to anticipate the detection of obstacles at higher speeds, through other sensors, such as a laser range finder. Vision methods might also be considered, which may be achieved by segmenting the green color (color of the obstacles) and subsequent search for objects in the image with obstacle features. Taking advantage of the use of two cameras, the adoption of a Stereo Vision System to determinate the distance to the obstacle, should also not be discarded.

Regarding the process of image fusion, it can be improved, bridging any discontinuity in the final fused image. This fusion can be performed in the real world coordinates, that is, for each camera a transformation function is associated, converting the image coordinates into the robot reference frame (a reference that moves with the robot), merging two images in this referential.

References

[1] P. Gomes, "FEUPCar: Condução Autónoma no Festival Nacional de Robótica", MsC Thesis, Electrical and Computers Engineering, Universidade do Porto, Porto, 2010.

[2] I. Encyclopædia Britannica, *Encyclopædia Britannica Eleventh Edition*, Eleventh Edition ed. vol. V28: Horace Everett Hooper, 1911.

[3] D. E. Horvath. (1997 - 2009). *Ralph Teetor - Inventor of Cruise Control*. Available at: http://www.cruise-in.com/resource/cismar08.htm (last accessed on December 2009)

[4] experiencefestival.com. *Cruise Control - History*. Available at: http://www.experiencefestival.com/cruise_control_-_history (last accessed on December 2010)

[5] tech-faq.com. *Driveless Car*. Available at: http://www.tech-faq.com/driverless-car.html (last accessed on December 2010)

[6] MedLibrary.org. *EUREKA Prometheus Project*. Available at: http://medlibrary.org/medwiki/EUREKA_Prometheus_Project (last accessed on December 2010)

[7] MedLibrary.org. *VaMP*. Available at: http://medlibrary.org/medwiki/VaMP (last accessed on December 2009)

[8] J. Schmidhuber. *ROBOT CARS - autonomous vehicles - history of self-driving cars*. Available at: http://www.idsia.ch/~juergen/robotcars.html (last accessed on December 2010)

[9] StateMasterEncyclopedia. *Driverless car*. Available at: http://www.statemaster.com/encyclopedia/Driverless-car (last accessed on December 2010)

[10] E. D. Dickmanns. (2004) Dynamic vision-based intelligence. *AI Magazine (AAAI)*. 21.

[11] Bookrags.com. *Driveless car Summary*. Available at: http://www.bookrags.com/wiki/Driverless_car (last accessed on December 2010)

[12] Vislab. *Vislab Prototypes*. Available at: http://vislab.it/Prototypes (last accessed on December 2010)

[13] M. B. Alberto Broggi, Alessandra Fascioli, Gianni Conte. (October 1998) Automatic Vehicle Guidance: the Experience of the ARGO Autonomous Vehicle. Available: http://ftp.utcluj.ro/pub/docs/imaging/Autonomous_driving/Articole%20sortate/Lazar%20Mircea/AutoDriving2/parma/www.ce.unipr.it/people/broggi/publications/argo.pdf

[14] TerraMax. *Team TerraMax - Big Truck Robotics*. Available at: http://www.terramax.com/ (last accessed on December 2010)

[15] T. K. C. Thorpe, M. Hebert, and S. Shafer, "Toward Autonomous Driving: The CMU Navlab. Part II: System and Architecture," 1991.

[16] Vislab. (2009). *BRAiVE*. Available at: http://braive.vislab.it (last accessed on December 2010)

[17] Vislab. (2010). *VIAC Intercontinental Challenge*. Available at: viac.vislab.it/ (last accessed on January 2011)

[18] DARPA. *Darpa Grand Challenge*. Available at: http://www.darpa.mil/grandchallenge/index.asp (last accessed on December 2010)

[19] ELROB. *ELROB - The European Robot Trial Website*. Available at: http://www.elrob.org/ (last accessed on December 2010)
[20] SPR. *Sociedade Portuguesa de Robótica*. Available at: http://www.spr.ua.pt/site/ (last accessed on January 2011)
[21] SPR. *Robótica 2011 - Festival Nacional de Robótica*. Available at: http://robotica2011.ist.utl.pt/ (last accessed on January 2011)
[22] SPR. *Autonomous Driving Competition Rules* Available at: http://robotica2010.ipleiria.pt/robotica2010/images/stories/robotica2010/CA/Conducao_Autonoma_2010_EN.pdf (last accessed on January 2011)
[23] M. d. R. C. Sequeira, "Perception and intelligent localization for autonomous driving", MsC Thesis, Electronics, Telecomunications and Informatics Department, Aveiro University, Aveiro, 2009.
[24] J. N. d. S. Carvalho, "ROTA'2008 : um robô para condução autónoma ", MsC Thesis, Electronics, Telecomunications and Informatics Department, Aveiro University, Aveiro, 2008.
[25] J. r. M. d. Oliveira, "World Representation for an Autonomous Driving Robot", MsC Thesis, Electronics, Telecomunications and Informatics Department, Aveiro University, Aveiro, 2009.
[26] R. Cancela, M. Neta, Oliveira, and M. Santos, "ATLAS III: Um Robô com visão orientada para provas em condução autónoma.", presented at the Proc. of the "Encontro Científico do Festival Nacional de Robótica (ROBOTICA2005)", Coimbra, 2005, pp. 32-40.
[27] M. Oliveira and V. Santos, "A Vision-based Solution for the Navigation of a Mobile Robot in a Road-like Environment", presented at the Robótica, 2007, p. 8.
[28] M. Oliveira, P. Stein, A. J, and V. Santos, "Modular Scalable Architecture for the Navigation of the ATLAS Autonomous Robots", presented at the 9th Conference on Autonomous Robot Systems and Competitions, Castelo Branco, Portugal, 2009.
[29] M. Oliveira and V. Santos, "Multi-Camera Active Perception System with Variable Image Perspective for Mobile Robot Navigation", in *8th Conference on Autonomous Robot Systems and Competitions*, April 2nd, 2008.
[30] M. Oliveira and V. Santos, "Real Time Road Line Extraction with Simple Statistical Descriptors", presented at the IEEE International Conference on Intelligent Robots Systems (IROS 2008), Nice, France, 2008.
[31] A. M. F. Carvalhosa and T. L. B. Leite, "Versa Robot: Robot móvel versátil para competições em provas de robótica", Intership Report, Department of Electrical and Computer Engineering, Faculdade de Engenharia da Universidade do Porto, Porto, 2006.
[32] A. Bovik, "Introduction to Digital Image and Video Processing", in *Handbook of Image & Video Processing*, 2nd ed Burlington,MA: Elsevier Academic Press, 2005, pp. 11-1372.
[33] J. Guinot. *Image Filtering - Convolution Kernels*. Available at: http://www.ozone3d.net/tutorials/image_filtering.php (last accessed on January 2011)
[34] D. A. Forsyth and J. Ponce, "Linear Filters", in *Computer vision - a modern approach*, P. Education, Ed., ed Upper Saddle River, NJ, 2003, pp. 136-693.
[35] E. Arias-Castro and D. L. Donoho, "Does median filtering truly preserve edges better than linear filtering?", *3*, vol. 37, p. 1172, 2009.
[36] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, First Edition ed. vol. 1. Sebastopol, CA: O'Reilly Media Inc., 2008.
[37] Mathworks. *Morphology Fundamentals (Image Processing Toolbox)*. Available at: http://www.mathworks.com/help/toolbox/images/f18-12508.html (last accessed on January 2011)
[38] E. R. Dougherty, *An Introduction to Morphological Image Processing*. Bellingham: SPIE Press, 1992.
[39] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures", presented at the Comm. ACM, 1972, pp. 11-15.

[40] absoluteastronomy.com. *Hough Tranform: facts, Discussion Forum, and Encyclopedia Article*. Available at: http://www.absoluteastronomy.com/topics/Hough_transform (last accessed on January 2011)

[41] J. Canny, "A Computational Approach To Edge Detection", presented at the IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, pp. 679-698.

[42] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*, 1st ed. Chichester, West Sussex, United Kingdom: Wiley, 2009.

[43] T. Losi. *Team Losi Website*. Available at: http://www.losi.com/Products/Default.aspx?ProdID=LOSB0105 (last accessed on December 2010)

[44] Sharp. *Sharp GP2D12-15 Datasheet* [pdf]. Available at: http://www.acroname.com/robotics/parts/SharpGP2D12-15.pdf (last accessed on December 2010)

[45] Lazarus. *Lazarus Project Homepage*. Available at: http://www.lazarus.freepascal.org/ (last accessed on January 2011)

[46] Atmel. *AVR ATmega8 Datasheet*. Available at: http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf (last accessed on January 2011)

[47] Farnell. *A-max 32 Motors & Planetary Gearheads* [xml]. Available at: http://uk.farnell.com/jsp/level5/module.jsp?moduleId=en/204327.xml (last accessed on December 2010)

[48] Omron. *E6A2-CW5C 100P/R 2M Encoder Datasheet*. Available at: http://www.ia.omron.com/product/item/e6a21055a/ (last accessed on December 2010)

[49] Pololu. *Pololu - Dual VNH3SP30 Motor Driver Carrier MD03A Datasheet*. Available at: http://www.pololu.com/catalog/product/707 (last accessed on December 2010)

[50] Aten. *Aten USB to Serial Adapter* Available at: http://www.aten-usa.com/?product&cat=595&Item=UC232A (last accessed on December 2010)

[51] faqs.org. *UDP Connections*. Available at: http://www.faqs.org/docs/iptables/udpconnections.html (last accessed on January 2011)

[52] T. Stathaki, *Image fusion: algorithms and applications*, First ed. vol. 1. London: Academic Press, 2008.

[53] Ubisense. *Ubisense Main Page*. Available at: http://www.ubisense.net/en/ (last accessed on January 2011)

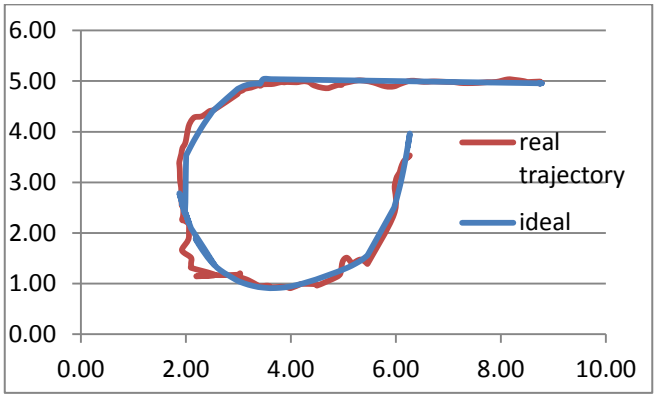
[54] A. Vidal. FEUPCAR 2.0 Ubisense and internal vehicle logs published on Feb. 2011 [Online]. Available: http://paginas.fe.up.pt/~ee05010/tese/

Appendix A

This appendix illustrates the various trajectories drawn by the vehicle during the validation process. It reports the five tests held for every situation (speed vs. number of considered samples). For every test the statistical indicators of average, maximum and the standard deviation error between the ideal trajectory against the real one are shown. Its analysis can be shown in detail in Chapter 7 – Experimental Results and Discussion of this thesis.

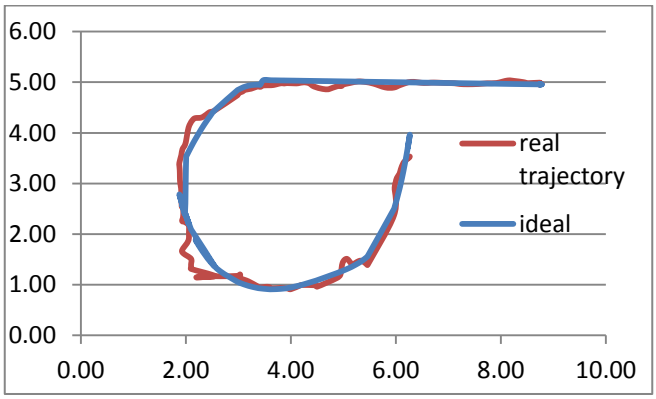
Straight Speed: 25cm/s , Curve Speed : 15 cm/s, 2 Samples

- **Test n°1**



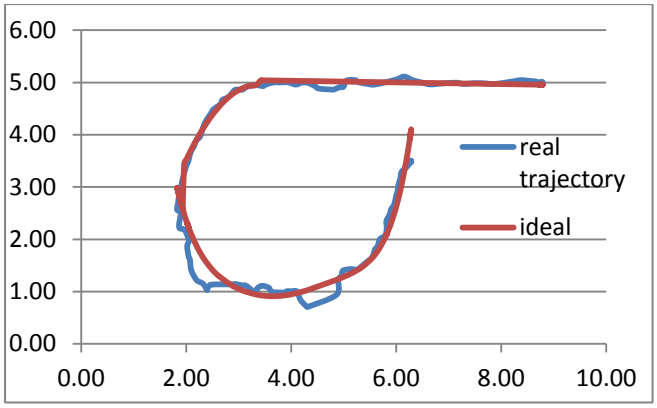
Average error	Max error	Standard Deviation
6.98 cm	52.91 cm	8.52cm

- **Test n°2**



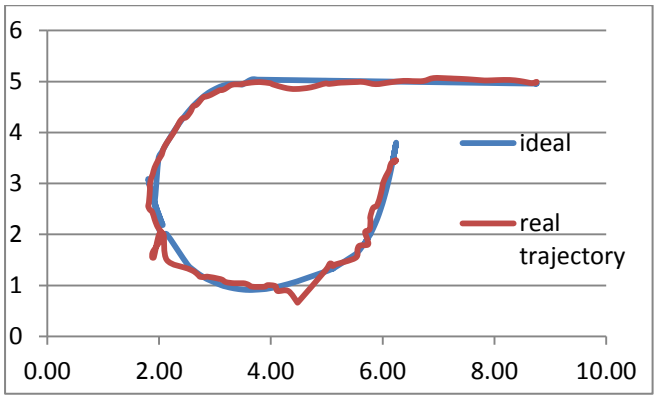
Average error	Max error	Standard Deviation
5.58 cm	57.261 cm	4.62cm

- **Test n°3**



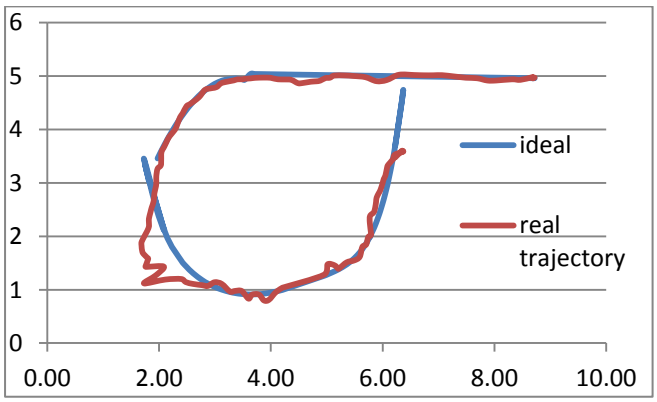
Average error	Max error	Standard Deviation
1.91cm	102.12 cm	8.6437cm

- **Test n°4**



Average error	Max error	Standard Deviation
8.24cm	68.51 cm	5.37cm

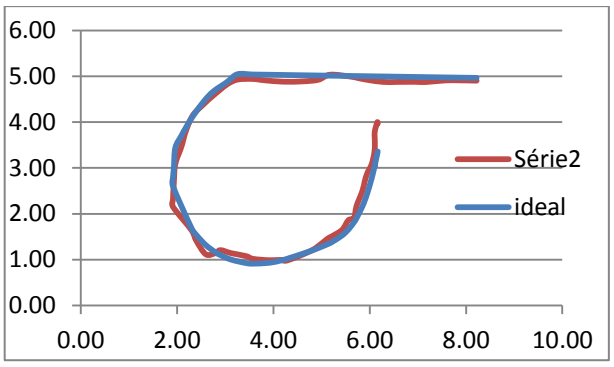
- **Test n°5**



Average error	Max error	Standard Deviation
0.24 cm	122.437 cm	16.57cm

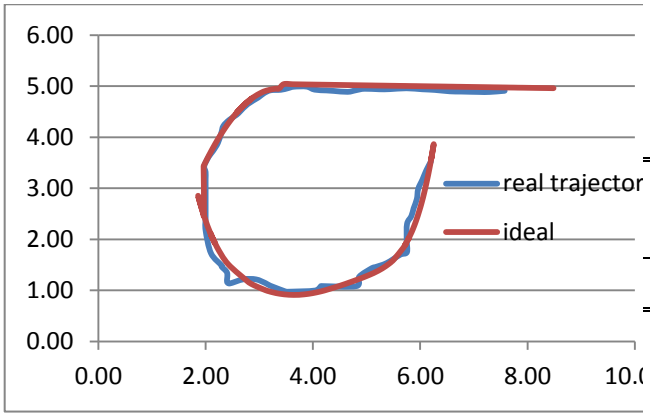
Straight Speed: 35cm/s , Curve Speed : 25 cm/s, 2 Samples

- **Test n°1**



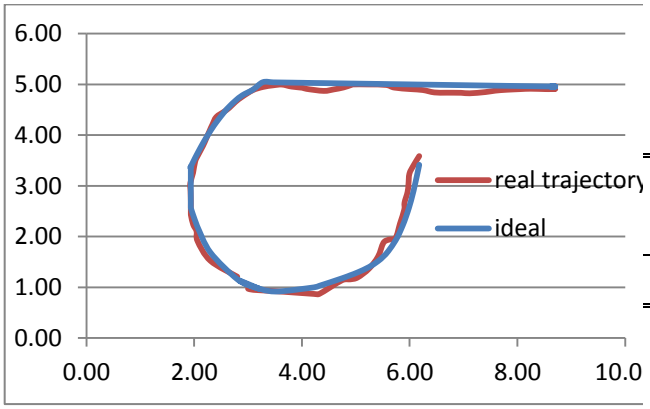
Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.22cm	52.437 cm	5.58cm

- **Test n°2**



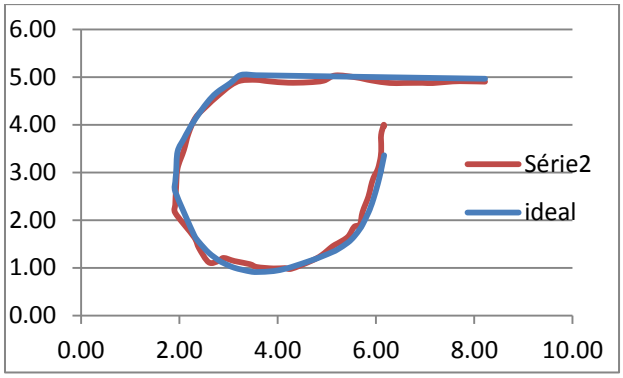
Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.22cm	52.437 cm	5.58cm

- **Test n°3**



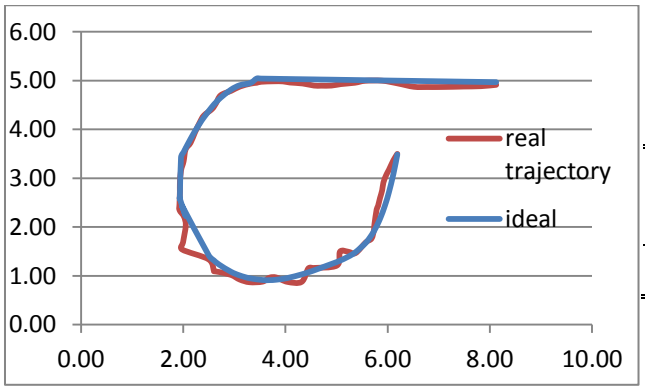
Experimental results:		
Average error	**Max error**	**Standard Deviation**
2.11cm	64.131 cm	2.76cm

- **Test n°4**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.24cm	73.964 cm	6.39cm

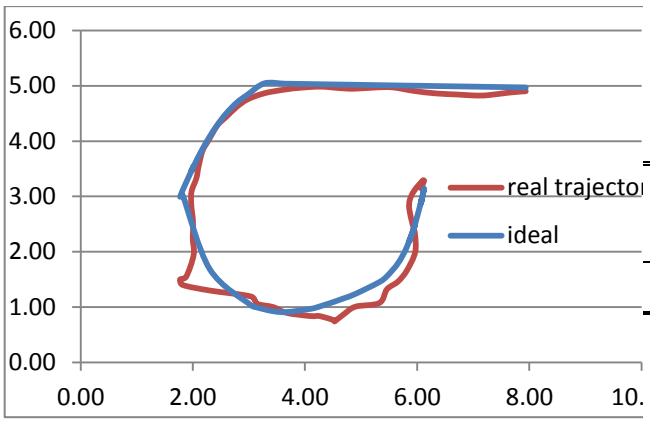
- **Test n°5**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
0.81 cm	56.25 cm	4.87 cm

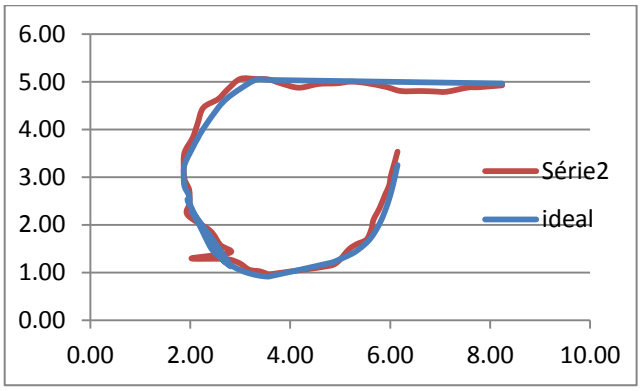
Straight Speed: 50cm/s , Curve Speed : 35 cm/s, 2 Samples

- **Test n°1**



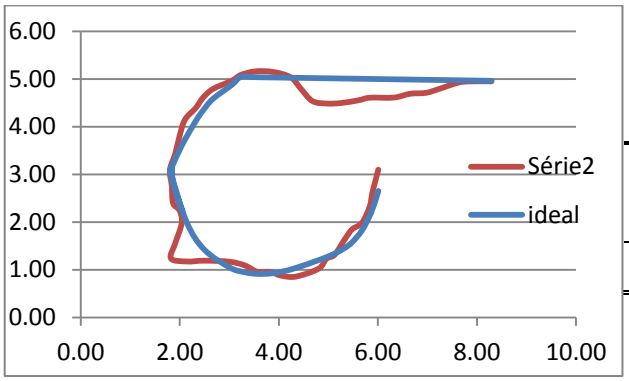
Experimental results:		
Average error	**Max error**	**Standard Deviation**
6.35 cm	72.39 cm	25.20 cm

- **Test n°2**



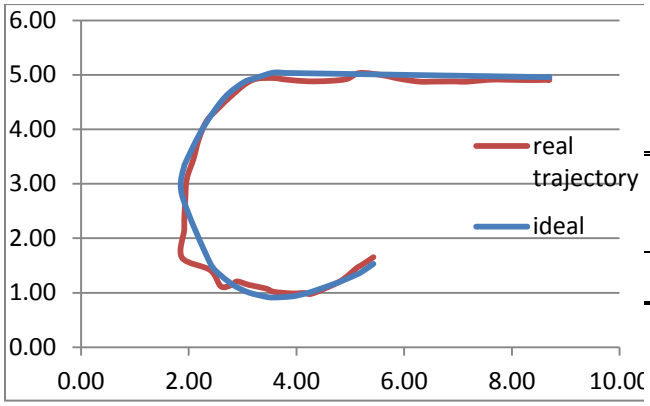
Experimental results:		
Average error	**Max error**	**Standard Deviation**
2.12 cm	43.94 cm	5.42 cm

- **Test n°3**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.67 cm	44.69 cm	9.09 cm

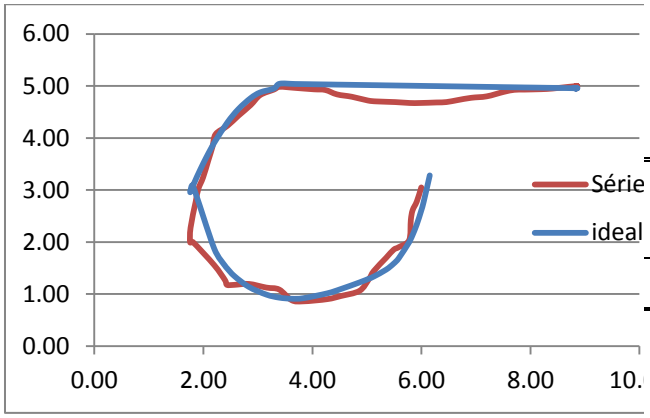
- **Test n°4**



Experimental results:

Average error	Max error	Standard Deviation
2.25 cm	44.05 cm	9.60 cm

- **Test n°5**

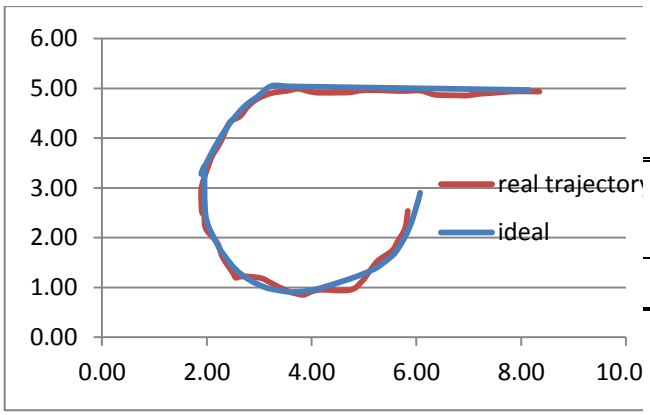


Experimental results:

Average error	Max error	Standard Deviation
3.41 cm	47.51 cm	6.43 cm

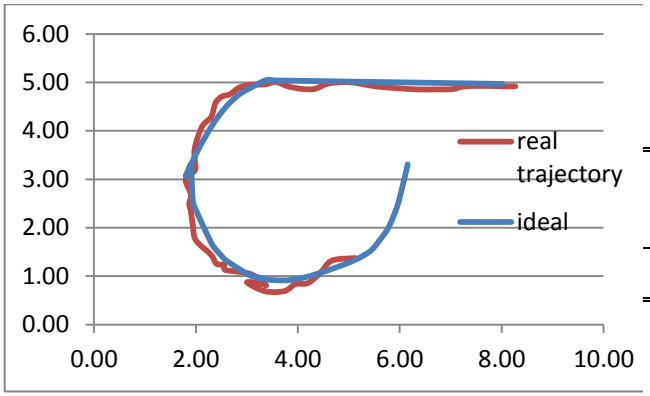
Straight Speed: 50cm/s , Curve Speed : 50 cm/s, 2 Samples

- **Test n°1**



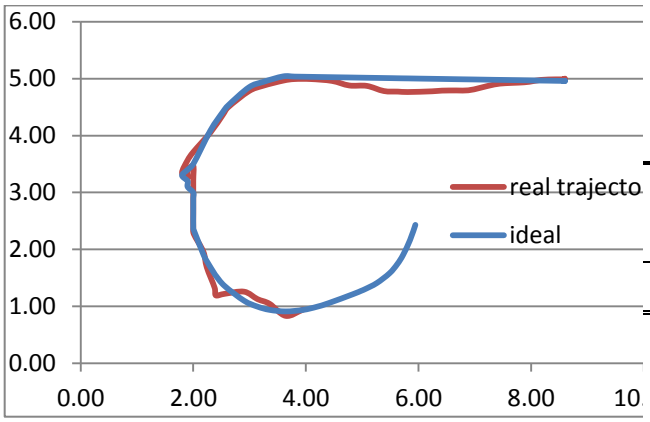
Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.59 cm	52.27 cm	8.67 cm

- **Test n°2**



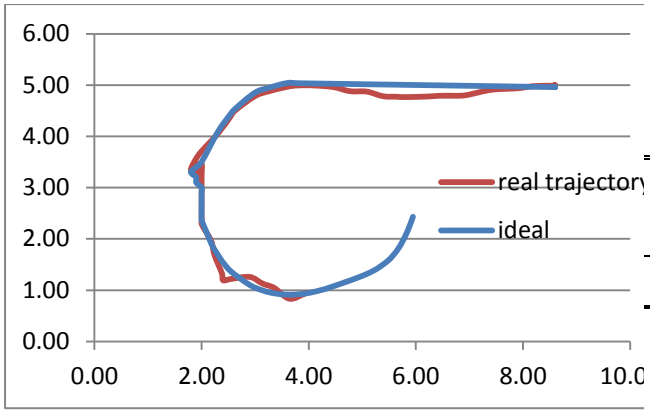
Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.33 cm	73.77 cm	11.03 cm

- **Test n°3**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
3.51 cm	46.75 cm	13.50 cm

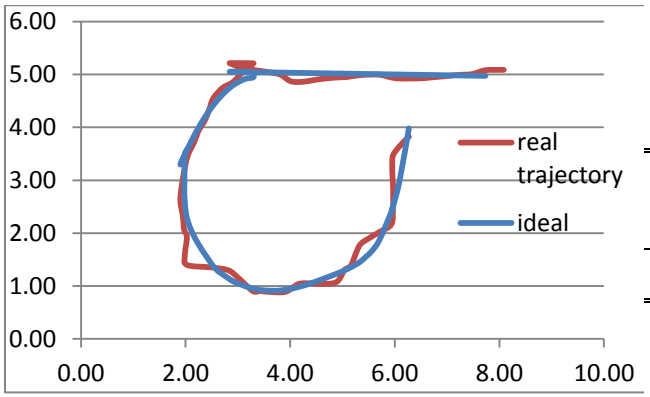
- **Test n°4**



Experimental results:

Average error	Max error	Standard Deviation
5.23 cm	50.35 cm	18.16 cm

- **Test n°5**

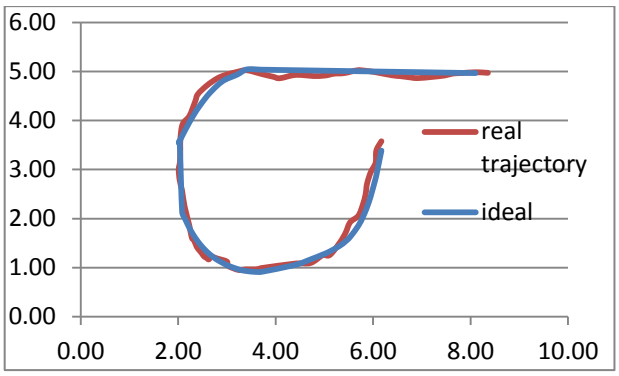


Experimental results:

Average error	Max error	Standard Deviation
2.50 cm	96.17cm	16.66 cm

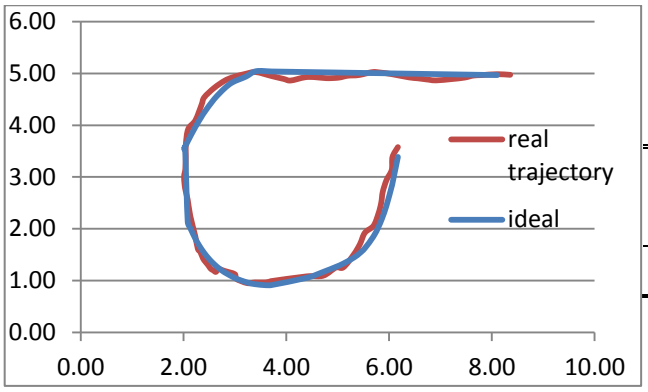
Straight Speed: 25cm/s , Curve Speed : 15 cm/s, 3 Samples

- **Test n°1**



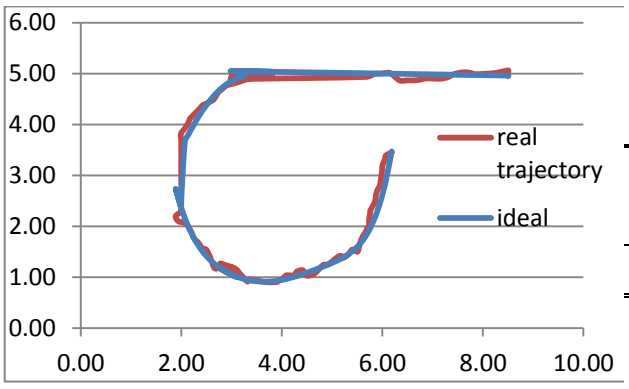
Experimental results:		
Average error	**Max error**	**Standard Deviation**
0.74 cm	50.63 cm	4.59 cm

- **Test n°2**



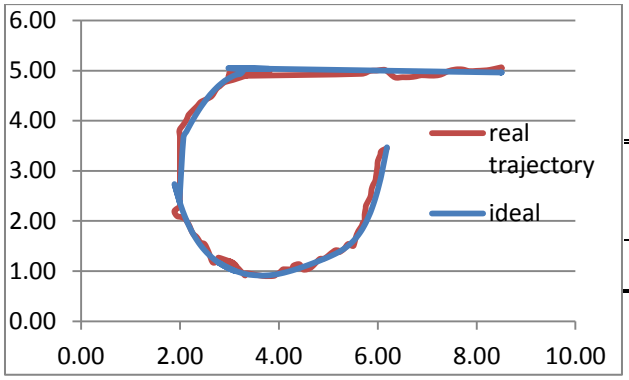
Experimental results:		
Average error	**Max error**	**Standard Deviation**
3.74 cm	65.63 cm	8.79 cm

- **Test n°3**



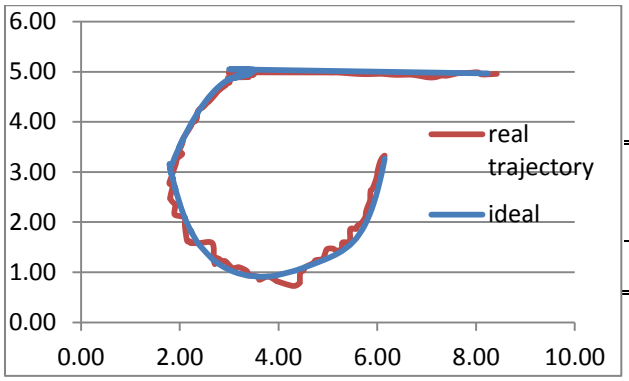
Experimental results:		
Average error	**Max error**	**Standard Deviation**
6.33 cm	55.53 cm	9.85 cm

- **Test n°4**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
17.03 cm	83.23 cm	21.09 cm

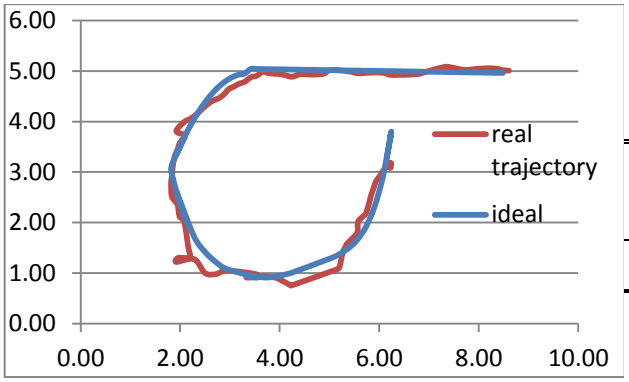
- **Test n°5**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
2.35 cm	42.93 cm	2.60 cm

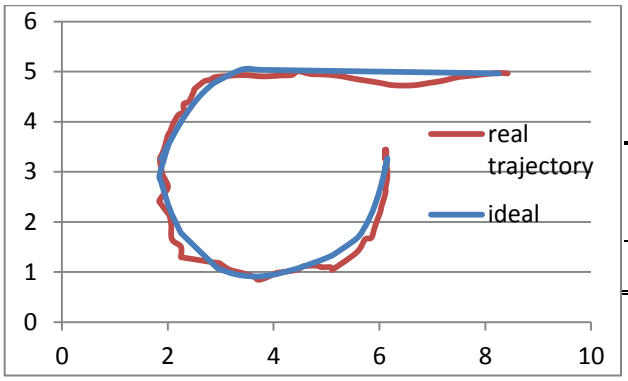
Straight Speed: 35cm/s , Curve Speed : 25 cm/s, 3 Samples

- **Test n°1**



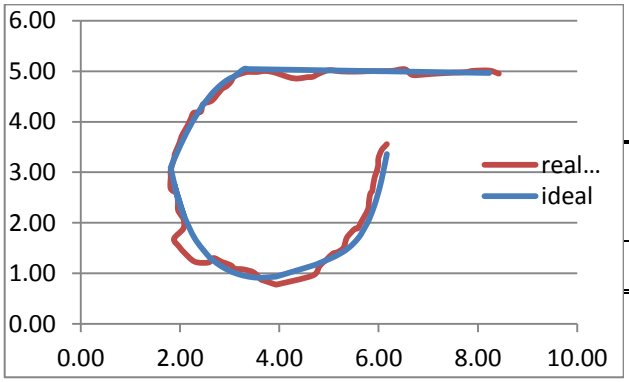
Experimental results:		
Average error	**Max error**	**Standard Deviation**
8.18 cm	46.08 cm	4.23 cm

- **Test n°2**



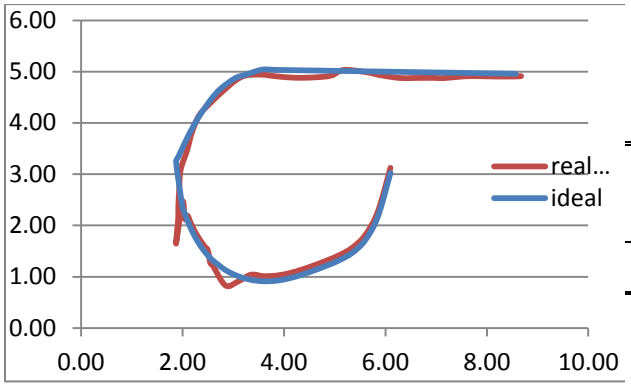
Experimental results:		
Average error	**Max error**	**Standard Deviation**
10.18 cm	36.28 cm	4.69 cm

- **Test n°3**



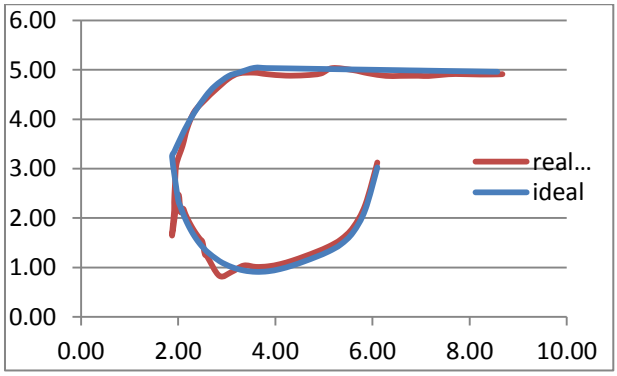
Experimental results:		
Average error	**Max error**	**Standard Deviation**
2.06 cm	34.65 cm	5.09 cm

- **Test n°4**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.26 cm	23.45 cm	3.09 cm

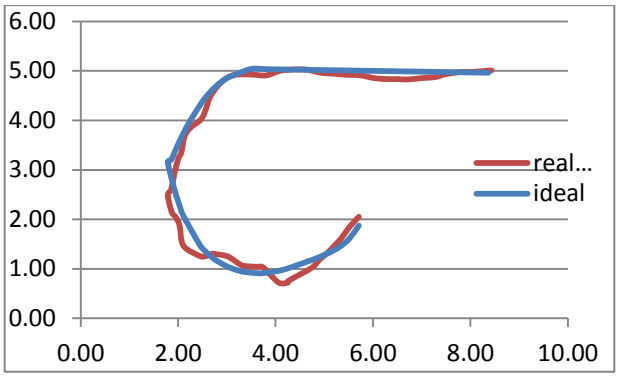
- **Test n°5**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
0.59 cm	30.45 cm	2.99 cm

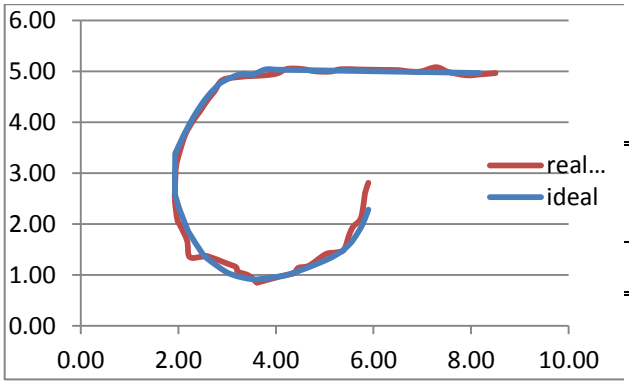
Straight Speed: 50cm/s , Curve Speed : 35 cm/s, 3 Samples

- **Test n°1**



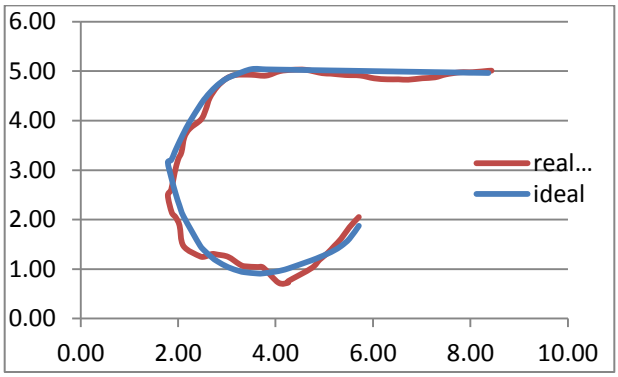
Experimental results:		
Average error	**Max error**	**Standard Deviation**
3.66 cm	52.67 cm	6.71 cm

- **Test n°2**



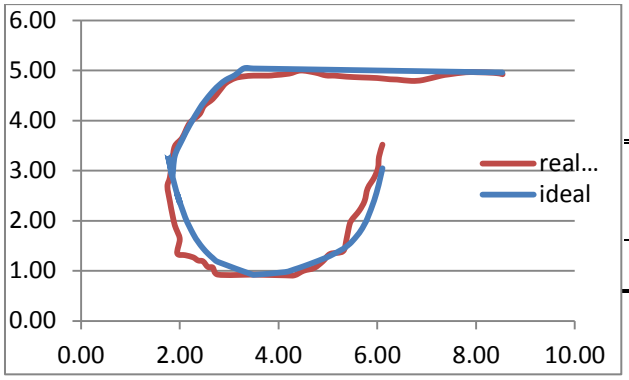
Experimental results:		
Average error	**Max error**	**Standard Deviation**
0.81 cm	32.67 cm	3.41 cm

- **Test n°3**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
4.21 cm	23.57 cm	5.45 cm

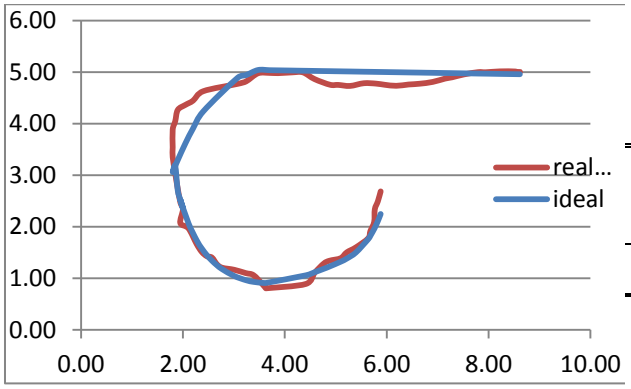
- **Test n°4**



Experimental results:

Average error	Max error	Standard Deviation
0.81 cm	32.67 cm	3.41 cm

- **Test n°5**

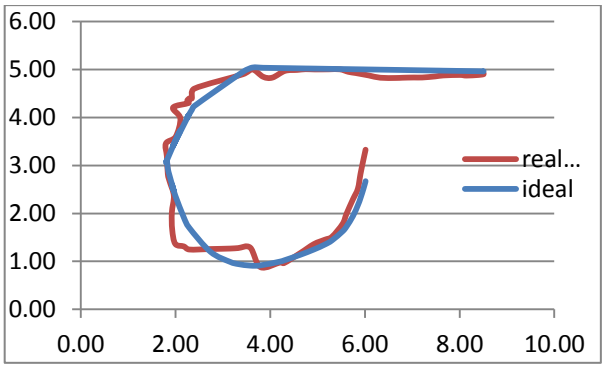


Experimental results:

Average error	Max error	Standard Deviation
2.04 cm	67.23 cm	6.01 cm

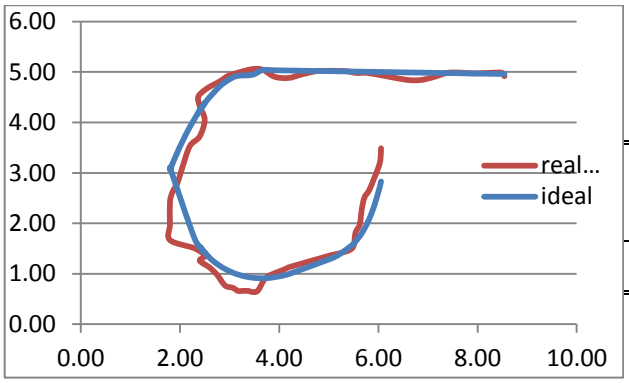
Straight Speed: 50cm/s , Curve Speed : 50 cm/s, 3 Samples

- **Test n°1**



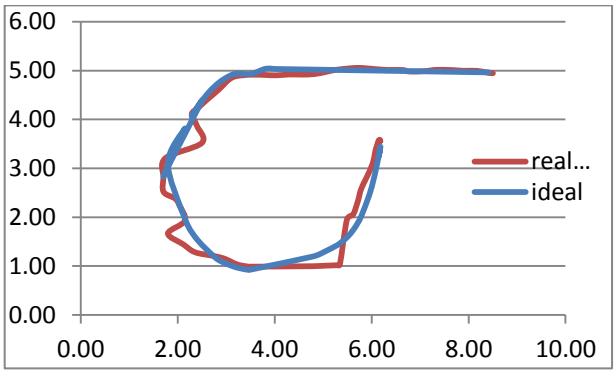
Experimental results:		
Average error	**Max error**	**Standard Deviation**
7.39 cm	110.23 cm	18.63 cm

- **Test n°2**



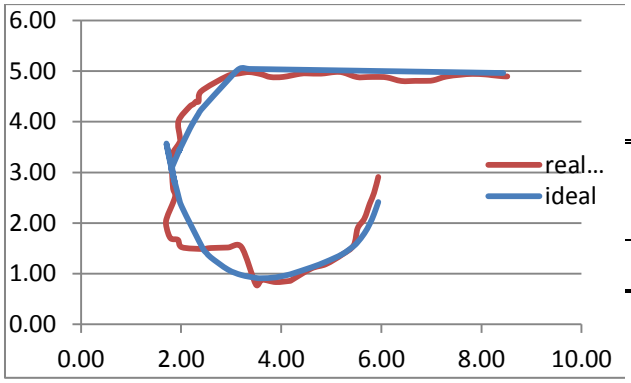
Experimental results:		
Average error	**Max error**	**Standard Deviation**
2.04 cm	67.23 cm	16.01 cm

- **Test n°3**



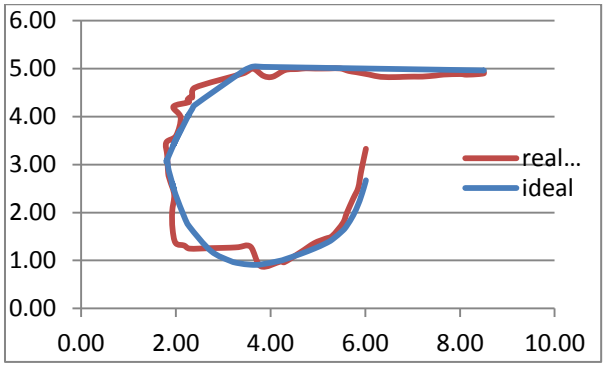
Experimental results:		
Average error	**Max error**	**Standard Deviation**
2.60 cm	56.22 cm	19.57 cm

- **Test n°4**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.64 cm	61.23 cm	18.93 cm

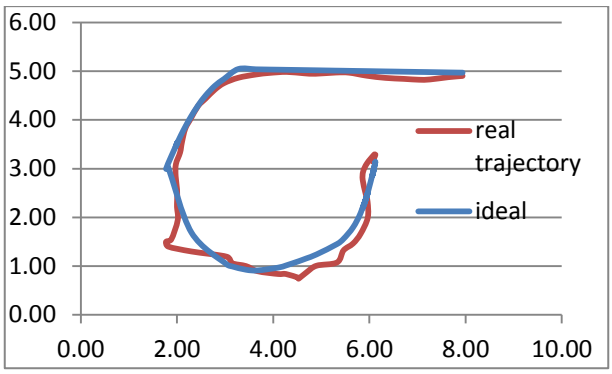
- **Test n°5**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
0.70 cm	78.83 cm	9.69 cm

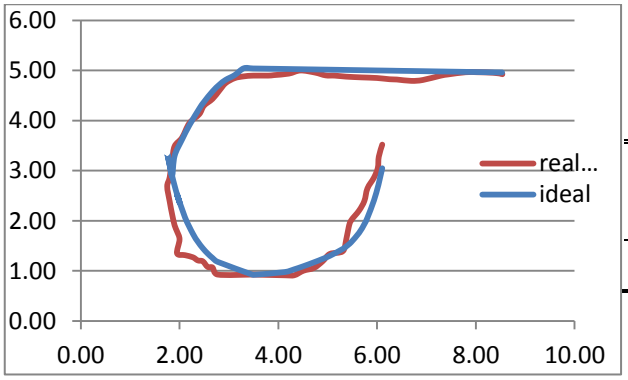
Straight Speed: 25cm/s , Curve Speed : 15 cm/s, 5 Samples

- **Test n°1**



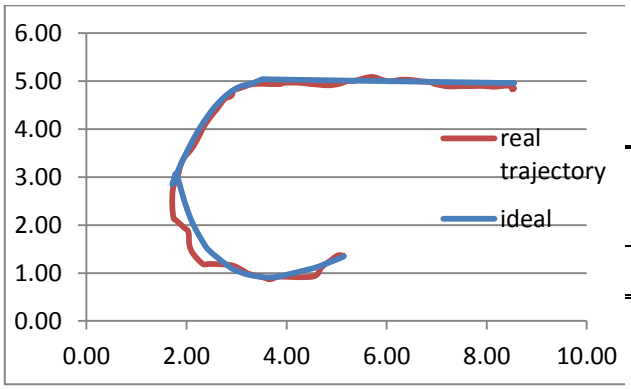
Experimental results:		
Average error	**Max error**	**Standard Deviation**
0.95 cm	52.55 cm	7.53 cm

- **Test n°2**



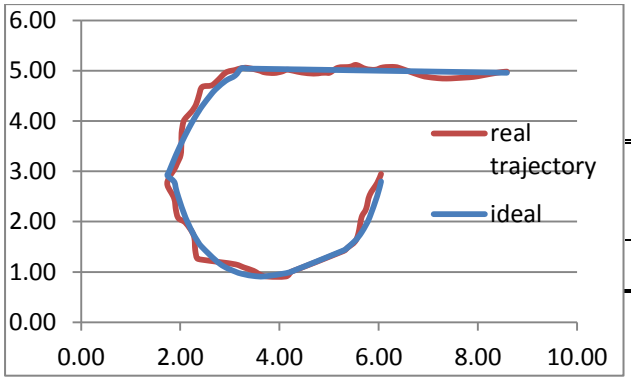
Experimental results:		
Average error	**Max error**	**Standard Deviation**
6.60 cm	37.47 cm	13.87 cm

- **Test n°3**



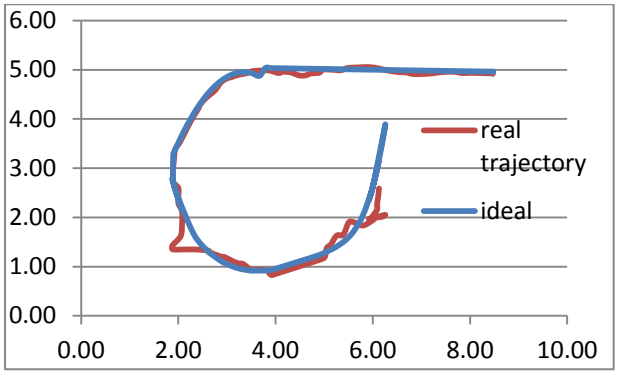
Experimental results:		
Average error	**Max error**	**Standard Deviation**
17.67 cm	46.98 cm	18.09 cm

- **Test n°4**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.79 cm	49.58 cm	5.75 cm

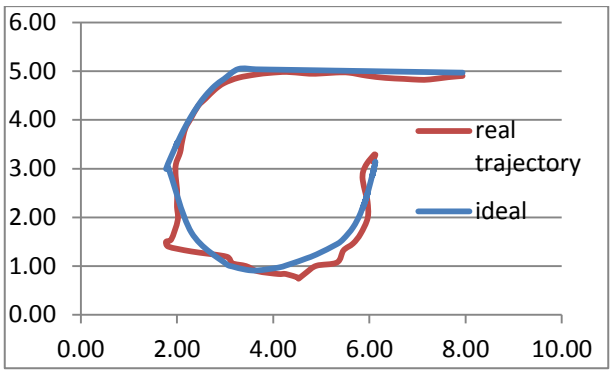
- **Test n°5**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
6.92 cm	49.73 cm	7.76 cm

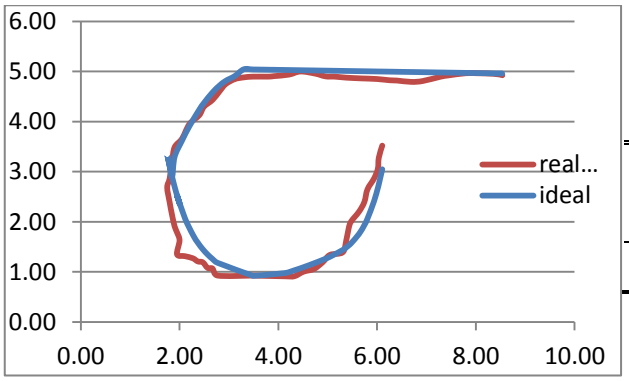
Straight Speed: 35cm/s , Curve Speed : 25 cm/s, 5 Samples

- **Test n°1**



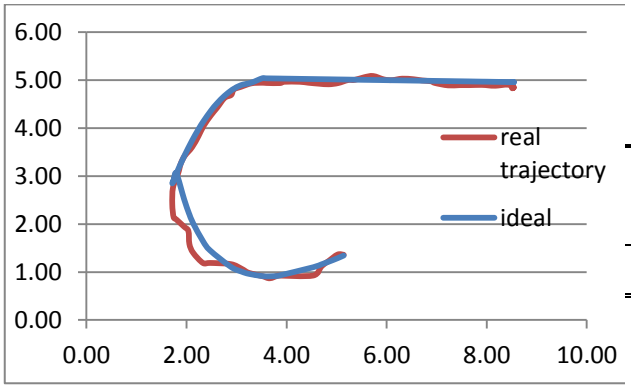
Experimental results:		
Average error	**Max error**	**Standard Deviation**
6.93 cm	51.95 cm	17.99 cm

- **Test n°2**



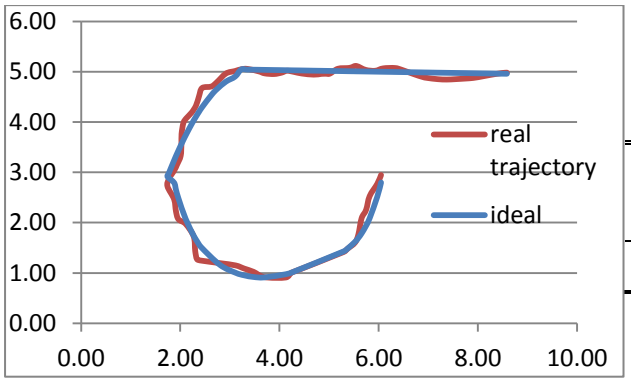
Experimental results:		
Average error	**Max error**	**Standard Deviation**
2.52 cm	71.56 cm	14.35 cm

- **Test n°3**



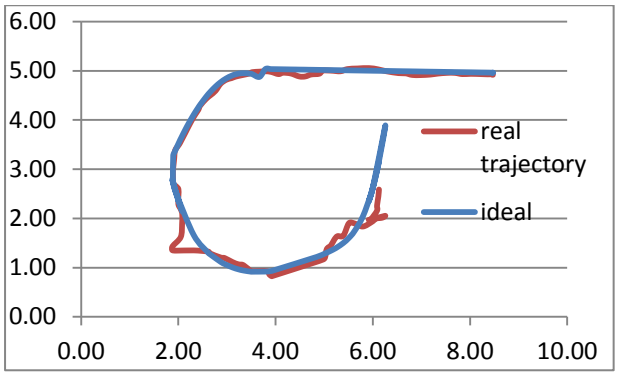
Experimental results:		
Average error	**Max error**	**Standard Deviation**
7.06 cm	79.90 cm	13.09 cm

- **Test n°4**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
8.96 cm	62.90 cm	9.00 cm

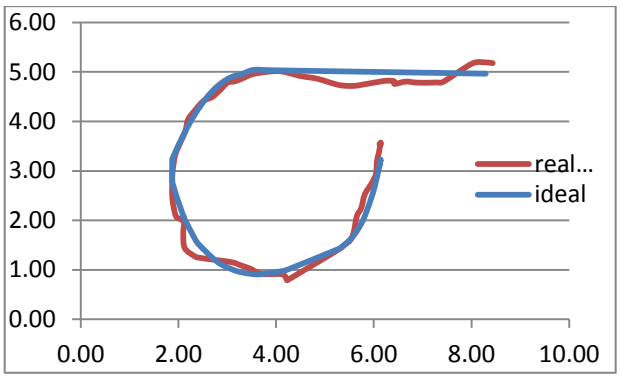
- **Test n°5**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
6.92 cm	49.73 cm	7.76 cm

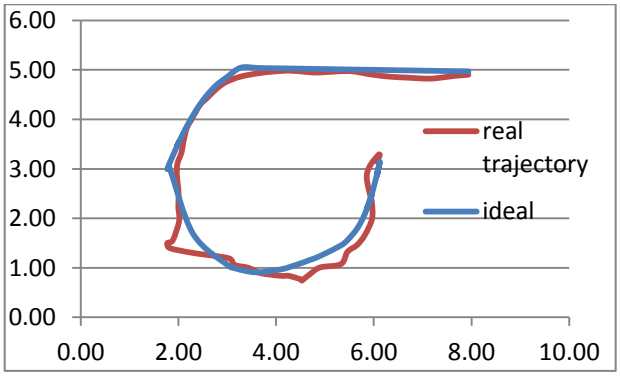
Straight Speed: 50cm/s , Curve Speed : 35 cm/s, 5 Samples

- **Test n°1**



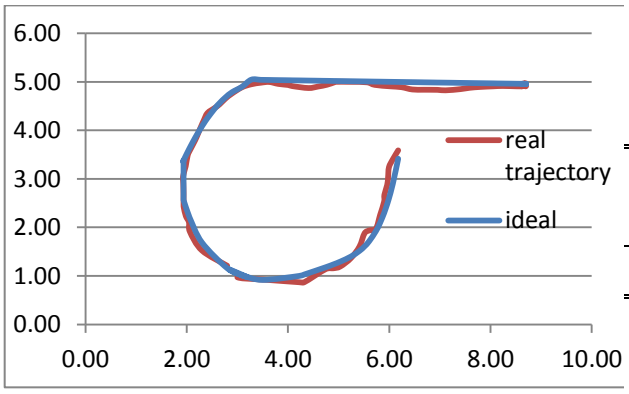
Experimental results:		
Average error	**Max error**	**Standard Deviation**
2.58 cm	28.95 cm	7.99 cm

- **Test n°2**



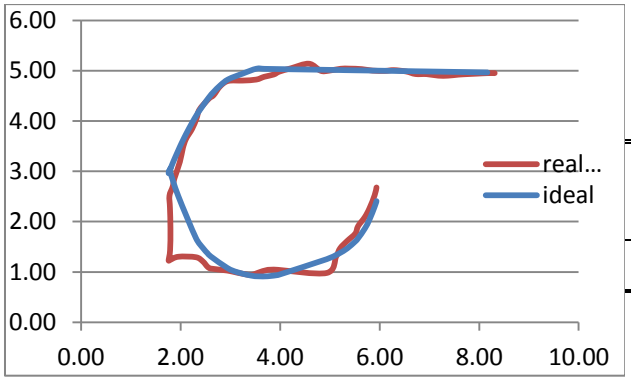
Experimental results:		
Average error	**Max error**	**Standard Deviation**
2.98 cm	42.35 cm	8.79 cm

- **Test n°3**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
5.08 cm	18.21 cm	4.68 cm

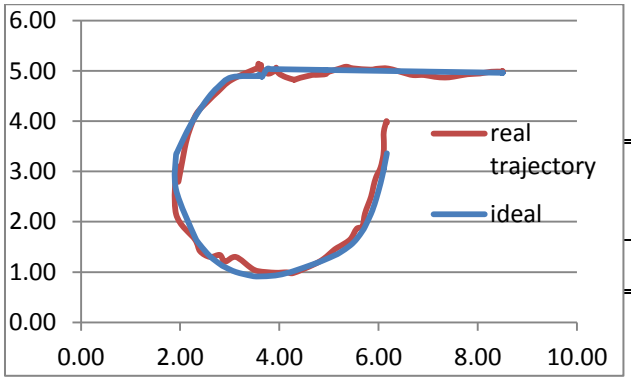
- **Test n°4**



Experimental results:

Average error	Max error	Standard Deviation
5.268 cm	43.21 cm	3.98 cm

- **Test n°5**

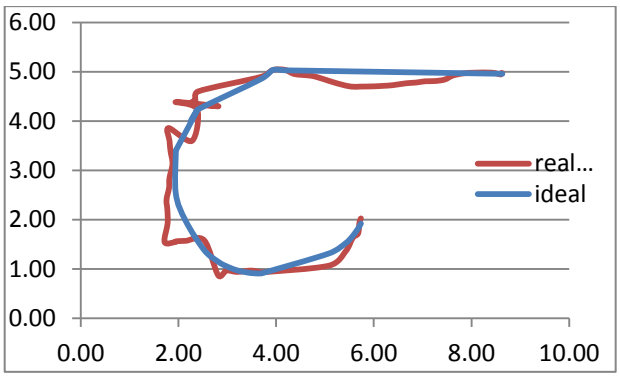


Experimental results:

Average error	Max error	Standard Deviation
4.85 cm	23.21 cm	4.20 cm

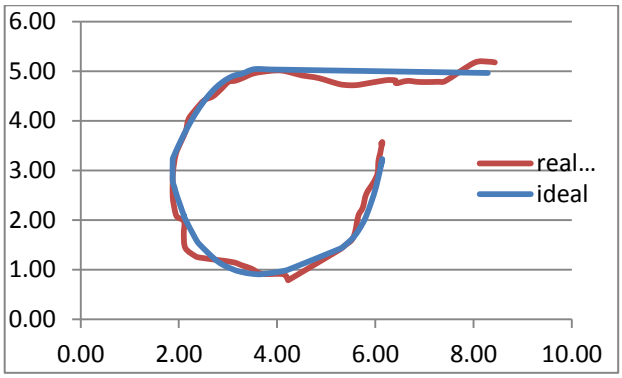
Straight Speed: 50cm/s , Curve Speed : 50 cm/s, 5 Samples

- **Test n°1**



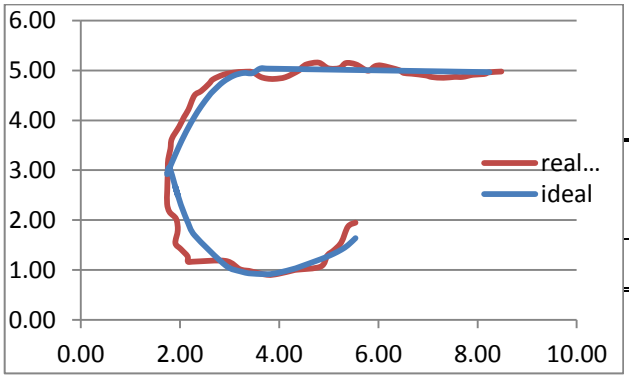
Experimental results:		
Average error	**Max error**	**Standard Deviation**
4.66 cm	58.33 cm	13.49 cm

- **Test n°2**



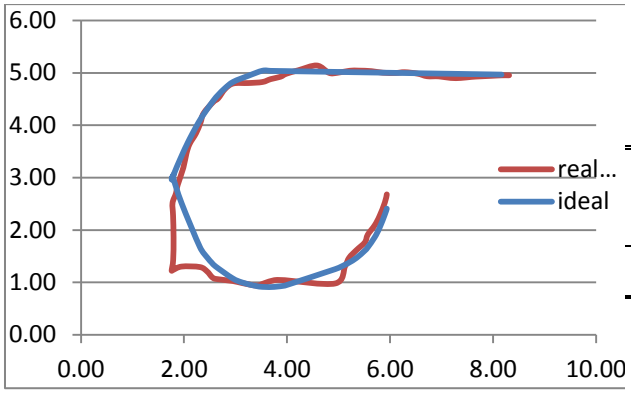
Experimental results:		
Average error	**Max error**	**Standard Deviation**
1.09 cm	43.93 cm	3.55 cm

- **Test n°3**



Experimental results:		
Average error	**Max error**	**Standard Deviation**
4.55 cm	37.19 cm	8.94 cm

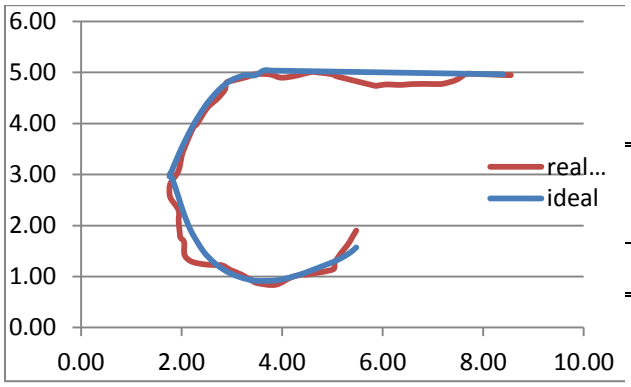
- **Test n°4**



Experimental results:

Average error	Max error	Standard Deviation
1.29 cm	37.22 cm	1.54 cm

- **Test n°5**



Experimental results:

Average error	Max error	Standard Deviation
4.54 cm	33.19 cm	4.67 cm