CaRINA Intelligent Robotic Car: Architectural Design and Applications

Leandro C. Fernandes, Jefferson R. Souza, Gustavo Pessin, Patrick Y. Shinzato, Daniel Sales, Caio Mendes, Marcos Prado, Rafael Klaser, André Chaves Magalhães, Alberto Hata, Daniel Pigatto Kalinka Castelo Branco, Valdir Grassi Jr., Fernando S. Osorio and Denis F. Wolf

Mobile Robotics Laboratory, University of São Paulo (USP)
Av. Trabalhador São-Carlense, 400 - P.O. Box 668 - 13.560-970, Sao Carlos, Brazil
{*lnd, jrsouza, pessin, shinzato, dsales, mgprado, rlklaser, hata, pigatto, kalinka, fosorio, denis*}*@icmc.usp.br,*
{*acmagalhaes,vgrassi*}*@sc.usp.br*

Abstract

This paper presents the development of two outdoor intelligent vehicles platforms named CaRINA I and CaRINA II, their system architecture, simulation tools, and control modules. It also describes the development of the intelligent control system modules allowing the mobile robots and vehicles to navigate autonomously in controlled urban environments. Research work has been carried out on tele-operation, driver assistance systems, and autonomous navigation using the vehicles as platforms to experiments and validation. Our robotic platforms include mechanical adaptations and the development of an embedded software architecture. This paper addresses the design, sensing, decision making, and acting infrastructure and several experimental tests that have been carried out to evaluate both platforms and proposed algorithms. The main contributions of this work is the proposed architecture, that is modular and flexible, allowing it to be instantiated into different robotic platforms and applications. The communication and security aspects are also investigated.

Keywords: Autonomous Vehicle Architecture, Embedded System Design, Robotic Vehicle Navigation, Computer Vision, Machine Learning, Intelligent Systems.

1. Introduction

According to World Health Organization [70], each year, approximately 1,2 million lives are lost due to traffic accidents worldwide, and around 50 million people suffer non-fatal car accidents. The majority of these are due to human error, like small distractions when driver uses a mobile phone or eats something, or more seriously faults like drunk driving or exceeding the speed limit. Self-driving cars promise to reduce this number of deaths and injuries as well as bring other benefits like optimal fuel usage, efficiency with freight transportation and traffic flow improvement in big cities, providing transportation for children and adults unable to drive like physically handicapped or visually impaired people. This paper describes the software and hardware architectures, and experimental results of the CaRINA project (Intelligent Robotic Car for Autonomous Navigation), which aims the development of an autonomous vehicle for urban environments.

In recent years, the Defense Advanced Research Projects Agency (DARPA) encouraged this research area by organizing three competitions for autonomous vehicles. The Grand Challenges [10], performed in 2004 and 2005, had focus on crossing the desert and skills known as "road following" and "obstacle avoidance". The first edition was disappointing. The best team traveled just 12 km from 224 km proposed by DARPA. In 2005, this challenge was proposed again and five vehicles finished the new route. In 2007, perhaps the most famous and relevant competition was launched, the Urban Challenge [11] where robot-cars should be able to safely navigate in an urban environment with other robot-cars and human-driven vehicles. They also had to obey some of the driving rules of California. This competition had 11 finalist teams and 6 robots finished the route of 96 km. Despite the impressive results of the 2007 edition which proved the feasibility of this kind of technology, it is important to highlight that the autonomous navigation in urban environments problem was simplified in order to evaluate the robots and perform this competition.

In Europe, the European Land-Robot Trial (ELROB) [16] is an initiative similar to DARPA competitions but without a winner. It is performed every year since 2006 alternating between military and civilian. Europe has also initiatives where companies, universities and research centers have cooperation projects to develop technologies to improve safety and intelligence from vehicles. Some of them are "The European Field Operational Test" (EuroFot) [17] and "Highly Automated Vehicles for Intelligent Transport" (HAVEit) [26] which had focused on developing systems like ACC (Adaptative Cruise Control), FCW (Foward Collision Warning), BSIS (Blind Spot Informa-

(a) CaRINA I (b) CaRINA II

Figure 1: The vehicular robotic platforms of CaRINA project

tion System), LDW (Lane Departure Warning) and others. Finally, in 2011, Europe hosted the "Grand Cooperative Driving Challenge" (GCDC) [22] that boosted research in cooperative driving systems, where a robot can communicate with other robots and environment in order to negotiate its planning and control.

Despite the great importance that the latest initiatives and competitions received, it is important to note that development of autonomous vehicles has been studied since 80s by various research groups around the world, such as the ARGO project [9], and VaMoRs vamp [24], NavLab [66], and several others. The development of autonomous vehicles is a topic of research relatively large and complex, allowing a large number of approaches that can be seen in several surveys as [6, 5, 14, 15]. Even today there are research projects on autonomous vehicles with different approaches and prototypes such as AUTONOMOSlabs [4] that has two vehicles - the "Spirit of Berlin" and "Made In Germany" which recently conducted the "Mission: Brandenburger Tor" where the vehicle traveled on the streets and highways of Berlin without driver's actuation. Another group which also has projects with smart cars is VisLab [69] that created several prototypes - and the most recent are "BRAiVE" (2009), "VDCL Grandeur" and "Porter" (2010) that traveled a route of 13,000 km from Italy to China. Currently, perhaps the most famous vehicle and considered state of the art is the "Google Self-Driving Car" [23] that took all technology developed in the DARPA Urban Challenge as its starting point.

Although several research groups have been actively working on autonomous vehicles around the world, few initiatives take place in Latin America. One of the first projects in this area is developed at the Federal University of Minas Gerais. Its software architecture is described in [52]. Experiments of autonomous navigation are reported in [34, 35]. Researchers from Federal University of Itajuba has also been working on autonomous navigation [39] and parking [67]. Both groups have fully automated cars and have been testing navigation algorithms in open spaces. A commercially automated vehicle has been recently acquired by Federal University of Espirito Santo, which has been actively working on obstacle detection using stereo vision [68]. Some research groups have also been working on algorithms such as: pedestrian recognition [51], autonomous parking [3, 29], and collision risk estimation [40]. They are still working on the automation of their platforms and testing in simulators and using offline data. Besides commercial vehicles, other types of outdoor robotic platforms have been also used to autonomous navigation [12, 1].

The CaRINA project has been developed at the INCT-SEC (Brazilian National Institute of Science and Technology for Critical Embedded Systems) and CRob/SC (Center for Robotics of University of São Paulo at São Carlos). It began in April 2010 with an electric service car, which has been modified for computer control and instrumented with a variety of sensors for perception. In October of the same year the vehicle (known as CaRINA I) was performing its first autonomous control test. Few months later

CaRINA I was able to autonomously traverse a 1 km trajectory in a controlled urban environment. The work on CaRINA II (a standard Fiat Palio Adventure) started in July 2011. One year later, to the best of our knowledge it has been the first commercial vehicle from Latin America capable to perform autonomous navigation in the streets[1]. Figure 1 shows the CaRINA I and II research platforms.

The main contributions of this paper are: *(i)* The proposed system architecture that is modular and flexible, allowing to instantiate it into different hardware platforms (CaRINA I and II), and allowing to be customized to different specific practical applications, as presented in this work; *(ii)* The system development methodology proposition based on an Open platform (ROS - described in section 2.1) exploiting its main characteristics as a service-based architecture, and also integrating it with a simulation facility based on Gazebo, that allows to perform realistic virtual simulations (switching between virtual and real components into the system architecture); *(iii)* The different navigation systems which present original solutions allowing practical autonomous vehicle navigation based on reactive behaviors, path planning and topological navigation based on state detection; *(iv)* The discussion, proposition and introduction of facilities regarding the communication security, in order to provide a safer service-based and distributed processing architecture/components; *(v)* The study, development and practical evaluation of VANETs (Vehicular Networks) integrated into this context of distributed services architecture, also providing a safe communication. The main advantages of the proposed architecture related to other works found in the literature are the flexibility of the proposed architecture, demonstrated by the different possible applications/configurations obtained using it, and also the aspects related to the communication security which usually are not provided/discussed in other works.

This paper describes the hardware and the software architectures of the CaRINA Project, experimental platforms, as well as the autonomous navigation results obtained. It is organized as follows: Section 2 describes the system archicture based on ROS, the hardware and software of the CaRINA experimental platforms, and aspects of the simulated models. Section 3 presents the application of the proposed architecture into four different navigation systems, along with experimental results. Section 4 describes an investigation on vehicular networks and security communication. Finally, Section 5 presents the conclusions and future work.

2. System Architecture

In order to design, build and test intelligent robotic car systems, we should be able to select, integrate and configure the hardware, and then develop the software to operate with different types of sensors and actuators. The sensors range from LIDARS (Laser sensors), monocular-stereo-spherical-panoramic cameras, GPS, Compass, IMU, and also, other devices used to obtain the feedback from the vehicle state (e.g. wheels odometer and steering angle encoder), which require specific drivers, interfaces and software setup. It is also necessary to consider the different types of actuators, used to control the steering wheel, the acceleration and brake pedals, and, if necessary, to make the gearbox shift. This is a very difficult and time consuming task if you do not have adequate tools to deal with this system design complexity.

Intelligent and autonomous vehicles applications design requires the development of different hardware and software modules, and also interfaces that connect users, robot control systems and the embedded hardware components. In the particular case of the intelligent robotic car design, the modeling and simulation of the system can be successfully done using VST (Virtual Simulation Tools/Technology) and after transpose the virtual system into an actual robotic system. Adopting VSTs to model and simulate autonomous vehicles, testing and evaluating the different aspects of the system before implementing it physically, is an efficient manner to decrease the development time. Also, it allows to guarantee the safety during the initial and critical phase of tests/validations of the system. This type of approach is becoming more and more common not only in the robotics design but also in other industrial products design [45, 44, 46, 20, 19].

Furthermore, it is also required a middleware providing tools for interfacing with different embedded system modules, hardware abstraction, and also providing communication facilities, since the implementation of the intelligent robotic car system usually requires the adoption of a distributed processing system with multiple dedicated processing modules.

ROS (Robotic Operating System) was chosen as the main software package tool to allow us to develop CaRINA I and CaRINA II, because it provides several different resources, as for example: hardware abstraction, communication facilities, management of distributed process, implementation of useful debugging tools (e.g. logging services, visualization and simulation tools). The ROS provides an almost complete suite of tools and resources that allows the design, simulation, validation, test and practical deploy of complex robotic systems.

2.1. Embedded System Design using ROS

ROS is a service-based architecture and has becoming one of the most important platforms for robotic research and development [73]. It is also fully integrated with important tool and libraries, such as: OpenCV that provides an extensive library of functions and methods for Computer Vision and Image Processing; PCL, which allows the 3D Point Cloud (e.g. Laser and Stereo-Camera data) processing and visualization; Gazebo that is a 3D Realistic Virtual Simulation Tool (VST); and Player, which

[1] http://www.youtube.com/lrmicmc

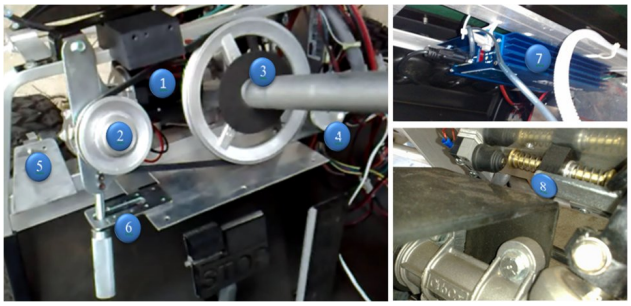


Figure 2: CaRINA I steering system. In details: (1) motor, (2) coupling system pulley, (3) steering system component, (4) digital encoder, (5) coupling detection switch, (6) coupling handspike, (7) RoboteQ motor controller and (8) limit switches.

provides a network interface to a variety of robot and sensor hardware [73, 72, 71]. As a meta-operating system for robots it provides: hardware abstraction, low-level device control, commonly-used functionality, message-passing between processes, and package management.

ROS also provides tools to implement a distributed system, with Services (Nodes) that run on a single machine or distributed over different machines. The communication between individual nodes may be done through a publish/subscription model. Nodes publish (write) and/or subscribe (read) from a topic, which can be local or remote.

Nodes, messages, and services can be organized into Packages, and a collection of similar/related packages that work together can be grouped into Stacks. ROS provides tools to facilitate the management of Nodes, Packages and Stacks, creating a repository of code that can be shared with the research community. Finally, the Player APIs/drivers can be encapsulated as nodes of ROS, in order to provide access to a large set of sensors and actuators available in the market, allowing the user to easily send/receive data to/from different devices. The sensors and actuators can also be replaced by virtual simulated components when using the Gazebo. ROS was adopted as the OS base of our autonomous vehicles, CaRINA I and CaRINA II.

2.2. CaRINA I Hardware

CaRINA I is a Carryall 232 electric utility vehicle (Figure 1(a)). Some of its features include: kinematic and dynamic similar to commercial vehicle, easily modifiable mechanics, and high flexibility to perform tests given the reduced speed, size, and weight. Furthermore, we can highlight the low environmental impact due to electric propulsion.

Mechanical and electronic changes have been made to allow computational control of the steering wheel, acceleration, and braking systems, and preserving the original driving and handling vehicle characteristics.

The steering system comprises a Bosch DC motor and an optical encoder that connects with the steering wheel through a coupling mechanism consisting of pulleys and belts (Fig. 2). It is controlled by a RoboteQ device, which communicates with the computer and the ROS module using a serial port.

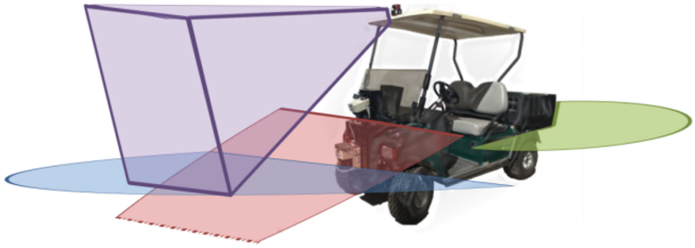


Figure 3: Intelligent Robotic Car for Autonomous Navigation - CaRINA I Test Platform and its Field of Perception.

The vehicle throttle is actuated by an Arduino microcontroller that emulates the signal generated by the throttle pedal. It is also connected to an optical encoder attached to the front wheel to obtain speed feedback. Brakes are operated using a Linak linear actuator.

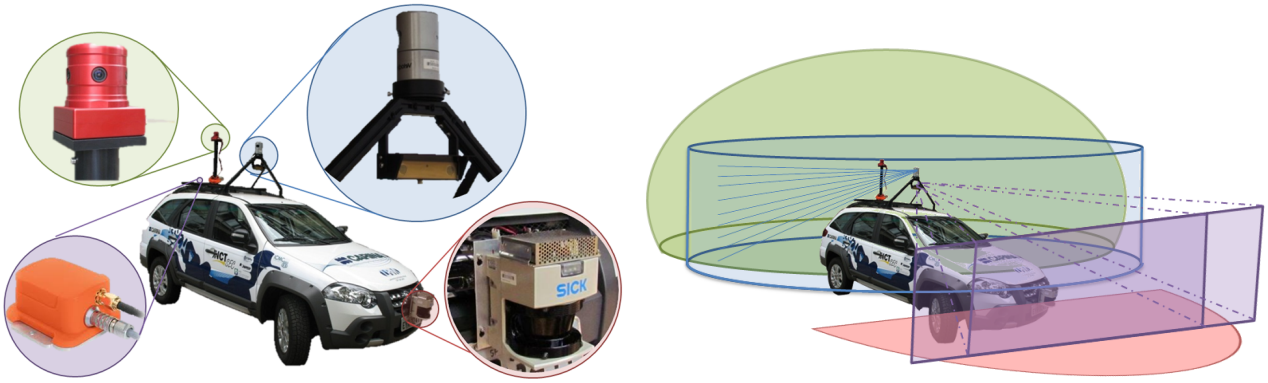
Besides the modifications for actuation (steering, throttle, brake), CaRINA I has also being instrumented with sensors for perception (Fig. 3). It has two Sick LMS 291 lasers in the front, one mounted aligned in parallel to the ground and another one pitched down, pointing at the ground just ahead. Stereo and mono cameras can be mounted either in front or on top of the vehicle. Two other Hokuyo UTM30LX laser range finders are mounted in the back, providing 360 deg of laser coverage around the vehicle. Finally, an MTI-G XSens GPS aided MEMS-based Attitude and Heading Reference System.

2.3. CaRINA II Hardware

CaRINA II is a standard commercial vehicle (Fiat Palio Adventure) modified for computational actuation of steering, throttle, and brake. Its first computational control tests (drive-by-wire) have been performed in the first quarter of 2012. In the second quarter of 2012, the vehicle has been tested at the university campus with fully autonomous control (as described in section 3.1).

Even though the two vehicles are different in several aspects, the CaRINA II modifications followed the basic ideas previously tested with CaRINA I, allowing the reuse of several software components and modules from the architectural design of CaRINA I, including the software modules design based on ROS.

One of the main differences between CaRINA I and CaRINA II is the mechanism used to switch between manual and computer-controlled steering. While CaRINA I uses a mechanical device, CaRINA II system has a computer controlled electromechanical device based on a magnetic coupling system. All mechanical and electrical modifications have been made while maintaining the original aesthetics of the vehicle. From the human driver point of view, the vehicle remains exactly the same as the original commercial car, but with some additional components carefully



(a) CaRINA II Sensors

(b) CaRINA II Perception

Figure 4: Intelligent Robotic Car for Autonomous Navigation - CaRINA II test platform and its Field of Perception.

projected and added to the vehicle in order not to disturb the normal way of conducting the vehicle.

This is very important for the following reasons: the vehicle still preserves its original characteristics, and can be used normally by human conductors in accordance with the public regulations; the vehicle can be used to capture data (logging) from the external sensors but also from the conductor behavior when in human controlled mode; the human can quickly regain the total control of the vehicle, or even overwrite commands from the computer controlled mode. All those features allow for improved security and also ensure that the human can always have the last decision about the actions and commands sent to the vehicle.

CaRINA features standard drive-by-wire throttle system, hydraulic steering, automated gearbox shift, and anti-lock braking system (ABS). It is able to develop much higher speeds than the electric vehicle. Therefore the modification process redoubled security concerns and the use of safety systems already available in vehicle was imperative.

The adaptation of the throttle and brake systems were the first implemented. For the brake system, a Linak linear actuator has been mounted directly on the shaft of the brake pedal. As the vehicle originally features a drive-by-wire gas pedal, we design a system coupling the onboard computer with an I/O interface device and a dedicated electronic circuit, which is capable to control the throttle.

Steering wheel system adaptations take advantage of existing resources in the vehicle. A gearbox has been attached to the steering column using an electromagnetic coupling device. Hidden behind the vehicle panel, this mechanism allows the electronic drive system to couple a Maxon RE40 motor and an optical encoder to the vehicle steering wheel system. Both steering and brake are controlled by a RoboteQ HDC 2450 device.

The current version of the CaRINA II vehicle has a local network of three embedded computers: the first one controlling the RoboteQ and related devices, responsible for low-level steering and speed control (throttle and break); the second one responsible for the sensorial perception devices (Laser, Cameras) and the third one responsible for the user interface system (e.g. data visualization, logging) and overall car management system.

Besides actuators, the vehicle is also equipped with GPS, IMU, cameras (stereo and spherical), and LIDAR (Velodyne HDL32 and SICK LMS). They are used for positioning/localization, obstacle detection, and navigation control (Fig. 4).

A Sick LMS 291 laser was mounted in the front bumper of the vehicle, providing a 180 degrees frontal view with a planar scanning parallel to ground, and covering up to 80 meters ahead of the vehicle. Additionally, a Velodyne HDL-32E was attached on top of the vehicle, allowing covering the entire area around of the vehicle (360 degrees) and ranging up to 100 meters of distance. The top mounted rack also has a PointGrey Bumblebee2 stereo camera and a LadyBug2 spherical camera. An integrated GPS/IMU (Xsens MTI-G) is also available for position and attitude estimation (Figure 4(b)).

2.4. Simulated Model

The CaRINA platforms have been modeled in the Gazebo (Figure 5), a 3D physics simulator compatible to ROS. Its physics simulation is based on ODE (Open Dynamics Engine) that provides rigid body dynamics.

Gazebo specific modelling descriptions are used to plug sensors and controllers to the virtual models and also allow the description of mechanical restrictions. The Ackermann Geometry used to model both CaRINA platforms is an example of that.

We also developed a virtual suspension mechanism that is independent on each wheel [2]. It allows us to simulate the vehicle behavior on uneven terrains, avoiding the kicking collision effect with the ground caused by the rigid body physics simulation. It keeps the four wheels in contact with the ground to avoid loss of traction, slippages and unstable poses of the vehicle that diverges from real situations. Other desired effect from the simulated suspension is the

[2] Video CaRINA I: Ackermann/Suspension http://goo.gl/KqHb3f

(a) CaRINA I (b) CaRINA II

Figure 5: Simulated models of the experimental platforms

balancing of the vehicle when bending, accelerating and breaking, which has a direct effect on the sensor readings, also approximating the simulation to real situations. This gives us a better simulated IMU (Inertial Measurement Unit) data with a more realistic associated noise (Figure 6).

Simulation has been a very important tool for software development in CaRINA Project, and more accurate models of the vehicles and sensors are continuously being developed. Different CaRINA I and II vehicle control and navigation systems implementations have been tested using Gazebo.

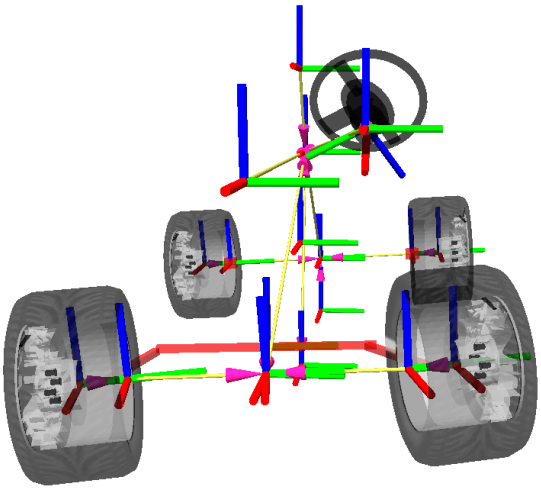


Figure 6: Suspension effect on an uneven terrain

2.5. Computational System Architecture and Modules

The intelligent and autonomous vehicle control software system should provide a series of basic and advanced functionalities while conducting a vehicle into urban spaces. These functionalities range from the driver assistance (ADAS - Advanced Driver Assistance Systems) to vehicle autonomous control and navigation, passing by the possibility of developing Vehicular networks (VANETs) for tele-operation and multiple-vehicles convoys.

The computational system architecture should provide services, as for example: safe remote communication for telemetry and/or teleoperation, reactive behaviors (e.g. keep the vehicle in the road/lane, detect and avoid obstacles and collisions, guarantee the safety for pedestrian and also related to other vehicles), auto-localization, path planning, safe navigation (including safe overtaking maneuvers, lane changing, negotiate crossings, respect the traffic signs and follow/interact with traffic flows). These services required the adoption of a Hybrid Control System Architecture for robotic applications [38], and also, a Distributed System and Service Oriented Architecture [43] allowing the communication among different modules and also among different agents. The Robot Operating System (ROS) is the framework that provided us the infrastructure and facilities to implement a distributed and service oriented architecture, as so as, allowing the development of the modules and components of the proposed hybrid control system architecture.

ROS-based applications should be designed in a modular manner, with parts of the system interacting with each other through messages passing and services exchanges. Custom messages should be specified to represent each type of action commands (e.g. define set-points for steering wheel angle, acceleration and braking), representing

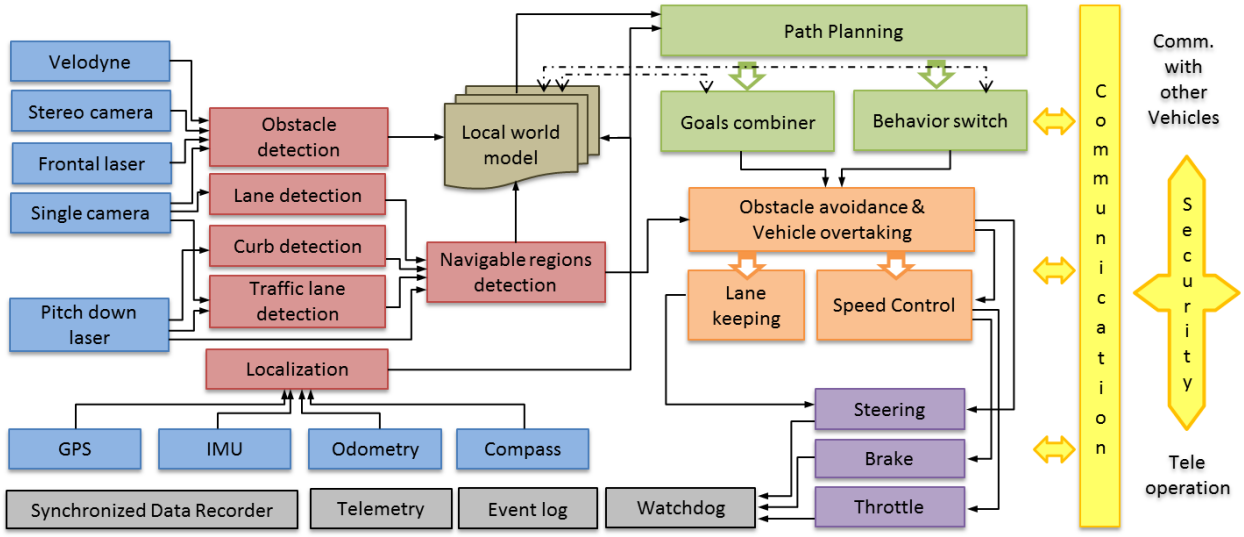


Figure 7: CaRINA's System Architecture. Modules: Sensing (blue), fusion and feature extraction (red), high level judge (green), behaviors in different levels (orange), actuation drivers (purple) and backward system guard (light gray).

the actions independently of the specific robotic platform used. This allows to reuse components, as for example, in both platforms CaRINA I and II, with a minimum of changes.

The CaRINA project defined a computational system architecture and modules, based on the use of the ROS platform, thus creating a reference architecture. This reference architecture, portraying the main system services-modules and the interaction between them is depicted in figure 7. In fact, some services associated to these blocks can be performed by more than one module at ROS system.

Specific modules for each vehicle motor actuator were developed and are responsible for converting the abstract commands (set points) into their specific equivalent values for each platform, producing the desired action (Fig. 7 in purple). These modules are responsible for communicating with control devices of each system and encapsulate low-level protocols, acting as a software driver for the vehicle devices. Furthermore, these modules are responsible for maintaining active the communication channels and guarantee the integrity of the transferred information. The modules of teleoperation and speed control interact directly with them. The teleoperation module allows human operators control each one of these three systems using a joystick, cooperatively (sum of behaviors) or selecting to overwrite (bypass) the commands directly to the vehicle devices, ignoring other modules and commands.

The Speed Control Module (responsible for the functionality of Adaptive Cruise Control - ACC) interacts with the elements of higher levels. It generates the appropriate values for acceleration and braking subsystem according to the desired speed and does it considering also the information about possible obstacles located ahead of the vehicle. The Lane Keeping Module performs the reactive behavior, sending steering commands to the vehicle in order to keep it on the road and inside of the correct lane. At the same level of these modules, but hierarchically dominant, a specific module can inhibit the other reactive behaviors when necessary and performs maneuvers to avoid collisions or to overtake slower vehicles and obstacles. These modules (Fig. 7 in orange) comprise the reactive subsystem of the architecture and are subordinated to the deliberative ones (Fig. 7 in green).

The perception modules (Fig. 7 in red) are responsible for acquiring sensorial data, providing them into a richer representation that allows simple, fast, and reliable identification of relevant environment aspects. Some modules of this same subsystem, localized in a subsequent layer, combine the perceived information for feature extraction and detection of structural elements in scenario. Services like obstacle detection, localization and identification of navigable regions are performed by modules at this level, the results being stored in a representative model of the local surrounding environment. Combining the localization with the world model (previously stored in a global map or currently acquired into a local map), allows us to plan the path and to execute the vehicle navigation task.

Finally, a security and backup subsystem (Fig. 7 in light gray) is needed to maintain critical aspects of safety, records of decisions and operations taken by the system (log), remote monitoring and storage of raw data, allowing more thorough analysis, post-processing or even the creation of databases for simulation. It is noteworthy that some of these elements, especially the higher level services, are continuously being developed.

3. Navigation Systems

We have developed four navigation systems based on the proposed architecture. Though these systems use different approaches, they originate from the proposed architecture and exemplify the variety of navigation systems that can be developed depending on the application and available sensors. This section presents the navigation systems and their experimental results.

The first two systems presented in this section rely mostly on computer vision for maintaining the vehicle on the navigable lane and for avoiding obstacles. The navigation in these systems is guided by a destination point or sequence of waypoints (GPS coordinates). The other two systems rely on a representation of the environment for planning and navigation. One of the systems uses a topologic map for navigation, and the other uses a planning method that considers the model of the vehicle to find suitable maneuvers for obstacle avoidance on an occupancy metric map.

3.1. Supervised Learning for Autonomous Navigation

One of the autonomous navigation systems developed for CaRINA is capable of learning to conduct the vehicle keeping it into the road. This system uses a single monocular camera to acquire data from the environment, a compass for guiding the vehicle and a GPS to locate it. All these sensors are useful for the vehicle to reach destination through a safe path. It also detects the navigable regions (roads) and estimates the most appropriate maneuver. Artificial Neural Networks (ANNs) [28] were used in order to recognize the road and keep the vehicle in a safe path, and finally control the vehicle's steering and speed.

The proposed system [62] is composed by three steps. In the first, a single monocular camera attached to the vehicle captures color images, which are processed in order to identify road lane regions. For this purpose we apply a method based on [59]. This method decomposes the image in many regions also called sub-images, and creates a set of ANNs (committee classifier). The classification of each sub-image, whether it is navigable or not, uses an average of the results of all ANN outputs (committee machine). The result from the first step is a Visual Navigation Map (VNMap) that combines all sub-images. This process corresponds to the Navigable Regions Detection module in the software architecture. In the second step of the system, a template matching algorithm [8] uses the VNMap to identify possible geometries of the road ahead of the vehicle (straight line, soft and hard turn to left/right) and calculates an occupation percentage for each of them. Applying a threshold in these occupation values, the system distinguishes free and obstructed areas based on the VNMap and road geometry. Finally, in the third step of the system, another ANN is used to define the action that the vehicle should take to remain in the road safely (Lane Keeping and Obstacle Detection modules). The supervised learning of this ANN uses the classified image (output from the second step) as input, and the desired controls as output.

Figure 8: Intelligent Robotic Car for Autonomous Navigation (CaRINA) I test platform

Figure 8 shows the CaRINA I vehicle capable of autonomous navigation in an urban road, using the above described approach. In this experiment it was adopted a configuration equipped with a color camera, a Roboteq motor controller for steering, a TNT Revolution compass (orientation) and a Garmin GPS (localization). The image acquisition resolution was set to 320 x 240 pixels, and the road identification subsystem converts this image in a matrix of 32 x 24 sub-image blocks. Each block is individually classified. The learning data that is used to train the ANNs of this subsystem is manually created by a user who classifies parts of one or more scenes as navigable or non-navigable.

Experiments were performed in an open environment in the streets of the USP University Campus. Combining the approaches [63] and [62], several tests were performed in small trajectories [61], and after some improvements, the vehicle performed a path of approximately 1.1 km using as reference seven GPS points defining the path (Waypoints P0 to P6 represent the GPS points). They were more than 100 meters away of each other and delimitated by sidewalks between the points and non-navigable areas. The red dashed line represents the straightforward orientation obtained by both GPS and compass, which should be traversed by the vehicle. The yellow line is the real path followed by the vehicle, showing the obstacles avoidance by using the camera images. We only show the orientation lines between P0 and P1, and P1 and P2 to keep the GPS coordinates image more clear. Figure 9 shows the vehicle trajectory considering the provided GPS coordinates.

The results have shown that CaRINA I was able to identify and stay in the navigable area. In particular, crossings of roundabouts were also done successfully. Even when a virtual straight path, defined by the alignment between the vehicle local GPS position and the destination position, passed through and over sidewalks, vegetation and grass fields, in all those situations CaRINA I followed the trajectory correctly. All over the path the vehicle stayed on the road and reached the destination without going outside the borders of the road (nor bumping or

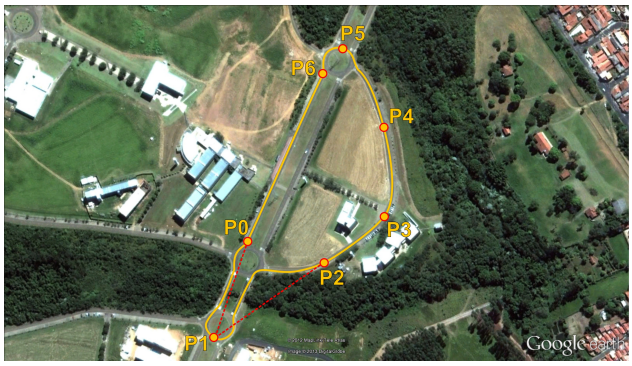


Figure 9: GPS coordinates (P0-P6) defining a sparse waypoint to be followed by CaRINA I and the traveled path.

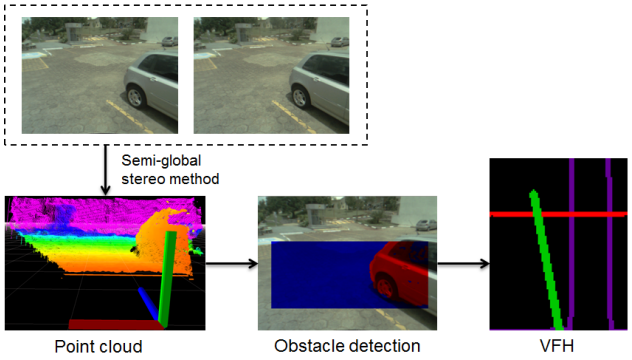


Figure 10: Diagram of proposed methodology.

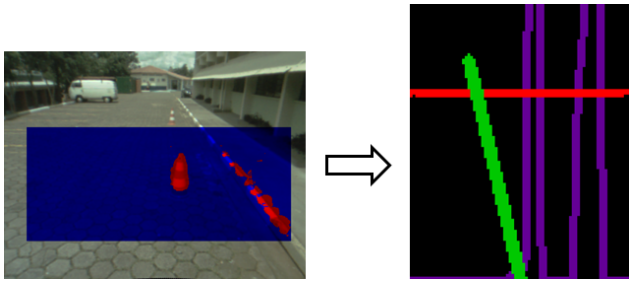


Figure 11: Navigability map and its respective density obstacle histogram.

scraping the sidewalk borders).

The integration of the camera and GPS navigation, using our proposed Supervised Learning method for Autonomous Navigation, was the main original contribution of this work. CaRINA I was able to safely navigate autonomously in urban environments. It did not get too close to the sidewalk or run over it, and also was able to track the GPS points provided to the system. This autonomous navigation system has been used to negotiate the curves and roundabouts successfully.

3.2. Autonomous Navigation using Stereo Vision

Another autonomous navigation system based on computer vision was developed. This one, instead of using a monocular camera, relies on a stereo camera (PointGrey Bumblebee2) to find traversable paths. The goal of the system is to safely drive the vehicle to a predefined GPS coordinate. It comprises a GPS for localization, a stereo camera for obstacle detection and the Vector Field Histogram (VFH) algorithm [7], which was employed for obstacle avoidance.

Due to the use of a stereo camera, a method to extract depth information from stereo images (stereo method) is required. While their results vary in quality and computational cost, the vast majority of navigation systems uses local stereo methods (many of them developed by the own manufacturer of the camera) [42, 58]. We have opted for a semi-global stereo method because it can achieve similar results of global methods while maintaining a relatively low computational cost [30].

The method from [65] uses the depth information (disparity map) provided by the stereo method to estimate obstacles presence and position. It analyzes the point's dispersion according to a conical projection. Considering a range limited by an upper and lower threshold, any giving point that belongs to the same projection is considered compatible. The algorithm premise is: if two points are compatibles they belong to an obstacle.

Finally, the detected obstacles are projected onto a polar histogram. Every point considered as an obstacle is added to the histogram in their respective sector (or angle) and its height or influence is equal to the inverse of its distance (Goal Combiner module). The VFH algorithm searches for valleys in the histogram (traversable paths) and selects the one that is near to the goal direction. We employed this output as the steering angle to the vehicle (Figure 10).

An experiment was conducted where CaRINA I was able to navigate autonomously, avoiding obstacles such as traffic cones, curbs, cars and people; safely reaching the goal. The environment chosen for the experiment was a parking lot at USP, São Carlos campus (Figure 11).

The disparity map precision enabled the detection and avoidance of obstacles as low as 10 cm (curbs), making it suitable for urban environments. The trajectory performed by the vehicle is shown by the map in Figure 12.

It is possible to notice that the car oscillated towards the curb. This is due to the fact that the VFH algorithm does not consider the car kinematics and steering speed. A filter may be employed to remove this oscillation effect, thereby smoothing the car trajectory.

The parameters were selected based on the environment, the presented evaluation and system processing capability. The selected block size for the stereo method was 7 and the disparity range 32.

A possible limitation of the system was its computational performance. The steering angle was updated at a rate of approximately 5 hertz using a resolution of 400x300 in the stereo images pair (the most computational demanding phase) and running on a notebook *Semp Toshiba* model

Figure 12: Vehicle's path (green segment) provided by the MTi-G sensor. Red cylinders are the traffic cones, blue rectangles the cars and yellow circle a person.

1413G with an *Intel Core 2 Duo T6500* processor. To tackle this issue a GPU implementation is under development and the preliminary tests demonstrated that the method can achieve a speedup of 80 when compared to a standard single-core implementation.

3.3. Autonomous Topological Map based Navigation

Recently, some works have described and proposed [36, 41] the use of hybrid approaches based on topological maps and landmarks detection applied to autonomous vehicles. In these topological approaches, the environment is mapped as a graph containing the main elements (topological nodes), in a simple path representation, so a very detailed environment map (metric map) and accurate pose estimation are not required. The vehicle can estimate its approximate position and safely navigate choosing adequate reactive behaviors for each current node. The main task in this topological navigation system is to detect the current node in the topological map, autonomously deciding when and how to go straight, turn left or right, even when more than one possibility is detected simultaneously, at an intersection, for example (Behavior Switch module).

We have developed a topological navigation system that uses an Artificial Neural Network (ANN) [27] to classify the input data obtained from the sensorial system, recognizing patterns on this data which represent the current context (state). For that, an Adaptive Finite State Machine (AFSM) has been developed, integrating the ANN classifier with a traditional FSM, used to represent the state sequence which describes any path at the environment. The ANN is trained to recognize all possible states according to a specific application, and a FSM generator converts any predefined path into a sequence of well-known states to be recognized and followed using the AFSM. The input data was obtained using only monocular images from the Pointgrey Bumbleblee2 camera.

This navigation approach combines the high-level deliberative control (path planning) with different reactive behaviors (each state has its own set of associated behaviors), allowing a safe motion. For example, when the robot is following a straight forward road, the reactive behavior keeps the mobile robot moving forward in the correct lane and restricted to the navigable/safe area. As the vehicle moves along the road, there is a sequence of "local behaviors" defined by the FSM and selected according to each new situation identified, as for example, if there is an intersection, the robot selects a local behavior/action (e.g. crossing the intersection, turn to the left or turn to the right), according to the predefined path described by the FSM.

The developed topological navigation system is composed by three main modules: preprocessing unit, state detection unit and reactive control unit. At the preprocessing step, the input data is converted into a simpler structure, used as input for the both other units of the navigation control system. The State Detection module (ANN) uses input data to detect the present state and the changes in the current state, also filtering oscillations to avoid unexpected state transitions (considering the path plan). The reactive control determines an appropriate behavior (speed control and steering angle) according to the navigable area detection and current state detected. Figure 13 shows the complete system flowchart.

3.3.1. Preprocessing Step

At this initial step, input data is preprocessed in order to identify its main structures and also to remove useless data, such as areas not useful for navigable area detection (above the horizon line). This way, only a subset of relevant information is selected, resulting in a lower amount of input neurons on ANN State Detection module. This preprocessing is useful especially when dealing with sensorial systems which provide high amount of data at each iteration, such as vision-based systems. For example, if all pixels in a 320x240 frame were used as inputs, the ANN input layer would have 76800 neurons, impairing the ANN training process. So, different techniques can be used and combined in order to simplify the environment input representation, allowing a faster training and response time.

Since each state is defined by environment's structural features (environment shape), an interesting way to process the input data is to detect edges and discontinuities in critical areas, creating a training database with patterns based on these features. Navigable area detection is also useful in this case. Image processing techniques can be used for a specific feature extraction, including color depth change or frame resolution reduction, grouping similar elements or even slicing the original frame into smaller pieces.

Figure 14 shows examples of input image frames of some topologic states and Figure 15 shows the result of the preprocessing step of these frames using and edge detection algorithm.

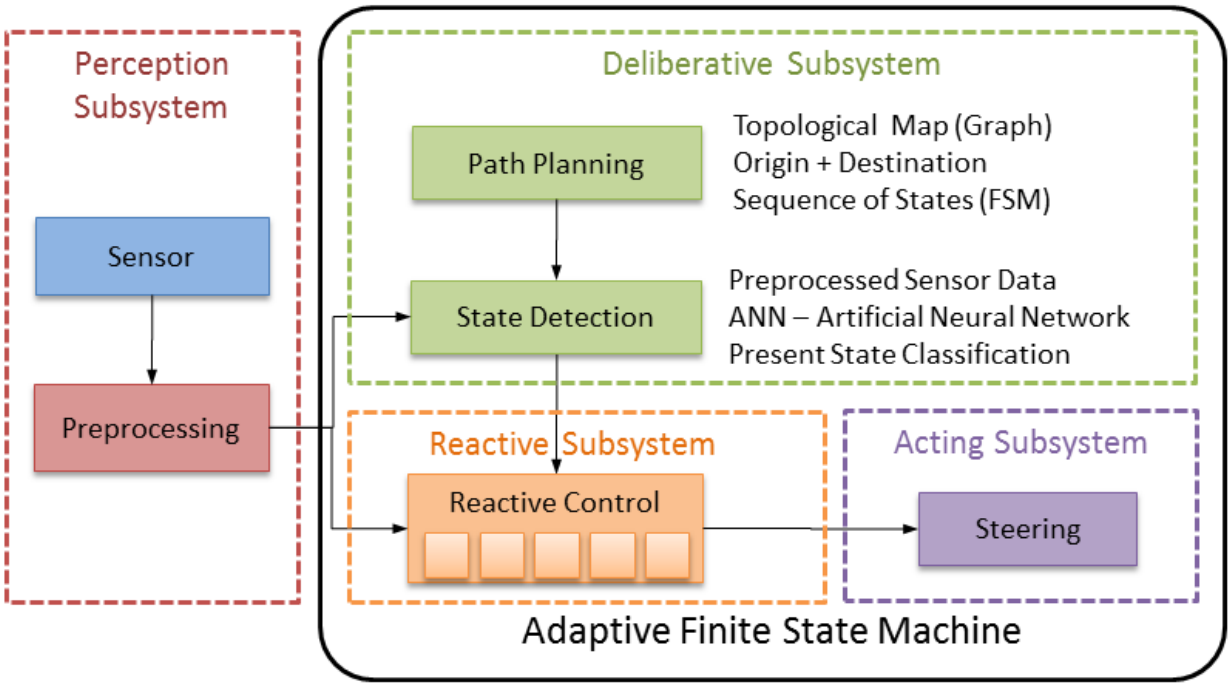


Figure 13: AFSM System overview: FSM + ANN + State/Reactive Behavior/Action

Figure 14: Examples of input frames for representative states.

3.3.2. State Detection

This module uses an ANN to recognize patterns on the preprocessed input data, allowing the detection of environment features used to determine the current state (context). The environment is mapped as a topological map in which each node is related to a specific part of the environment, described by its structural features such as straight path, turns and intersections. All possible states (environment features) are taught to the ANN, so possible paths on this environment can be described as a sequence of these learned states, being represented by FSMs. Figure 16 shows an example of a path map with a topological representation of a region used in our tests.

The ANN input is the preprocessed data obtained at the previous step, and the ANN output is the current state (environment feature) detected. Each different output class is related to a different track shape or condition. In order to begin the training process, a database must be generated collecting a set of examples for each possible situation. So, these examples are preprocessed and labeled, resulting in a set of input/desired-output pairs. After finishing the ANN training, the system should be able to detect all possible states. The combination and sequencing of these states allows specifying different possible paths in this environment.

Once the Topological Map of an environment is given, a route between two points can be established manually or using an automatic path planning algorithm. It is assumed that the robot's initial position is always known, as well as the topological map. The current position is estimated based on current state detection, so it is not necessary to estimate the robot's exact position. It is only necessary to keep track of the topological localization of the mobile robot (approximate localization), as the robot moves from one topological node (present state) represented in the map to the next one (next state). Navigation and self-localization are performed together.

The desired path is also used as input to the State Control unit (FSM), so the system can determine if a state transition is needed or if the robot still remains at the current state. State Control unit is also responsible to filter oscillations in the state detection. Then, an iteration

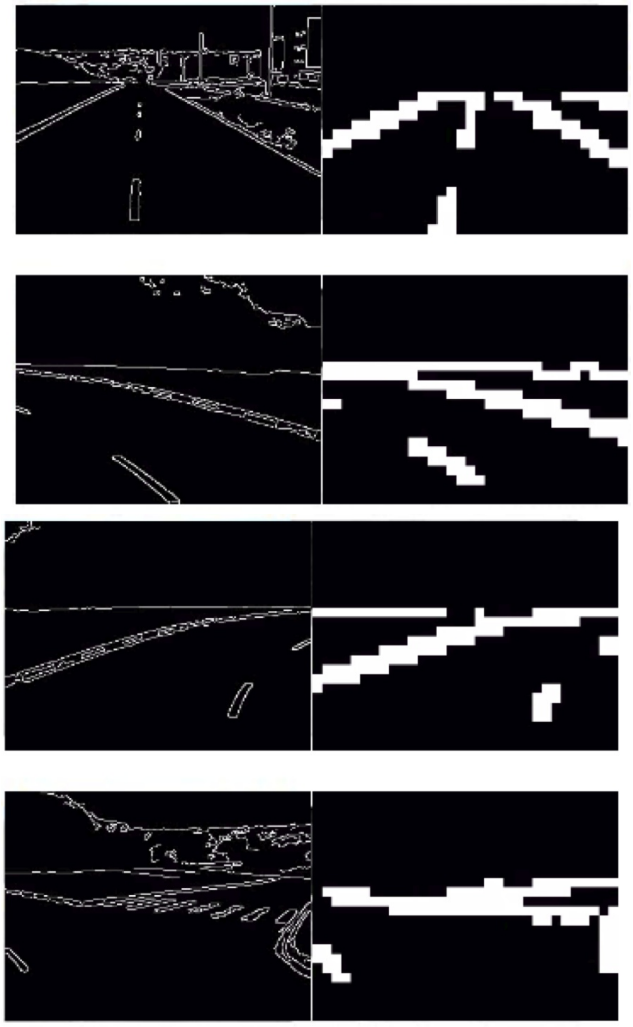


Figure 15: Frames processed with edge detection preprocessing unit.

counter was implemented, allowing a state transition only if the next expected state is observed by a certain amount of consecutive steps. If the observed state does not match the path plan, current state is kept.

Iteration limit must be enough to guarantee a state transition in higher speeds with less similar consecutive captures and also at lower speeds, demanding more similar consecutive captures before a state transition. If for some reason the system fails to detect the current state, localization task crashes. Then, only a reactive navigation can be performed, while the robot seeks for some environment reference points (topological nodes) to re-localize itself and then the path is re-planned. State Detection itself does not control vehicle motion directly. This control is performed by a reactive control unit, responsible for a safe driving through the environment, assuming that an adequate behavior was selected based on the current state. The reactive behavior is also responsible for avoiding obstacles.

3.3.3. Reactive Control

This module also uses the preprocessed sensor data as input, reacting to road current conditions in order to determine steering angle and behavior on decision making. For example, adjust steering to keep the robot at the center of the navigable area in a corridor, or choosing which way to turn at an intersection.

Reactive behaviors have been discussed for a long time, since the first autonomous navigation implementations. There are many different reactive approaches, as for example keeping a determined distance from walls using specific laser beams [56] or adjusting the steering angle after matching navigable area detection with specific templates [63]. Every state can have its own set of reactive behaviors, or the system can be implemented with a single reactive behavior, affected by current state in simpler environments. The adopted AFSM Navigation System is compatible with several existing reactive approaches.

This AFSM-based approach was successfully applied in previous authors' works [56, 54, 53] with different sets of sensors and states, showing the feasibility of this implementation for both indoor and outdoor environments. Tests with the autonomous vehicle CaRINA II were performed with a monocular vision-based sensorial system, and they were presented in [55]. In these experiments, the vehicle correctly followed the tested paths.

3.4. Autonomous navigation with motion planning using vehicle's model

As presented in previous sections, The CaRINA project has achieved successful results applying computer vision for local and topological navigation with obstacle avoidance. In this section we present the autonomous system developed for CaRINA I with global navigation capability using a motion planning algorithm that takes into account the kinematic model of the vehicle to plan feasible paths based on lattice state space and anytime dynamic A* [37].

This approach considers the motion constraints of the vehicle and also deals with obstacles not previously modeled by the map that are detected at execution time. Similar technique has been used in [18] and [32]. However, differently from these works, we generate motion primitives using an integration of the kinematic model instead of using a steering polynomial, and here we consider a three dimension state representation of the vehicle as its pose, i.e., its position and orientation in the Cartesian space.

3.4.1. Lattice State Space

Discretization of the state space is a well-established approach to reduce the computational complexity of a motion planning problem [49]. However, depending on how this discretization is done, it may compromise the satisfaction of the motion constraints which reflects the limited maneuverability of a real vehicle. In order to deal with this problem, the search space known as *state lattice*, was introduced by [49] as a discrete representation that can be

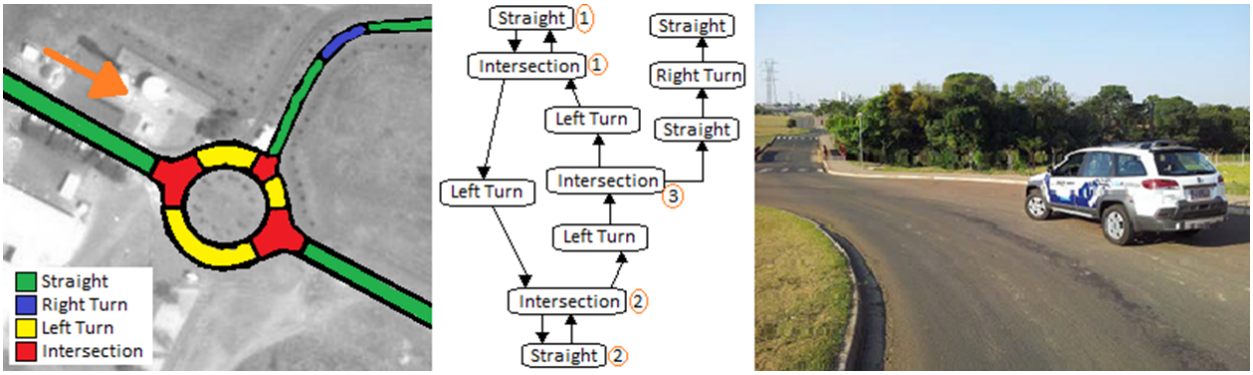


Figure 16: Path map with a topological representation of a validation test region.

used to formulate a non-holonomic motion planning problem as a graph searches.

A state lattice is obtained by the discretization of the configuration space into a set of states and connections between these states, where every connection represents a feasible path, i.e., a path created by taking into account the motion model of the robot. As the construction of a lattice is achieved by connecting states with feasible paths, the resulting connectivity graph reflects the motion constraints of the vehicle. Thus, any solution found using this graph is also feasible. More details of lattice state space representation may be found in [49], [50], [32] and [31].

The state lattice connectivity is defined as a finite subset of actions and states, which includes only trajectories from the origin to somewhere within a distance radius. This subset of actions is composed by short primitive paths generated using the model of the robot. These primitives can be used to generate any path in the lattice state space when they are connected with each other.

*3.4.2. Anytime Dynamic A**

Given a search space, which could be a lattice state space, for example, and a cost function associated to each action performed, we need an efficient method to find solutions in this space for the motion planning problem [32]. Several graph search algorithms have been developed to solve this problem. The A* algorithm and the Dijkstra algorithm are two examples of methods commonly applied to search for optimal paths.

However, planning in a real world is often very complex to be solved within acceptable time constraints. Besides, even when an optimal plan is found, the environment is subject to changes, and at execution time the robot may detect an obstacle which was not considered by the initial plan, thus, replanning may be necessary. For this purpose, an algorithm called AD* (Anytime Dynamic A*) developed in [33] presents great advantages. This algorithm is capable of rapidly search for a suboptimal solution using an inflated A* heuristic, and then whenever there is time available, it improves this solution decreasing the inflation factor and reusing previous searches. Besides, AD* is able to deal with changes in the environment, adapting the current solution whenever a new obstacle observation is made.

3.4.3. Kinematic model and path generation

The kinematic model of the vehicle is based on the Ackerman steering geometry and the model equations are given by

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \frac{v}{L}\tan\delta \end{cases} \quad (1)$$

where $[x, y, \theta]^T$ represents the vehicle's pose with respect to an inertial coordinate system, $\rho = \frac{L}{\tan\delta}$ is the curvature radius of the vehicle, δ is the steering angle of the front wheels, L is the distance between axles, and v is the linear velocity of the vehicle.

In order to generate the state lattice we used here a set of 14 primitive motions, 7 of these are forward motions and the other 7 are backward motions, as shown in Figure 17. As the configuration space used is tridimensional (x,y,θ), the lattice generated by the combination of these actions considers the position and orientation of the vehicle. Each motion primitive were obtained using a Runge-Kutta integration of the kinematic model for one possible control input, $\mathbf{u} = (v, \delta)$. The 14 primitive motions were generated using a subset of the possible combinations of the discrete control variables, $v = [-1, -0.3, -0.1, 0.1, 0.3, 1]$, with values in m/s, and $\delta = [-\pi/6, -\pi/8, -\pi/12, 0, \pi/12, \pi/8, \pi/6]$, by integrating the kinematic model for up to the integration time of $3s$ using Runge-Kutta integration steps of $\Delta t = 0.01s$.

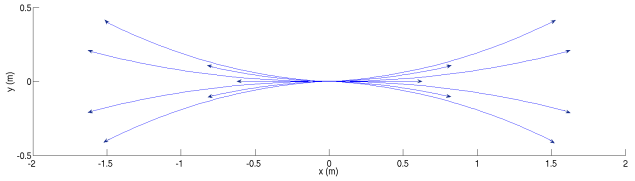


Figure 17: Set of primitive actions used to generate the state lattice.

Given a desired destination state for the vehicle, its current state, and the map of the environment, a state

lattice is created using these motion primitives, and a path is found searching the lattice using the AD* algorithm.

3.4.4. Localization

In this navigation system, the vehicle uses three laser based sensors mounted one in the front of the vehicle and the other two mounted at the rear left and rear right corners to detect obstacles and to map the environment. The signal of these three sensors are integrated and we apply an ICP (Iterative Closest / Corresponding Point) [13] based algorithm for pose estimation. The purpose of ICP is finding a geometric transformation between two point clouds obtained from the laser sensors in different time frames. A Kalman filter combines the pose estimation provided by the three laser sensors with the vehicle's odometry system and an IMU (Inertial Measurement Unit) positioned in the center of rear axle.

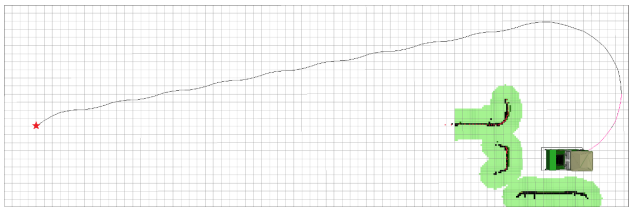
3.4.5. Results

Given a global plan to follow, the vehicle needs to achieve the goal specified avoiding all obstacles in your way safely. For autonomous navigation, the vehicle must perceive the environment in which it is inserted and interact with it. In the tests performed, the vehicle has no information *a priori* about the environment, i.e., the environment is considered fully navigable early in the path planning. Therefore, when the vehicle navigates and detects new obstacles within the range of the laser sensors, these obstacles are mapped by the navigation system. If the current planned route is in collision with detected obstacles, the system must takes an action. A local control method based on the Dynamic Window Approach (DWA) Fox et al. [21] was used to avoid collision with obstacles detected within a close region around the vehicle. And If the route is in collision with obstacles detected outside that window, the vehicle replans the path. More details on how the DWA and global replanning were applied in this navigation system can be found in [37].

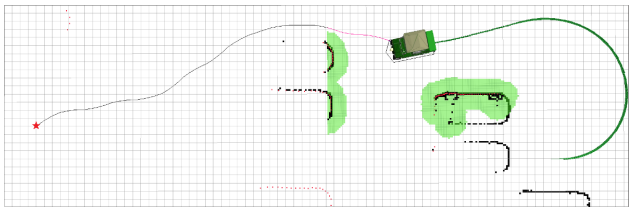
Figure 18 and Figure 19 show two tests, where the first is performed in simulated environment and the second in a real environment. The first test (Figure 18) has been performed such that the vehicle needed to travel a longer distance.

Figure 18(a) shows the planned path represented by the black line and the target by the red star. As the vehicle travels the road, obstacles are found , and as shown in Figure 18(b) and Figure 18(c) a new path is found to deviate these obstacles.

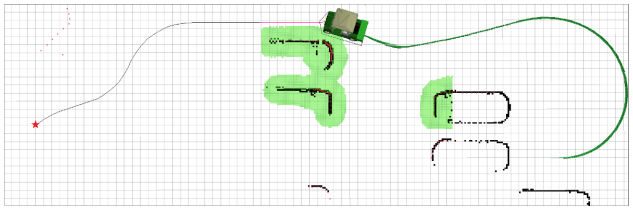
The planning system has also been tested in a real environment where the vehicle traveled a given path avoiding obstacles (Figure 19). The left side of each figure illustrates the real vehicle and the right side shows the trajectory created and obstacles detected. In this test, the driver was responsible for accelerating the vehicle because the vehicle does not have yet a speed control. From a given destination ahead of the vehicle represented by a red star, a path is created and is illustrated in Figure 19(a) by a



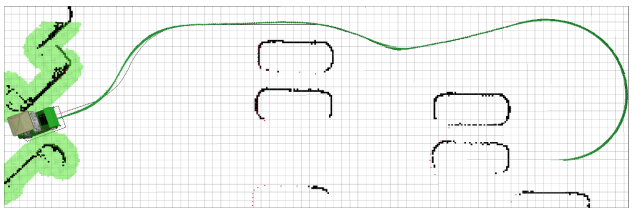
(a) Start navigation



(b) Obstacle Detection and Replaning



(c) Obstacle Avoidance and Replaning again



(d) Goal reached

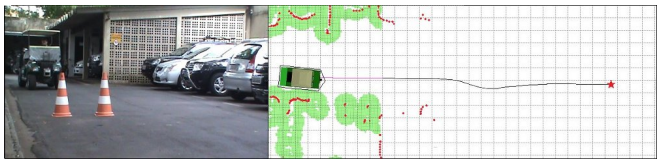
Figure 18: Moving out from a parking spot to diagonal parking

black line. Along the way, the perception system detects obstacles, represented in red, in the trajectory region (Figure 19(b)). The planner immediately finds a new route avoiding these obstacles (Figure 19(c)). In Figure 19(d), a slight deviation from the trajectory can be noticed after replanning, but then the vehicle returns to the desired path. Finally, Figure 19(e) shows the vehicle arriving at the destination, thus completing its task.

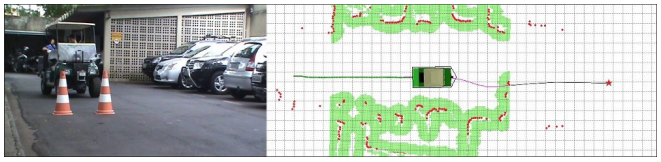
4. Vehicular networks

Vehicular networks (VANETs) and services are an important research topic, driven by safety requirements and by the investments of car manufacturers and Public Transport Authorities. In this way, the efforts of the CaRINA project are in the use of secure communication. Connectivity is one of the main issues in embedded system design, especially in VANETs.

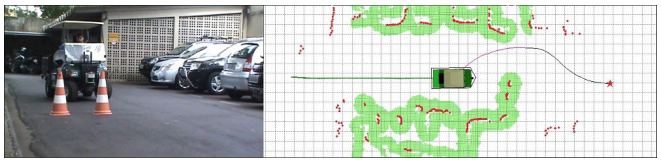
The challenge in VANETs application scenarios is related to the high mobility and the different speeds of the



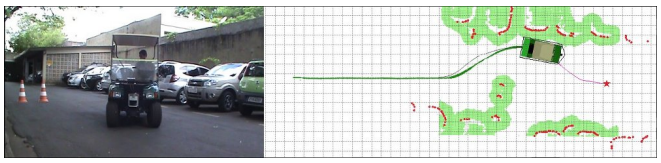
(a) Start navigation



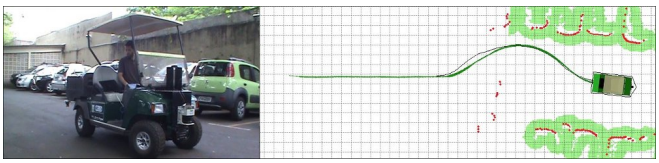
(b) Obstacle detect



(c) Replaning



(d) Obstacle avoiding



(e) Goal reached

Figure 19: Maneuver of obstacles avoiding in a real environment.

vehicles, which causes frequent changes in the arrangement of elements interconnected and thus requires approaches specifically designed for such scenarios.

There are three approaches for VANETs architectures: ad hoc, infrastructured and, hybrid. In the first case (ad hoc) communication occurs directly between vehicles (V2V - Vehicle-to-vehicle communication) and depends on the density and mobility pattern [2].

In the second case (infrastructured), static nodes behaves as access points for IEEE 802.11 networks (V2I - vehicle-to-infrastructure communication) and have the advantages of increasing connectivity and creating the possibility of communicating with other networks, such as, the Internet. However, in infrastructured approach there is the disadvantage of high cost in deployment of network equipment to cover the entire route of the road. In some cases it is possible to reduce costs by deploying a third type of architecture known as hybrid. This last one uses ad hoc communication type associated with a minimal infrastructure to increase network connectivity in service provision.

The choice of these models architectures must be based on factors such as the density of vehicles on the road, obstacles in the path and the applications to which the VANET is proposed. There are three main applications of vehicular networks: traffic safety, entertainment and driver assistance. The increase of traffic safety is a major motivation for the use of vehicular networks, because it generally aims to reduce accidents by exchanging information among vehicles and gathering information about the conditions of the road by sensors. In this type of application, restricted requirements for latency, security and reliability messages are required, and there is a need for greater robustness against insertion of false messages [25].

The convoy of autonomous vehicles is being increasingly used to transport cargo in order to save money and improve safety. It is a scenario that can explore the use of communication, especially the use of VANETs. This has caused automobile manufacturers to invest in proprietary communication protocols for this type of application. These protocols should take into consideration security issues and have been studied and exploited in the CaRINA project. This example illustrates the needs and bases presented here.

Another type of application is entertainment delivery. This type of application is motivated by the increasingly desired Internet connectivity, which moves much of the entertainment on mobile platforms. This kind of service has different requirements than traffic safety applications, because the criticality of entertainment applications is smaller and more failures are acceptable, which allows a lower priority to this application.

Regarding the communication security in the case of entertainment applications using encryption algorithms to ensure the confidentiality of information exchanged may be dispensable. It happens because games and other entertainment applications do not require high levels of privacy. This allows a vehicular network to use different approaches to entertainment and traffic safety applications. It is possible to create different communication channels for each case, applying safer channel dedicated to traffic safety services and less safety delivery channel dedicated to entertainment, for example.

In addition to the applications already mentioned, there is also the driver assistance. This type of service is the delivery of useful information to the driver and can be treated with a priority level higher than entertainment applications and less than traffic safety applications. The most common benefits in the use of driver assistance services may be information about: parking spots, pathways dissemination, traffic control, intersections decision, convoy, location maps, increasing visibility and autonomous vehicles. Failures in such services will not cause injuries to drivers and passengers of the vehicle, but the driver can leave bereft of important information in a travel planning or a package of services that have been contracted by the user, which can involve losses.

Based on the requirements of each application, the road

conditions where a VANET will be implemented and the services it intends to offer for drivers and passengers, generates a plan for a vehicular and better defines the architecture to be adopted. Subsequently, it sets up the routing algorithms best suited for the case, taking into consideration once again, the requirements posed.

The convoy and the driver assistance are the two applications treated by the CaRINA project. Practical tests performed in this study aimed to evaluate some aspects of vehicular networks considering a path of approximately 5.4 km linking the two campuses of the University of Sao Paulo in Sao Carlos, showed in Figure 20.

The first study was performed in the 2 km (marked in red bold in Figure 20). The mode of architecture adopted is infrastructured that despite increasing the total cost of implementing the VANET, was the only option available, since only two vehicles are currently available and adapted to perform practical experiments. It would be necessary to implement a vehicular network ad hoc to increase the density of vehicles on the road, which is not possible under current conditions. A hybrid mode network is planned, but has been not used in the experiments.

Figure 20: A map with the estimated provision of routers (named AP1 to AP19) along the path.

In this experiment two cars go from one point to another exchanging information just between them and getting information from the base station. The cars must follow each other like a convoy. All telemetry and video obtained by cars during the mission will be transmitted to the base station through the VANET. In this experiment the security is necessary as previously shown.

4.1. A Case Study Scenario to Provide Security in intelligent vehicles

The implementation of cryptographic algorithms in computer systems adds an additional layer of enforcement, which therefore increases the response time during the execution of tasks. Security and performance are opposite concepts, i.e. increasing the security level of a cryptographic solution leads to a lower performance and vice versa. This implies the need of a balance between performance and security in accordance with the final application.

When we work with critical autonomous vehicles, the role of security is vital to ensure that these devices can function properly without suffering interventions of malicious entities that may divert them from their routes and/or pre-defined areas of activity. Hence, we took into consideration some common scenarios where normally it is employed this type of vehicle to propose a security solution that ensures the smooth operation of these devices.

An important factor is the delimitation of an area in which the vehicle should operate and move freely within the predefined limits. A system must monitor the position of the vehicle through its location coordinates and the area imposed for their actions. When the vehicle exceeds the limits and does not return within a specified period of time that is normally too short, commands must be sent back to avoid the loss of information or compromise the vehicle if it encounters unforeseen obstacles.

This communication should use authentication mechanisms to ensure that the commands sent from an authorized control station to perform that action (or even a controller that is responsible for monitoring the actions of the vehicle). This is a way to prevent external entities to deliberately shoot commands for changing the direction of the vehicle for its own profit or to cause damage to the devices.

Besides the digital signature used for authentication, there is a need to ensure confidentiality and integrity of information sent in real time to a control base or other vehicles involved in the operation. This implies the necessity of using symmetric and asymmetric algorithms for secure exchange of information.

VANETs need privacy and the security for a successful development. In this way, we propose the use of ECC (Elliptic curve cryptography) algorithm to provide communication privacy and security [48] [57]. It works with key sizes considerably smaller than other cryptographic algorithms and performs operations that demand less computational resources. Nevertheless, the level of security offered by the algorithm with small keys is equivalent to the use of other public key algorithms with larger key sizes. The level of security offered by the 160-bit ECC is equivalent to 1024-bit RSA.

In the intelligent vehicle scenario mentioned above, these algorithms must be implemented and tested on-the-fly. Most studies in the literature perform evaluations with fixed elements communicating with each other.

What distinguishes this project from other works is exactly the movement of the points of communication, which portrays the real scenario of moving vehicles within a predetermined area and the incorporation of all the communication security characteristics. Figure 21 presents a schema of the interaction of the parts in the described situation.

We developed an application that allows communication between cars and a computer (allocated as a server and a ground station), or between car to car. The platform used for application development is J2ME (Java Platform

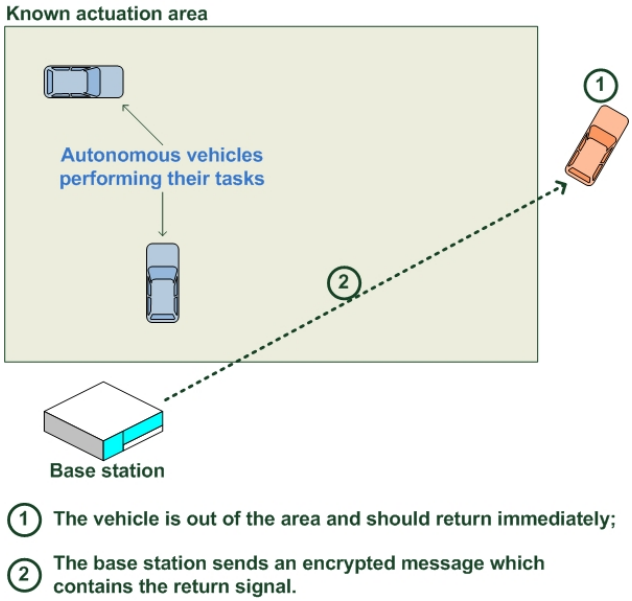


Figure 21: The adopted scenario for the use of cryptography in unmanned ground vehicle.

Micro Edition) technology that enables the development of embedded software and applications and JAVA platform for mobile devices such as mobile phones and PDAs (used to send information to the vehicle). The bouncycastle library has also been used, which already implements routines for using RSA and MD5 algorithms, allowing work with existing digital signature [60].

The devices used in the vehicles are Gumstix Overo EVM Packs, which are equipped with an ARM Cortex-A8, DSP processing, 3D graphics acceleration, Bluetooth, 802.11 (b/g) wireless communications, 10/100baseT Ethernet, high speed USB Host, 8GB of storage and a 4.3" touch screen LCD display.

For communication between mobile device and a microcomputer, it has been used Wi-Fi technology and in some cases Bluetooth technology, the last one implemented with the library bluecove (an implementation of JSR-82 API for computers) and MARGE framework (a framework for Java applications that uses Bluetooth technology).

To create the signature of the communication protocol, we used the RSA algorithm, which uses a private-key to sign messages and a public-key to verify the authenticity. To ensure that the message was not changed during the communication, it was created a cryptographic digest of the message, known as hash. For this project we used the MD5 (Message Digest 5).

The concept behind Bluetooth is providing a universal short-range wireless capability. Using the 2.4-GHz band, available globally for unlicensed low-power uses, two Bluetooth devices within 33 feet each other can share up to 720 Kbps of capacity [64]. The study using all this technology is to provide multiple ways to access the vehicles and supporting multiple ways to avoid the access to confidential information.

For message exchanging, a client application was developed using Java Micro Edition platform and an application server using Java 2 Standard Edition. For the development of the Bluetooth server and client, it has been used the MARGE framework. These two applications allow communication between a mobile device and a computer.

The J2SE (Java 2 Standard Edition) is a development tool for Java. It contains all environments necessary for building and running Java applications, including Java Virtual Machine (JVM), the compiler, APIs and other utility tools.

For transferring data we used the RFCOMM protocol, a simple protocol that supports transporting up to 60 simultaneous connections between two Bluetooth devices. For the purposes of RFCOMM, a complete communication path involves two applications running on different devices with a communication segment between them. The communication segment is a Bluetooth connection from one device to another (direct connection). RFCOMM is only concerned in case of direct connection between the devices, or in case of the network between a device and a modem. RFCOMM may support other configurations, such as modules that communicate via Bluetooth technology on one hand, and provide a wired interface on the other end.

We divide the communication module in two parts: server and clients applications. The server application is responsible for:

- Starting a server that allows connection with the application of the clients;
- Registering each vehicle, it must generate a key pair for the digital signature of each vehicle;
- Receives requests via a socket for each transaction. It must be able to receive and authenticate a transaction data and vehicle-premises, checking the validity of keys;
- This server is responsible for sending transaction data to the clients, so that the data should be signed using the key.

The clients (vehicles) application is responsible for:

- Connecting to the server and to other clients (vehicles) so that it is possible for the clients to receive the transaction data and, if necessary, send its data to other vehicle, by signing the data sent with the key pair for signing and encrypting the message matching the client's password.

Results obtained during the practical experiments do not have great impact when we make a comparison between running on mobile vehicles and on PC. In both environments, the execution routine has been run exactly

30 times, allowing a statistical precision, by simple arithmetic average, eliminating cases in which other factors may have influenced during the experiments. In the first scenario, when running experiments on the mobile device, the response time was 0,4 seconds. This results that our signature algorithm is working in an embedded system and that will not overload and damage the complete communication time.

The communication scheme developed allows the ground station, while receiving a message, to communicate (via the Internet and safely) with the server key pair and public-key to authenticate. If this authentication really happens and the signature is valid, it is proved that the server key pair has authorized the device that sent the message. The communication scheme is presented on Figure 22.

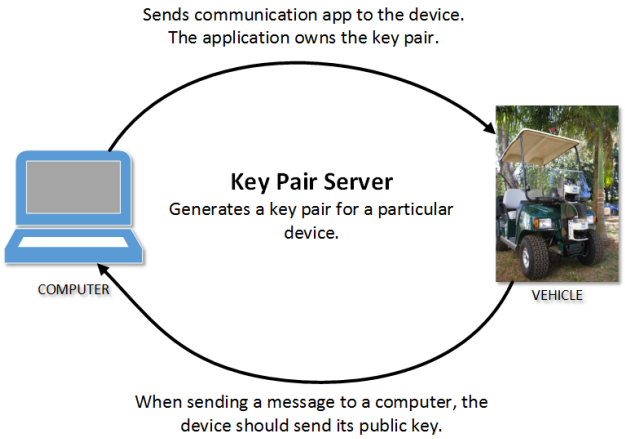


Figure 22: Communication scheme.

To send a message, the device communicates with a computer in order to send a message to the device following the actions below (Figure 23):

- Runs a hash algorithm over the message being sent. As it was used the "bouncycastle" library, it is possible to use different hash algorithms, such as MD5 or SHA1;
- Uses the private-key to sign the hash of the message;
- The device sends to the computer the original message, the signature of the hash, and the public-key.

Upon receiving a message, the computer must run the same hash algorithm used in the device and verify the signature with the public-key. If the signature is valid, the computer communicates with the server of key pair to verify the public-key received from the device that was generated by server (see Figure 24).

Using the digital signature, associated with the use o ECC algorithm, we provide a way to avoid many kinds of attacks like: DoS (Denial of Service), Brute force, Flooding, spooging, etc [47].

Preliminary experiments were carried out to evaluate the conditions of communication between two computers

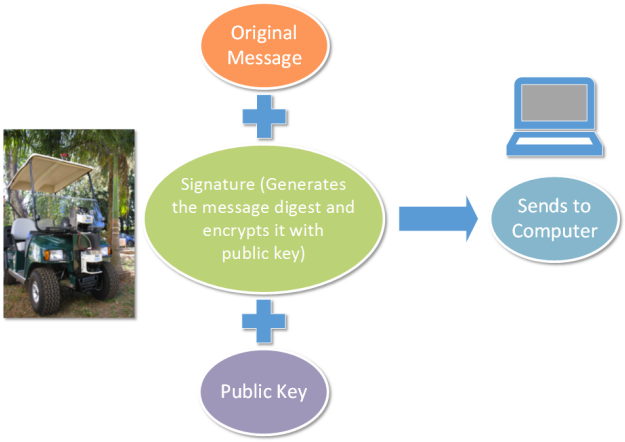


Figure 23: Interaction between car and base station.

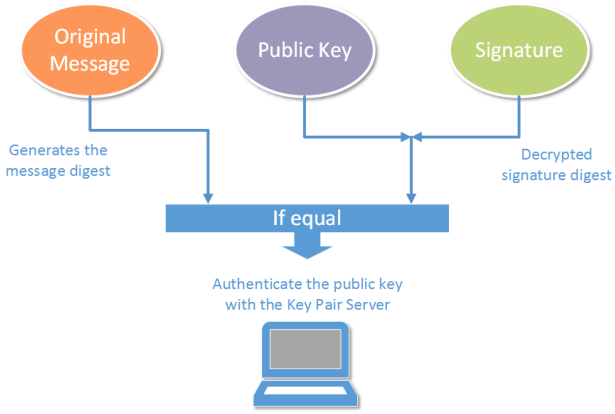


Figure 24: The signature verification.

without external antennas for data transmission. These computers were positioned internally in two small vehicles exclusively responsible for handling the communication nodes. The computers were connected by na adhoc network. The operating system was Linux Ubuntu and ROS software was used for message exchanging. Station 1 was responsible for collecting data from the GPS receiver, immediately sending it to Station 2. This one was responsible for storing the location data received.

In the first stage (a car following the other with about 50 meters between them, transmitting only GPS data), the data was received by Station 2 lossless, except in stretches of the road where the view between the computers was harmed (whether by obstacles on the track or the fact that computers were positioned inside the vehicles). A plot of the GPS data received can be seen in Figure 25.

In the second stage (a car following the other with about 50 meters between them, transmitting GPS data and video simultaneously), GPS data was received by Station 2 with some losses and the video data could not be transmitted due to the low flow of data and the fact that the video does not go through any kind of compression. However, the amount of GPS data received was sufficient to be used in a real application. A plot of GPS data re-

Figure 25: A car following another one with approximately 50 meters between them, transmitting only GPS data.

ceived in this step of the experiments can be seen in Figure 26.

Figure 26: A car following another one with approximately 50 meters between them, transmitting GPS data and video simultaneously.

Finally, the third phase of testing, which was the crossing of two vehicles in opposite directions reaching a relative speed of 100 km/h, the GPS data was received by Station 2 lossless. It is illustrated in Figure 27 a plot of the GPS data received and in Figure 28 it can be observed that at the instant of the intersection (3m00s) there were no data losses. It may also be noticed that in the period from 3m30s to 4m00s losses occurred due to the higher distance and the obstruction of view.

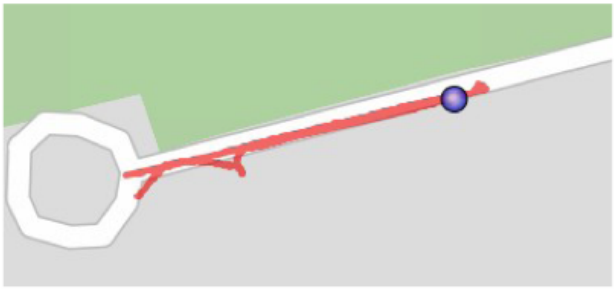


Figure 27: Crossing between two cars at a relative speed of 100 km/h, transmitting only GPS data.

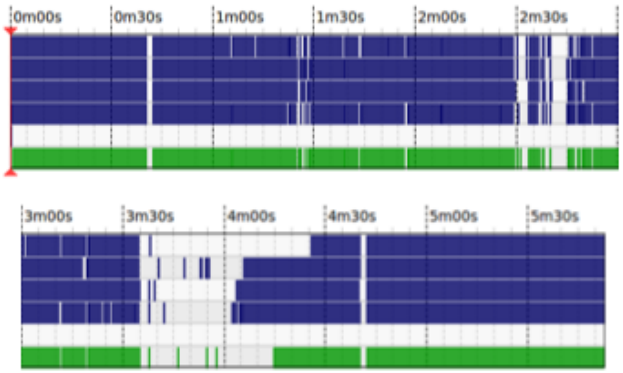


Figure 28: GPS data received by Station 2.

Based on these results, further experiments were planned and performed to evaluate the conditions of communication again between two computers. However, we tried to solve the problems listed in the previous experiments, applying a Wireless USB Adapter (TP-Link TL-WN7200ND 150Mbps 802.11n 1000mW) on each computer with an antenna transmitter/receiver fixed outside of each vehicle. As in the previous experiments, it was created an adhoc network, the Linux Ubuntu was used as operating system and ROS software has been used for message exchanging.

In the first step of these experiments, one car was parked at a starting point and the second one went to the maximum possible distance. The map in Figure 29 shows the location of each car (a distance of approximately 900 m) with no losses in communication.

Figure 29: Location of the cars used in experiments within the Campus II of USP Sao Carlos. The distance between them was about 900 m.

In the second stage, a car followed the other in a random path, occurring in few moments a complete loss of

view. These cases occurred when there were obstacles such as buildings and trees interrupting the view between both vehicles. However, the GPS data was received in satisfactory amount for implementating e.g. a convoy of vehicles.

Using the solutions proposed here we can provide an efficient way to guarantee that the Unmanned Ground Vehicle will be protect against the most common attacks when using VANETs. This solution has also been tested in a real environment where the vehicle traveled 2 km. For further work we intend to travel 5,4 km in an autonomous way, providing video in real time and using a secure transmission both in video and in telemetry.

5. Conclusions and Future Work

The development of self-driving cars is an important research topic that has direct impact in the society. Some of the main goals of this technology are: reducing the number of car deaths and injuries, improving efficiency with freight transportation, fuel usage, and traffic flow in huge cities, providing transportation for children and adults unable to drive like physically handicapped or visually impaired people. It also has direct applications in agriculture, mining and several other activities that heavily rely on mobility.

Although several research groups have been working on autonomous vehicles since 80s, the efforts to develop robust and reliable software algorithms to safely conduct cars in the streets have increased considerably in the last decade. It is mainly due to the increase in funding from government and industry, associated with the development of more sophisticated and accessible sensors and computational systems.

Most research in the field is still concentrated in North America and Europe but few initiatives are taking place in Latin America, where there are several potential applications for intelligent self-driving vehicles.

In this paper we presented the hardware and software architectures of the vehicles used for experimental tests of the CaRINA project under development at the LRM Lab [3]. Both mechanical and electrical modifications have been described, a well as the main software modules for perception, planning, and control.

Our system is based on ROS, which has becoming a standard software tool for robotic applications. It provides libraries, tools, and a message-passing interface for processes that allows distributed robotic control systems to be executed. We also address the security and communications issues in such systems along with a description of the work in progress in these fields.

Finally, four navigation systems, ranging from reactive to a more deliberative approach, are presented to illustrate the use of the proposed software and hardware architectures. The tests have been performed in a variety of scenarios with both experimental platforms.

As future work, we plan to develop and integrate higher level planning algorithms and maps of the environment. It will allow for more robust path planning and longer trajectories. The perception and planning modules are also being improved for more precise and efficient results, allowing faster speed operation of the vehicles in more complex environments.

Acknowledgments

The authors would like to acknowledge the support granted by CAPES, CNPq and FAPESP to the INCT-SEC (National Institute of Science and Technology - Critical Embedded Systems - Brazil) - processes 573963/2008-9 and 08/57870-9.

References

[1] Adams, M., Mullane, J., Vo, B.-N., 2011. Laser and radar based robotic perception. Now Publications, Foundations and Trends in Robotics 1(3), 135–252.

[2] Alves, R. S., Abdesslem, F. B., Cavalcanti, S. R., Campista, M. E. M., Costa, L. H. M. K., Rubinstein, M. G., Amorim, M. D., Duarte, O. C. M. B., september 2008. An experimental analysis of vehicular ad hoc networks capacity (original text in portuguese). XXVI Simposio Brasileiro de Telecomunicacoes (SBrT'08). Rio de Janeiro., 1–6.

[3] Andrade, K. O., Hernandes, A. C., Becker, M., ???? Simulating a neural controller for mobile robot parallel parking (original text in portuguese).

[4] AUTONOMOSlabs, 2012. Artificial vision and intelligent systems laboratory. `http://autonomos.inf.fu-berlin.de/`, visitado em Abril.

[5] Bertozzi, M., Broggi, A., Fascioli, A., 2000. Vision-based intelligent vehicles: State of the art and perspectives. Robotics and Autonomous Systems 32 (1), 1–16.

[6] Bonin-Font, F., Ortiz, A., Oliver, G., 2008. Visual navigation for mobile robots: A survey. Journal of Intelligent & Robotic Systems 53, 263–296, 10.1007/s10846-008-9235-4.

[7] Borenstein, J.and Koren, Y., 1991. The vector field histogram-fast obstacle avoidance for mobile robots. IEEE Transactions on Robotics and Automation 7(3), 278–288.

[8] Bradski, G., Kaehler, A., 2008. Learning OpenCV. O'Reilly Media.

[9] Broggi, A., Bertozzi, M., Fascioli, A., January 1999. Argo and the millemiglia in automatico tour. Intelligent Systems and their Applications, IEEE 14 (1), 55 –64.

[10] Buehler, M., Iagnemma, K., Singh, S., 2007. The 2005 DARPA Grand Challenge: The Great Robot Race, 1st Edition. Springer Publishing Company, Incorporated.

[11] Buehler, M., Iagnemma, K., Singh, S., 2009. The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, 1st Edition. Springer Publishing Company, Incorporated.

[12] Bueno, S. S., Azevedo, H., Mirisola, L. G. B., Paiva, E., Ramos, J. J. G., Victorino, A. C., Azinheira, J. R., 2009. A platform for research and development of outdoor ground robotics (original title in portuguese). In: Simposio Brasileiro de Automacao Inteligente.

[13] Censi, A., May 2008. An ICP variant using a point-to-line metric. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Pasadena, CA.
URL `http://purl.org/censi/2007/plicp`

[14] Desouza, G., Kak, A., Feb. 2002. Vision for mobile robot navigation: a survey. Pattern Analysis and Machine Intelligence, IEEE Transactions on 24 (2), 237 –267.

[3] LRM: Mobile Robotics Laboratory of ICMC/USP - http://lrm.icmc.usp.br (Videos, Publications, Resources)

[15] Dickmanns, E. D., August 1998. Vehicles capable of dynamic vision: a new breed of technical beings? Artif. Intell. 103, 49–76.

[16] ELROB, 2012. Eureopean land-robot trial. http://www.elrob.org, visitado em Abril.

[17] EuroFOT, 2012. European field operational test. http://www.eurofot-ip.eu, visitado em Abril.

[18] Ferguson, D., Howard, T. M., Likhachev, M., 2008. Motion planning in urban environments. Journal of Field Robotics 25 (11-12), 939–960.

[19] Fischer, X.; Bennis, F., 2005. An overview of interactive methods in design and manufacture. In: Fischer, X., Coutellier, D. (Eds.), Virtual Concept Proceedings (Research in Interactive Design) - Biarritz, France. Vol. 1. ESTIA - IEEE - Springer-Verlag.

[20] Fischer, X., 2005. Characteristics, development and use of models in interactive design. In: Fischer, X., Coutellier, D. (Eds.), Virtual Concept Proceedings (Research in Interactive Design) - Biarritz, France. Vol. 1. ESTIA - IEEE - Springer-Verlag.

[21] Fox, D., Burgard, W., Thrun, S., Mar. 1997. The dynamic window approach to collision avoidance. IEEE Robotics Automation Magazine 4 (1), 23 –33.

[22] GCDC, 2012. Grand cooperative driving challenge. http://www.gcdc.net, visitado em Abril.

[23] Google, 2012. The official google blog, 2010. http://googleblog.blogspot.com/2010/10/what-were-driving-at.html by Sebastian Thrun, visitado em Abril.

[24] Gregor, R., Lutzeler, M., Pellkofer, M., Siedersberger, K., Dickmanns, E., 2000. Ems-vision: a perceptual system for autonomous vehicles. In: Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE. pp. 52 –57.

[25] Hartenstein, H., Laberteaux, K., june 2008. A tutorial survey on vehicular ad hoc networks. Communications Magazine, IEEE 46 (6), 164 –171.

[26] HAVEit, 2012. Highly automated vehicles for intelligent transport. http://www.haveit-eu.org, visitado em Abril.

[27] Haykin, S., 1998. Neural Networks: A Comprehensive Foundation, 2nd Edition. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[28] Haykin, S., 2008. Neural Networks and Learning Machines. Prentice Hall.

[29] Heinen, M., Osorio, F., Heinen, F., Kelber, C., 2006. Seva3d: Using artificial neural networks to autonomous vehicle parking control. In: IEEE International Joint Conference on Neural Networks - IJCNN'06. IEEE, IEEE Press, pp. 4704–4711.

[30] Hirschmuller, H., June 2005. Accurate and efficient stereo processing by semi-global matching and mutual information. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 2. pp. 807 – 814 vol. 2.

[31] Knepper, R. A., Kelly, A., 2006. High performance state lattice planning using heuristic look-up tables. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3375 – 3380.

[32] Likhachev, M., Ferguson, D., 2009. Planning long dynamically feasible maneuvers for autonomous vehicles. International Journal of Robotics Research 28(8), 933–945.

[33] Likhachev, M., Ferguson, D. I., Gordon, G. J., Stentz, A., Thrun, S., 2005. Anytime dynamic a*: An anytime, replanning algorithm. In: International Conference on Automated Planning and Scheduling/Artificial Intelligence Planning Systems. pp. 262–271.

[34] Lima, D. A., Pereira, G. A. S., 2010. Stereo vision system for autonomous vehicle navigation into environments with obstacles (original text in portuguese). In: Congresso Brasileiro de Automatica. pp. 224 – 231.

[35] Lima, D. A., Pereira, G. A. S., 2011. Autonomous vehicle safe navigation using vector fields and the dynamic window method (original text in portuguese). In: Simposio Brasileiro de Automacao Inteligente. pp. 1167–1172.

[36] Luettel, T., Himmelsbach, M., Wuensche, H.-J., 13 2012. Autonomous ground vehicles – concepts and a path to the future. Proceedings of the IEEE 100 (Special Centennial Issue), 1831 –1839.

[37] Magalhães, A. C., Prado, M., Grassi Jr, V., Wolf, D. F., 2013. Autonomous vehicle navigation in semi-structured urban environment. In: Proc. of the 8th IFAC Symposium on Intelligent Autonomous Vehicles. Vol. 8. pp. 42–47.

[38] Medeiros, A. A. D., 04 1998. A survey of control architectures for autonomous mobile robots. Journal of the Brazilian Computer Society vol.4, no.3.
URL http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-65001998000100004&nrm=iso

[39] MG, L., Honorio, L. M., Rodrigo, M. A. A., 2010. Modeling path searching for local navigation of an intelligent autonomous vehicle (original text in portuguese). In: Congresso Brasileiro de Automatica.

[40] Miranda Neto, A., Victorino, A. C., Fantoni, I., Ferreira, J. V., 2013. Real-time collision risk estimation based on pearson's correlation coefficient. In: IEEE Workshop on Robot Vision. pp. 1 –6.

[41] Mueller, A., Himmelsbach, M., Luettel, T., v Hundelshausen, F., Wuensche, H.-J., oct. 2011. Gis-based topological robot localization through lidar crossroad detection. In: Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on. pp. 2001 –2008.

[42] Murarka, A., Kuipers, B., October 2009. A stereo vision based mapping algorithm for detecting inclines, drop-offs, and obstacles for safe local navigation. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. pp. 1646 –1653.

[43] Oliveira, L. B. R., Osorio, F., Nakagawa, E. Y., 2013. An investigation into the development of service-oriented robotic systems. ACM SAC 2013 Proceedings 1, 223–228.

[44] Osorio, F., Musse, S., Santos, C. d., Heinen, F., Braun, A.and Silva, A. T. d., 2005. Intelligent virtual reality environments (ivre): Principles, implementation, interaction, examples and practical applications. In: Fischer, X., Coutellier, D. (Eds.), Virtual Concept Proceedings (Research in Interactive Design-Tutorials) - Biarritz, France. Vol. 1. IEEE - ESTIA, Springer-Verlag, Biarritz, France, pp. 1–64.

[45] Osorio, F., Musse, S., Vieira, R., Heinen, M., Paiva, D., 2006. Increasing reality in virtual reality applications through physical and behavioural simulation. In: Fischer, X. (Ed.), Proceedings of the Virtual Concept Conference. Vol. 1. ESTIA - Virtual Concept, Springer Verlag, pp. 1–46.

[46] Osorio, F., Wolf, D., Castelo Branco, K., Pessin, G., 2010. Mobile robots design and implementation: From virtual simulation to real robots. In: Proceedings of IDMME - Virtual Concept (Research in Interactive Design)- Bordeaux, France. Vol. 3. Springer-Verlag, pp. 1–6.

[47] Pigatto, D., Silva, N., Branco, K., 2012. Performance evaluation and comparison of algorithms for elliptic curve cryptography with el-gamal based on miracl and relic libraries. Journal of Applied Computing Research (JACR). 1 (6), 1–9.

[48] Pigatto, D., Silva, N., Sikansi, F., Branco, K., 2011. Applying cryptography and digital signatures for safe communication for unmanned ground vehicles (original text in portuguese). In: II Escola Regional de Alto Desempenho de São Paulo (ERAD-SP 2011). pp. 1–4.

[49] Pivtoraiko, M., Kelly, A., 2005. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. International Conference on Intelligent Robots and System.

[50] Pivtoraiko, M., Knepper, R. A., Kelly, A., 2009. Differentially constrained mobile robot motion planning in state lattices. Journal of Field Robotics 26, 308–333.

[51] Roncancio, H., Hernandes, A. C., Becker, M., 2012. Vision-based system for pedestrian recognition using a tuned svm classifier. In: IEEE Workshop on Engineering Applications. pp. 1 –6.

[52] Sabbagh, V. B., Freitas, E. J. R., Castro, G. M. M. E., Santos,

M. M., Baleeiro, M. F., Silva, T. M., Iscold, P., Torres, L. A. B., Pereira, G. A. S., 2010. Development of a control system for an autonomous car (original text in portuguese). In: Congresso Brasileiro de Automatica. pp. 928 – 933.

[53] Sales, D., Feitosa, D., Osorio, F., Wolf, D., may 2012. Multi-agent autonomous patrolling system using ann and fsm control. In: Critical Embedded Systems (CBSEC), 2012 Second Brazilian Conference on. pp. 48 –53.

[54] Sales, D. O., Feitosa, D., Osório, F. S., Wolf, D. F., 2012. Vision-based autonomous navigation system using ann and kinect sensor. In: Conference Proceedings EANN 2012 – CCIS. Vol. 311. London, UK.

[55] Sales, D. O., Fernandes, L. C., Osório, F. S., Wolf, D. F., 2012. Fsm-based visual navigation for autonomous vehicles. In: II ViCoMoR – IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Vilamoura, Algarve, Portugal.

[56] Sales, D. O., Osório, F. S., Wolf, D. F., 2011. Topological autonomous navigation for mobile robots in indoor environments using ann and fsm. In: Proceedings of the I Brazilian Conference on Critical Embedded Systems (CBSEC). São Carlos, Brazil.

[57] Schoaba, V., Sikansi, F., Pigatto, D., Branco, L., Branco, K., 2010. Disimod digital signature for mobile devices. In: In the proceedings of the ICHIT-2010 - International Conference on Convergence and Hybrid Information Technology, Daejeon. pp. 1–8.

[58] Sermanet, P., Hadsell, R., Scoffier, M., Muller, U., LeCun, Y., September 2008. Mapping and planning under uncertainty in mobile robots with long-range perception. In: Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on. pp. 2525 –2530.

[59] Shinzato, P. Y., Wolf, D. F., 2010. A road following approach using artificial neural networks combinations. Journal of Intelligent and Robotic Systems 62 (3), 527–546.

[60] Shoaba, V., Sikansi, F., Pigatto, D., Branco, L., Branco, K., 2011. Digital signature for mobile devices: A new implementation and evaluation. IJFGCN - International Journal of Future Generation Communication and Networking. 4 (2), 23–36.

[61] Souza, J. R., Pessin, G., Buzogany, G., Mendes, C. C. T., Osorio, F. S., Wolf, D. F., 2012. Vision and gps-based autonomous vehicle navigation using templates and artificial neural networks. Proceedings of the 27th Annual ACM Symposium on Applied Computing, 280–285.

[62] Souza, J. R., Pessin, G., Shinzato, P. Y., Osorio, F. S., Wolf, D. F., 2012. Vision based waypoint following using templates and artificial neural networks. Neurocomputing (Elsevier), Accepted for publication, 1–21.

[63] Souza, J. R., Sales, D. O., Shinzato, P. Y., Osorio, F. S., Wolf, D. F., 2011. Template-based autonomous navigation and obstacle avoidance in urban environments. ACM SIGAPP Applied Computing Review 11, 49–59.

[64] Stallings, W., 2004. Wireless Communications & Networks (2nd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[65] Talukder, A., Manduchi, R., Rankin, A., Matthies, L., june 2002. Fast and reliable obstacle detection and segmentation for cross-country navigation. In: Intelligent Vehicle Symposium, 2002. IEEE. Vol. 2. pp. 610 – 618 vol.2.

[66] Thorpe, C., Hebert, M., Kanade, T., Shafer, S., May 1988. Vision and navigation for the carnegie-mellon navlab. Pattern Analysis and Machine Intelligence, IEEE Transactions on 10 (3), 362 –373.

[67] Veermas, L. L. G., Honorio, L. M., Vidigal, M. C. A. C. F., MG, L., 2010. A methodology for supervised learning applied to intelligent vehicles (original text in portuguese). In: Congresso Brasileiro de Automatica.

[68] Veronese, L., J., J. L. L., Oliveira Neto, J., Forechi, A., Badue, C., de Souza, A. F., 2011. Parallel implementations of the csbp stereo vision algorithm. In: Simposio em Sistemas Computacionais. pp. 12 – 21.

[69] VISLAB, 2012. Artificial vision and intelligent systems laboratory. `http://www.vislab.it`, visitado em Abril.

[70] WHO, 2012. World health organization. Road traffic deaths. `http://www.who.int/gho/road_safety/mortality/traffic_deaths_number/en/index.html`, visitado em Abril.

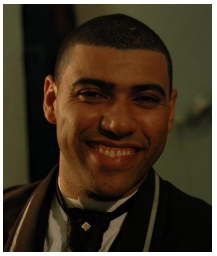
[71] Willow-Garage, January 2013. Gazebo simulator - site: http://ros.org/wiki/gazebo or http://gazebosim.org/. URL `http://gazebosim.org/`

[72] Willow-Garage, January 2013. Opencv - open source computer vision library. site: http://opencv.willowgarage.com/wiki/ or http://opencv.org/. URL `http://opencv.org/`

[73] Willow-Garage, January 2013. Ros - robotic operating system - site: http://www.willowgarage.com/pages/software/ros-platform or http://www.ros.org. URL `http://www.ros.org`

Vitae

Leandro Carlos Fernandes is a PhD student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He also serves as professor in computer science course at Paulista University and obtained a M.S. in Computer Science in 2003 from the University of Sao Paulo. Currently he has been working in autonomous vehicle architecture, low level interfaces and control systems. His current research interests are Artificial Intelligence, Mobile Robotics, Perception and Sensors Fusion.



Jefferson R. Souza is a PhD student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He obtained his MSc degree in Computer Science at the Federal University of Pernambuco in 2010. At ICMC/USP, he has been working in machine learning techniques for mobile robots and autonomous vehicles. More specifically, autonomous driving based on learning human behavior. His current research interests are Mobile Robotics, Machine Learning, Hybrid Intelligent System and Gaussian Process.

Gustavo Pessinis a PhD student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He obtained an M.S. in Computer Science in 2008 from the University of the Sinos Valley. At ICMC/USP,

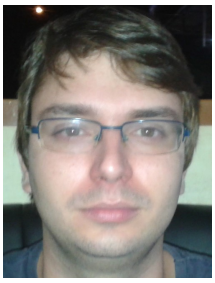
he has been working on several research projects related to robotics and machine learning techniques. His current research interests are Hybrid Intelligent System, Mobile Robotics and Multi-agent Systems.

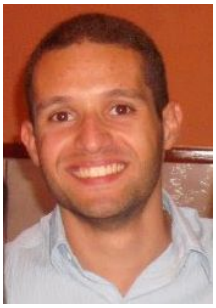
Patrick Yuri Shinzato is a PhD student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He obtained a M.S. in Computer Science in 2010 from the University of Sao Paulo. At ICMC/USP, he has been working in computer vision, machine learning and neural networks. His current research interests are Computational Vision, Artificial Intelligence, Mobile Robotics and Sensors Fusion.

Daniel Oliva Sales is a PhD student at the University of Sao Paulo (ICMC-USP). He obtained a M.S. in Computer Science in 2012 from the University of Sao Paulo. At ICMC/USP, he has been working in Mobile Robotics Lab and his current research interests are Mobile Robotics, Computational Vision and Machine Learning.

Caio Cesar Teodoro Mendes is a PhD student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He obtained a M.S. in Computer Science in 2010 from the University of Sao Paulo. His current research interests are Mobile Robotics, Machine Learning, GPGPU and Computer Vision.

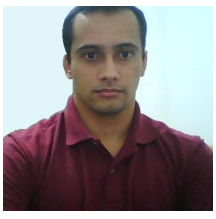
Marcos Gomes Prado is a professor in the Independent Faculty of Northeast. He obtained a M.S. in Com-





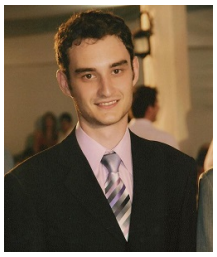
puter Science in 2013 from the University of Sao Paulo. At ICMC/USP, he worked in path planning, sensor fusion and autonomous vehicles. His current research interests are Path Planning, Artificial Intelligence and Mobile Robotics.

Rafael Luiz Klaser, graduated in Computer Science at UNISINOS-RS, is a M.S student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. His main research area are Computer Vision and Visual Navigation. He also has been working with physical simulations in autonomous navigation systems. He also have a background in IT industry and consulting services for many years.



Andre Chaves Magalhaes is a Master's student in the School of Engineering of Sao Carlos at the University of Sao Paulo At EESC-USP, he has working in motion planner and mobile robots. His current research interests is Autonomous Vehicles.

Alberto Yukinobu Hata is PhD student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He obtained a M.S. in Computer Science in 2010 from the University of Sao Paulo. He has been working on Mobile Robotics, Terrain Mapping, Neural Networks and Support Vector Machines. His current research interests are Mobile Robotics, Gaussian Process and Robot Localization.



Daniel Fernando Pigatto is a PhD student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He obtained a M.S. in Computer Science in 2012 from the University of Sao Paulo. At ICMC-USP, he has been working in network security, embedded software development and critical embedded applications. His current research interests are Unmanned Aerial and Ground Vehicles, Network Security and Mobile Robotics.

Kalinka Regina Lucas Jaquie Castelo Branco has Master in Computer Science from University of Sao Paulo (1999) and Ph.D. in Computer Science from University of Sao Paulo (2004). She is currently an Associate Professor of the Institute of Mathematics and Computer Science (ICMC-USP), working in the department of Computer Systems. She has experience in Computer Science, with emphasis on Computer Networks, Security, Embedded Systems, and Distributed Computing.

Valdir Grassi Jr is an Assistant Professor at the Department of Electrical Engineering of the Sao Carlos School of Engineering at the University of Sao Paulo (EESC-USP). He received his Ph.D degree in Mechanical Engineering from the Escola Politecnica at the University of Sao Paulo (EPUSP). His current research interests include autonomous mobile robot navigation, motion planning and control, and computer vision applied to robotics.

Fernando Osorio is Professor/Researcher in the Department of Computer Systems at the University of Sao Paulo (ICMC-USP). He obtained his PhD degree in Computer Science at the INPG-IMAG (Grenoble, France - 1998). Currently he is member of the Mobile Robotics Laboratory at ICMC-USP and also of the Center for Robotic USP Sao Carlos (CRob-USP). His main research interests are Intelligent Robotics, Machine Learning and Computer Vision, with a large number of journal and conference published papers.

Denis Fernando Wolf is an Associate Professor in the Department of Computer Systems at the University of Sao Paulo (ICMC-USP). He obtained his PhD degree in Computer Science at the University of Southern California - USC in 2006. Currently he is Director of the Mobile Robotics Laboratory at ICMC/USP and member of the

Center for Robotics . His current research interests are Mobile Robotics, Machine Learning, Computer Vision and Embedded Systems. He has published over 50 journal and conference papers in the last years.