Noname manuscript No.
(will be inserted by the editor)

A robot architecture for outdoor competitions

Rodrigo W. S. M. de Oliveira · Ricardo Bauchspiess · Letícia H. S. Porto · Camila G. de Brito · Luis F. C. Figueredo · Geovany A. Borges · Guilherme N. Ramos

Received: date / Accepted: date

Abstract Autonomous navigation in unstructured environments is a common topic of research, being motivated by robotic competitions and involving several sets of skills. We present a modular architecture to integrate different components for path planning and navigation of an autonomous mobile robot. This architecture was developed in order to participate in the RoboMagellan competition hosted by RoboGames. It is divided in the organizational, functional and executive levels in order to secure that the developed system has programmability, autonomy, adaptability and extensibility. Global and local localization strategies use unscented and extended Kalman filters (UKF and EKF) to fuse data from a Global Positioning System (GPS) receiver, inertial measurement unit (IMU), odometry and camera. Movement is controlled by a model reference adaptive controller (MRAC) and a proportional controller. To avoid obstacles a deformable virtual zone (DVZ) approach is used. The architecture was tested in simulated environments and with a real robot, providing a very flexible approach to testing different configurations.

Keywords System architecture · Autonomous robot · Robotics competition · Kalman Filters · MRAC · Tracking

Rodrigo W. S. M. de Oliveira
E-mail: rodrigowerberich@hotmail.com

Ricardo Bauchspiess
E-mail: ricardobauchspiess@gmail.com

Letícia H. S. Porto
E-mail: leticiahelenasp@hotmail.com

Camila G. de Brito
E-mail: camilagdebrito@gmail.com

Luis F. C. Figueredo
E-mail: figueredo@ieee.org

Geovany A. Borges
E-mail: gaborges@unb.br

Guilherme N. Ramos
E-mail: gnramos@unb.br

1 Introduction

Autonomous navigation in unstructured environments is a task that brings robotics closer to daily applications like autonomous cars [4, 28, 42, 43]. Robot competitions encourage the development of robotics in such tasks, and one of the better known is RoboMagellan[1] — an outdoor navigation competition which requires the robot to move in an unconstrained and unstructured real-world outdoor environment with different obstacles. In this paper, we expand on a previous work on the development of a robot platform to complete the task proposed by the RoboMagellan challenge [40].

Our robot (Fig. 1) was built on top of a six-wheeled differential platform with DC gear motors. As for sensing hardware, we included eleven ultrasonic sensors, a frontal bar, a camera, a Global Positioning System (GPS) receiver, a 9-axis inertial measurement unit (IMU) and two incremental encoders, one for each side center motor. We used a Raspberry Pi 3 as the central processing unit for controlling the robot and two Arduino Mega 2560 for sensor data and real time processing.

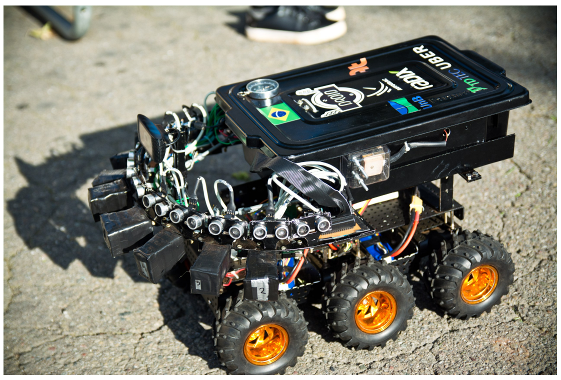

Fig. 1 Bruce, the robot

We developed five modules for our system. A route planner module is responsible for defining the best path between the target points. The localization module fuses the GPS, IMU and odometry data using Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) to provide a global position for the robot in the map. The vision module fuses a vision based target tracking algorithm with odometry through an EKF providing a local localization for the robot. Additional sensing is done by the ultrasonic sensors and the frontal bar, responsible for obstacle detection and target encounter, respectively. The motion control module is responsible for making the robot follow the planned path, using either the information from the localization module or the vision module, all while avoiding obstacles with a deformable virtual zone (DVZ) reactive obstacle avoidance algorithm. All of this is brought together by the navigation module, which selects which information to use and which actions to take.

In this paper, we expand the details on our algorithms, the implementation of each module, and present additional experimental results for specific modules as well as for the complete robot architecture, showing the effectiveness of our approach.

[1] `http://robogames.net/rules/magellan.php`

The rest of the paper is organized as follows. We present related works in Section 2 and describe the robot's kinematic modeling in Section 3. In Section 4 we characterize the system's architecture and give an overview of each module that composes the system in Section 5. In Section 6, we show the experimental results and present concluding remarks in Section 7.

2 Related Works

The literature on navigation, localization and control is vast reflecting its importance on real-world applications [4, 12, 39, 42]. Amid navigation, a more challenging problem is the task of autonomous outdoor unstructured navigation. In such a scenario, large uncertainties, exogenous noises and the unpredictability of the environment—e.g., light variations, terrain conditions, dynamic obstacles, etc—hampers the prescribed plan to be executed, degrades the localization system and removes the reliability of different sensors (high variability of sensor uncertainties). Due to these challenges, autonomous outdoor navigation in unstructured environment is still an open problem being tackled through individual contributions in related areas [11, 14, 15, 19, 20, 22, 30, 41, 45] and through sensor fusion [6, 17, 18, 29, 32, 33] and system integration [3, 5, 21, 26, 35, 44] .

To increase the performance and reliability of the localization system, sensor fusion has been widely used in the literature. Among existing works, it is worth mention [3, 29] for their contribution on Global Navigation Satellite System (GNSS) based sensor fusion and navigation. The work in [29] proposes an adaptive Kalman filter to combine data from the GNSS and the Inertial Navigation System (INS) from the inertial measurement unit (IMU) sensors. Notwithstanding, the amount of information and how to address and pose the estimation fusion is also a relevant problem, particularly in a system with large set of different sensors and reduced computational and consumption power. To address such issue, the work in [3] addresses the problem using a Convolutional Neural Network (CNN) to cope with data from a GPS, a camera and from the odometry readings. This scheme returns a system that selects data between odometry and visual systems avoiding fusion analysis and reducing the computational complexity in run time.

In this work, we take a more general strategy, combining estimation fusion techniques with a high-level navigation framework that ensures reliable performance, modularity and adaptability to different tasks and scenarios. In this sense, the closest work in the literature is the recent work of Valls et al. [44] which proposes and presents a complete system with the integration of key components required for the navigation of an autonomous race car, i.e., system design, EKF–based state estimation, LiDAR–based perception, and particle filter-based estimation. This seminal work is crucial for roboticists and for the developed of the field as only a modicum of papers have formalised a complete description of robot architecture and navigation framework. The current work was inspired by importance and challenges of describing the complete framework of an autonomous navigation system and the gap in the field robotics literature. In this sense, our work also presents a navigation framework that integrates in a fluid manner the motion planning and control with the collision avoidance strategies and the localization scheme. The main difference is the general design where our focus is mostly on the challenges inherent of the uncertain and unpredictability of the terrain and the task, and the required reduced computational and consumption power, in contrast to [44] whose goal is to autonomously complete 10 laps of an unknown racetrack as fast as possible.

In other words, the result herein seeks to analyze and integrates all aspects of a navigation framework in a concise manner providing a complete, easy to follow and implement, and efficient framework for autonomous outdoor navigation robotics.

3 System Modeling

The proposed architecture is based on modules, which work based on a model of the actual robot. Although Bruce has six wheels, its kinematic model was simplified into a two-wheel differential robot, illustrated in Fig. 2, since the wheels on each side rotate at the same speed.

The kinematic model of the robot may be approximated by the direct form [16]:

$$\dot{\xi} = J(p, \gamma, \theta)\dot{p}, \tag{1}$$

in which J is the kinematic model Jacobian matrix, p is the vector of variables that interferes with the robot's movement in the body frame and γ represents the set of geometric parameters of the robot. The pose $\xi = (r_x, r_y, \theta)^T$, in which $(r_x, r_y) \in \mathbb{R}^2$ is the robot's position and θ is the robot's angular orientation, is given in the global frame.

For a differential robot, $p = (\varphi_r, \varphi_l)^T$ is composed by φ_r and φ_l which represents the angles of the traction axes of the right and left wheels respectively. The rotational velocities of the wheels, $\dot{\varphi}_r$ and $\dot{\varphi}_l$, form the $\dot{p}$ vector. The geometric parameters of the robot, $\gamma = (r, b)^T$, represent the radius of its wheels (r) and half the distance between them (b).

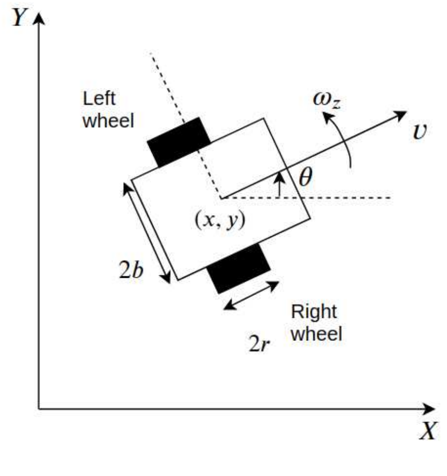

Fig. 2 Differential robot model.

Thus, the kinematic model can be defined as:

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{r}{2}cos(\theta) & \frac{r}{2}cos(\theta) \\ \frac{r}{2}sin(\theta) & \frac{r}{2}sin(\theta) \\ \frac{r}{2b} & -\frac{r}{2b} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix}, \tag{2}$$

in which v is the linear velocity of the robot and ω its the angular velocity [23].

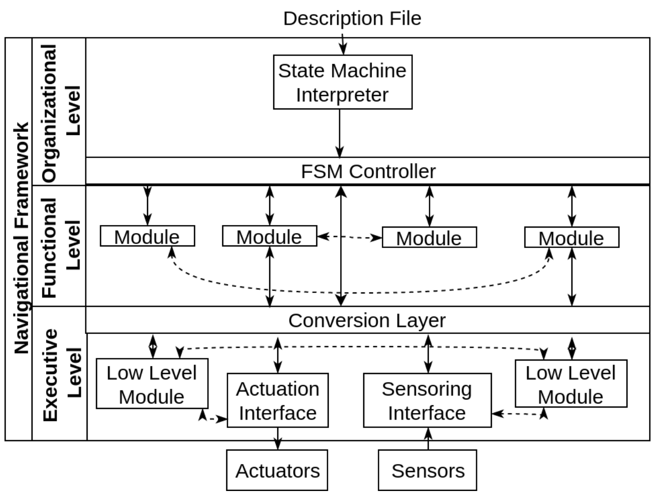

Fig. 3 The Navigational Framework for the system architecture.

4 System Architecture

To help create and organize our system's architecture, we developed our own framework [8, 13,37,38]. An autonomous robot's organizational structure must be able to predict and adapt to the situations it has before itself. It must be able to have a real time response to the events presented, make decisions and take actions in run time [1]; To make all of this possible a system architecture should have the following properties: (i) programmability, (ii) autonomy and adaptability, (iii) reactivity, (iv) consistent behavior, (v) robustness and (vi) extensibility.

The proposed framework helps to ensure the programmability, autonomy, adaptability and extensibility of the system. The other properties must be provided by the actual architecture implemented.

In this work, the framework focuses on the navigation problem due to the nature of the RoboMagellan competition, which emphasizes autonomous navigation and obstacle avoidance over varied, outdoor terrain. We used the Robot Operating System (ROS[2]) and the C++ language to create it. As shown in Fig. 3, the framework is composed of three different abstraction levels working together, each with its own responsibility. The organizational, functional and executive levels work together, communicating through messages.

The organizational level, also known as the finite state machine (FSM) controller, is responsible for orchestrating the robot's behavior. It has two key parts: the FSM description file and the FSM interpreter. The description file is an input file that is read at run time and dictates the robot's behavior using its four defining sections: actions, conditions, states and the finite state machine definition.

Actions execute a task and conditions check if an event happened, so that the FSM can either execute a different action in the same state or trigger a transition between FSM states. These are precompiled code segments whose actual implementations depend on the specific architecture and the lower levels. They are self contained and must be easy to understand and use, which allows us to quickly test and reuse them. The state section simply declares

[2] `https://www.ros.org/`

all states the FSM will have while the FSM definition section describes each state using the predefined actions and conditions.

The description file is interpreted and its information sent to the FSM controller, which orchestrates the modules from the lower level accordingly and keeps track of the robot's current FSM state, which communicates with the modules in the functional level. The modules are responsible for executing the actions and detecting the conditions. Each module declares its actions and conditions to create the library used by the description file. When an action is required by the FSM controller, a message is sent to the proper module requesting its execution. When a condition is triggered, a message is sent from the module to the FSM controller which may cause a change in the current state.

The modules and the FSM controller run in parallel and independently, so each can decide and act on what is needed to accomplish the requested action. If necessary, modules can exchange messages, but these must be predefined and well documented. For example, the trajectory control module receives an itinerary from the route planner module.

The lowest is the executive level, which is responsible for converting the computational abstraction into environmental actuation and vice-versa. It uses three key components: the actuator interface, the sensor interface and the conversion layer.

Both interfaces are responsible for direct communication with the physical sensors and actuators. They provide all the information to the conversion layer, which then transforms it into an abstract representation that is sent to the higher levels. The executive level makes the Navigational Framework really extensible since it is the only part that needs to be updated in case of changes in hardware - the other levels work in higher abstraction and may remain unchanged.

Low level, hardware specific modules, such as a speed controller, are also in the executive level. The conversion layer can introduce small delays in sensor readings, so modules that do not allow any delays belong in the executive level to avoid these delays.

From the Navigational Framework, a specific architecture for the RoboMagellan challenge can be specified, as shown in Fig. 4. The final robotic system for Bruce is composed of 5 modules in the functional level, each detailed in the following sections.

5 System Modules

The modules define how the robot handles specific tasks related to its objective of autonomous navigation and obstacle avoidance in outdoor terrain.

5.1 Navigation Control Module

The navigation control module in the organizational level implements a FSM with eight states shown in Fig. 5, where the initial and final states are colored in green and red, respectively. It uses the actions and conditions provided by the other modules to control the robot's behavior, either by activating or deactivating modules, passing parameters to them and controlling how the system operates (actions) or by launching events triggered by the modules or sensors, representing the changes on the environment or that a module has achieved its goal (conditions).

In the "Wait start button" state, all systems are off and the robot is waiting for a button press to define the system's coordinate system or to start executing the algorithm. The

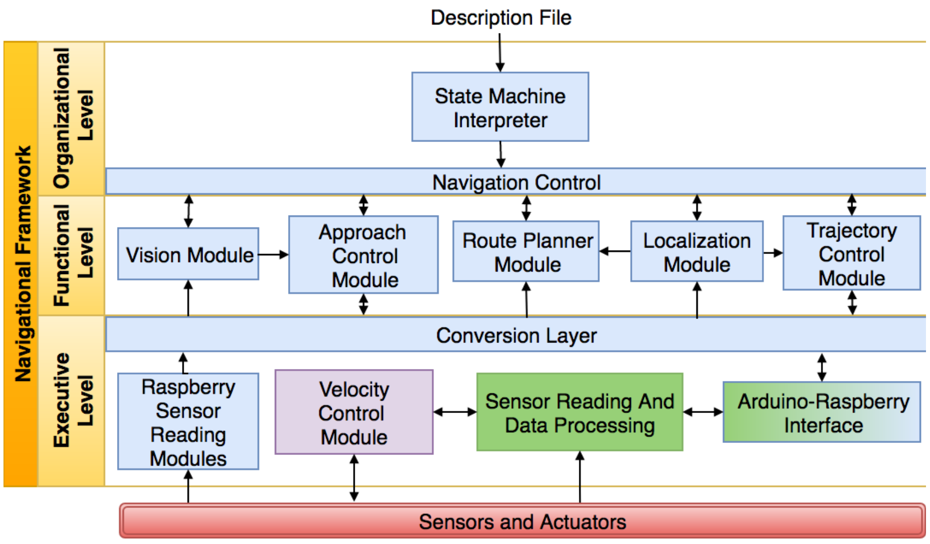

Fig. 4 Bruce's architecture for the RoboMagellan competition. Blue modules are implemented in the Raspberry, green modules are implemented in Arduino number 1 and purple modules in Arduino number 2.

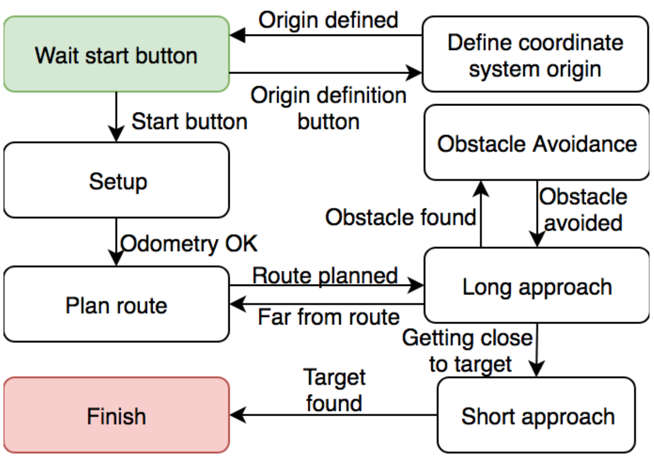

Fig. 5 The robot FSM diagram that describes the behavior the navigation module must execute.

"Define coordinate system origin" state uses the localization and route planner modules to check if the robot is inside the environment's map and define its coordinate system's origin.

The "Setup" state waits for a valid position estimate from the localization module, after which the "Plan route" state requests a global plan from the route planner module to transition to the "Long approach" state. It continuously communicates with the the trajectory control and localization modules in the functional level and the lower levels through the conversion layer to follow the plan.

When an obstacle is detected by the sensors, it switches to "Obstacle avoidance" state. The trajectory control provides reactive obstacle avoidance, which adds the detected objects to the local map so it may be partially adapted to create a new route around them. If this leads to the robot straying for more than ten meters from the global plan, the machine transitions back to the "Plan route" state in order to adapt to the changes - a reactive approach to the dynamic environment.

When the robot detects it is less than five meters from the target, it switches to "Short approach" state which searches for the target using the vision module and moves towards it with the approach control module. The goal is considered to have been reached after the contact sensors are activated and the vision module decides it is reasonable that the contact is with target according to its readings, triggering the "Finish" state which deactivates the robot.

5.2 Localization Module

Autonomous navigation implies that the robot recognizes its position as well as its movement in its surrounding environment. To achieve this, we fused data from GPS, IMU and incremental encoders to obtain the robot's three-dimensional position and attitude. Considering the our nonlinear system, we evaluated the data through an extended Kalman filter (EKF) and an unscented Kalman filter (UKF) [24, 36]. For further details on sensor fusion, readers are also referred [6, 33, 36].

We used the GPS receiver to obtain both the robot's velocity ($\mathbf{v}^n$) and position ($\mathbf{r}^n$) in the global navigation frame North-East-Down (N frame). The IMU is equipped with a triaxial gyroscope, an accelerometer and a magnetometer, and provides the angular velocity (ω_{nb}^b), the specific force ($\mathbf{f}^b$), and the magnetic field ($\mathbf{m}^b$) in the robot body coordinate frame (B frame). The robot's acceleration is then integrated in the standard GPS/IMU fusion to estimate the robot's three-dimensional velocity. The incremental encoders provide the angular velocities of the robot's wheels ($\dot{\varphi}$), which enables the odometry to be computed and, thus, to estimate the robot's two-dimensional velocity.

Experimentally, we observed that the IMU solution errors grew faster than those obtained from the odometry and that the vibration caused by uneven and harsh terrains further increased the degradation in the IMU solution. Thus we opted to use the odometry's two-dimensional velocity instead of the IMU's three-dimensional one.

Therefore, the localization model may be summarized by the following state and measurement equations, respectively:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k,\ \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k), \tag{3a}$$

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k,\ \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k), \tag{3b}$$

where $\mathbf{x}$ is the state vector, $\mathbf{y}$ is the measurement vector, $\mathbf{w}_k$ and $\mathbf{v}_k$ are, respectively, the process and measurement noises, of Gaussian nature with zero mean and $\mathbf{Q}$ and $\mathbf{R}$ covariances, f is the function that models the process and h models the measurement.

The state vector $\mathbf{x}$ and the input vector $\mathbf{u}$ are defined as:

$$\mathbf{x} = \left[\mathbf{q}_n^b\ \mathbf{r}^n\ \mathbf{v}^n\right]^T \tag{4a}$$

$$\mathbf{u} = \left[\omega_{nb}^b\ \dot{\varphi}\right]^T, \tag{4b}$$

in which $\mathbf{q}_n^b$ is the attitude quaternion, $\mathbf{r}^n$ is the position and $\mathbf{v}^n$ is the robot's velocity in the N frame, ω_{nb}^b is the gyroscope measurement and $\dot{\varphi}$ is the odometry measurements.

The function f (Eq. 3a), which models the process [13, 14], is detailed by:

$$\mathbf{q}_k = e^{-\frac{1}{2}\mathbf{W}\Delta t}\mathbf{q}_{k-1}, \tag{5a}$$

$$\mathbf{r}_k = \mathbf{r}_{k-1} + \mathbf{v}_{k-1}\Delta t, \tag{5b}$$

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{r}{2}cos(\theta_{k-1}) & \frac{r}{2}cos(\theta_{k-1}) \\ \frac{r}{2}sin(\theta_{k-1}) & \frac{r}{2}sin(\theta_{k-1}) \\ \frac{r}{2b} & -\frac{r}{2b} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix}, \tag{5c}$$

$$v_{z,k} = v_{z,k-1}, \tag{5d}$$

in which $\mathbf{W}$ is the skew-symmetric matrix of the angular velocities [14]. Equation 5c represents the kinematic model of the differential robot [23], in which r is the wheels' radius and b represents half the distance between them.

Function h in Eq. 3b is given by the identity

$$\mathbf{y} = \mathbf{I}_{10\times10}\left[\mathbf{q}_n^b \; \mathbf{r}^n \; \mathbf{v}^n\right]^T. \tag{6}$$

in which the position and velocity are provided by the GPS receiver and attitude quaternion is estimated by processing the accelerometer and gyroscope readings with the TRIAD algorithm [9].

The whole process is illustrated in Fig. 6. To fuse the data, we evaluated both the EKF and the UKF, in which $\hat{\mathbf{x}}$ represents the state vector estimation and $\mathbf{P}$ represents its covariance matrix. While the EKF is the most widely used in localization problems [7, 17], it may present filter divergence if the errors are not within the linear region. Therefore, we also implemented the UKF [34], and compared the results with the ones obtained with the EKF. The complete algorithm for both filters can be found in [13].

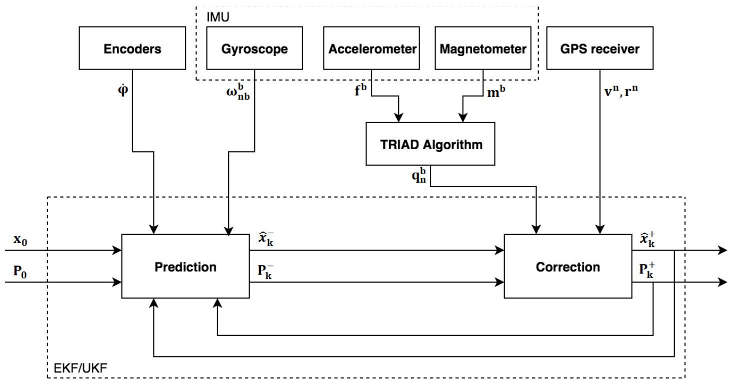

Fig. 6 Localization module diagram

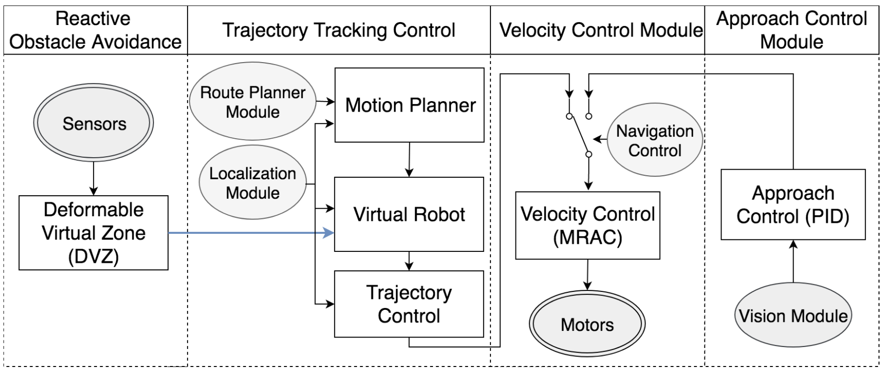

Fig. 7 Motion and trajectory control module's control architecture diagram.

5.3 Motion and Trajectory Control Module

An autonomous mobile robot must try to follow a trajectory in a dynamic environment without hitting any obstacles. Bruce's motion and trajectory control module, presented in Fig. 7, is responsible for this. The module is composed of four layers: trajectory tracking control, reactive obstacle avoidance, velocity control and approach control.

Firstly, in the trajectory tracking control layer, the motion planner generates the path in the shape of line segments, from the route points calculated by the route planner module, to compose the robot's planned trajectory. These segments are input into the reference model for Bruce, called virtual robot, which is a simulation that has the real robot's kinematic model and executes its motion ideally since he moves with a constant speed through the planned path segment and does not suffer any interference in his movement, as if in an ideal environment (no slippage, errors, etc.). Then, the trajectory controller acts by making the real robot follow the movement performed by the reference, based on [23].

The purpose of the trajectory controller is actually to minimize the error between the virtual robot's position ($\mathbf{r}_r$), and the vehicle's actual perceived position ($\mathbf{r}$). The error is computed by:

$$\begin{bmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\end{bmatrix} = \begin{bmatrix}\cos(\theta) & \sin(\theta) & 0\\ -\sin(\theta) & \cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{rx}-r_x\\ r_{ry}-r_y\\ \theta_r-\theta\end{bmatrix}, \tag{7}$$

The trajectory controller calculates the error and provides the velocity vector $\mathbf{Vq}$, that would make the real robot reach the reference model. This vector $\mathbf{q}$ is calculated according to the control laws [22, 38]

$$\mathbf{Vq} = \begin{bmatrix}v\\ \omega\end{bmatrix} = \begin{bmatrix}v_r\cos(\varepsilon_3)+K_1\varepsilon_1\\ \omega_r+v_rK_2\varepsilon_2+K_3\sin(\varepsilon_3)\end{bmatrix}, \tag{8}$$

where v_r and ω_r are, respectively, the virtual robot's linear and angular velocities, v and ω are, respectively, the real robot's linear and angular velocities, and K_1, K_2 and K_3 are positive constants.

The stability of this controller can be verified through the Lyapunov criteria [22, 23, 38]. A candidate function is defined by:

$$V_0(\varepsilon_1,\varepsilon_2,\varepsilon_3) = \frac{1}{2}(\varepsilon_1^2+\varepsilon_2^2)+\frac{1-\cos\varepsilon_3}{K_2}, \tag{9}$$

so that $V_0(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ is always a real positive function and equals to zero if $\varepsilon_1, \varepsilon_2$, and ε_3 are null. In addition, it can be verified that its derivative satisfies the inequality

$$\frac{d}{dt}V_0(\varepsilon_1, \varepsilon_2, \varepsilon_3) = \varepsilon_1 \frac{d\varepsilon_1}{dt} + \varepsilon_2 \frac{d\varepsilon_2}{dt} + \varepsilon_3 \frac{d\varepsilon_3}{dt} \frac{\sin \varepsilon_3}{K_2} = -K_1 \varepsilon_1^2 - \frac{K_3 \sin^2 \varepsilon_3}{K_2} \leq 0. \tag{10}$$

The motion planner also determines which of the path segments should be tracked at any instant, thus it is crucial to correctly identify when to switch from the current segment being followed to the next segment in the plan.

In order to recognize if a path segment λ_i is the current one, a new coordinate axis, X^bxY^b, is defined with its origin at the end point of λ_i, $P_b = (x_b, y_b)$, as shown in Fig. 8. Then we simply transform the robot's position, (r_x, r_y), into this new coordinate system and verify if its position has a negative x^b axis component.

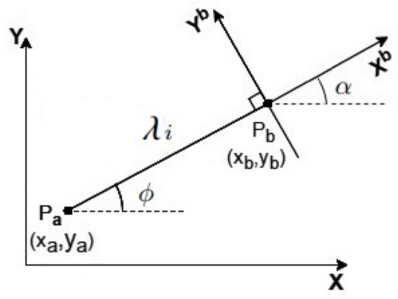

Fig. 8 Path segment being followed.

The point $(r_x, r_y) \in X$xY will belong to the path segment λ_i if

$$\cos(\alpha) * (r_x - x_b) + \sin(\alpha) * (r_y - y_b) < 0. \tag{11}$$

Thus, when the goal point in the current segment is achieved, the new path segment is defined between the robot's current position and the next route point.

The obstacle avoidance module executes in parallel to trajectory tracking control, applying the Deformable Virtual Zone (DVZ) [15, 16], a reactive obstacle avoidance algorithm. This method considers a protective zone around the robot, which is deformed by the presence of an obstacle, as shown in Fig. 9, and allowing it to act in a dynamic environment.

When the protective zone is deformed, the virtual robot's speed calculation is updated to avoid collision [16, 38], as follows:

$$\mathbf{Vq}'_{r,k} = \begin{bmatrix} v_{r,k} \\ \omega_{r,k} \end{bmatrix} = \begin{bmatrix} v_{r,k-1} \\ \omega_{r,k-1} \end{bmatrix} + \begin{bmatrix} K_t f_k \cos(\sigma_k) \\ K_r \sin(\sigma_k) \end{bmatrix}, \tag{12}$$

where f_k is the deformation vector's sum module, σ is its orientation and K_t and K_r are positive constants.

The motion and trajectory control module combines these two layers, following the path segments when no obstacle is present and reacting to avoid one while the route planner creates a new path that considers it.

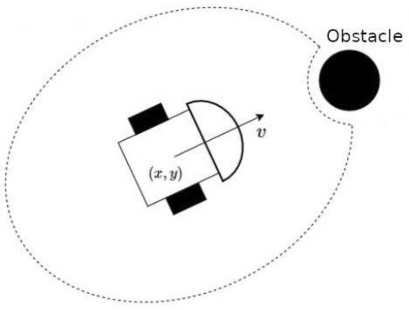

Fig. 9 Obstacle deforming the DVZ region.

When close to the target position, the approach control layer takes charge and a proportional controller receives as input the distance D and orientation ψ of the target relative to the robot in order to regulate its speed [38]. The velocity vector is calculated by:

$$\mathbf{Vq}'' = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} K_d D \\ K_\psi \psi \end{bmatrix}, \tag{13}$$

where K_d and K_ψ are positive constants.

The motion related modules output the velocity vector $\mathbf{Vq}$ in the same format, which is then input into the velocity controller to keep the robot at the reference speed needed to execute the desired path. There are, however, several unforeseen situations in an open environment, which interfere with the robot's motion.

In order to surpass the challenges inherent to the rough terrain and unknown environment, while maintaining an overall satisfactory performance in the competitive nature of RoboMagellan, this module uses a model reference adaptive controller (MRAC) [10]. The general idea of MRAC is to create a closed loop controller with parameters that are updated to change the system response. The output of the system is compared to a desired output of a reference model and an error signal is generated. The controller parameters are updated based on this error through adaptive rules based on Lyapunov stability theory, aiming for the parameters converge to values that make the responses match. This is illustrated in Fig. 10.

The controller design was based on the first-order representation of the robot's propulsion systems in the Laplace domain, given by:

$$G(s) = \frac{\Phi(s)}{U(s)} = \frac{h}{s+a}, \tag{14}$$

which relates the wheels' rotational speed $\Phi(s)$ to the motor's input voltage $U(s)$. h and a are real constants calculated by system identification methods. This model suits both wheels of the robot's model.

Analogously, the reference model is represented by:

$$G_m(s) = \frac{\Phi_m(s)}{\Phi^*(s)} = \frac{h_m}{s+a_m}, \tag{15}$$

which relates the its wheels' rotational speed $\Phi_m(s)$ to the desired input velocity $\Phi^*(s)$. Here, h_m and a_m are real constants chosen for the reference model in order to make it achieve a desired behaviour.

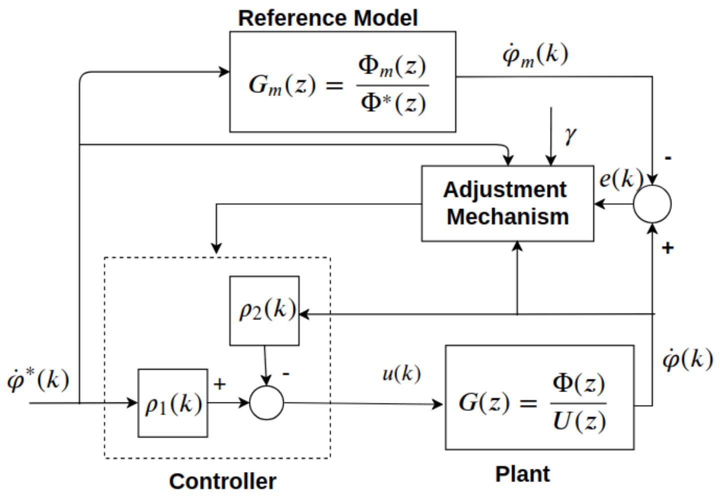

Fig. 10 MRAC's block diagram.

The controller combines a control law with an adaptive law, generating real-time estimations of its parameters [10, 38]. The control signal is set by a linear combination of the input signal $\Phi^*(s)$, the plant output signal $\Phi(s)$ and the controller parameters ρ_1 and ρ_2. Thus, the system follows the reference model with a control law given by:

$$U(s) = \rho_1 \Phi^*(s) - \rho_2 \Phi(s) = \frac{h_m}{h} \Phi^*(s) - \frac{a_m - a}{h} . \Phi(s), \tag{16}$$

These parameters ρ_1 and ρ_2 are adjusted according to the error [10, 38]:

$$e(t) = \dot{\varphi}(t) - \dot{\varphi}_m(t). \tag{17}$$

The adaptive law incorporates stability guarantees, because it satisfies the Lyapunov criteria [10]. Therefore, Lyapunov's function candidate is defined by [10]:

$$V(e, \rho_1, \rho_2) = \frac{1}{2}\left(e^2 + \frac{1}{h\gamma}(h\rho_2 + a - a_m)^2 + \frac{1}{h\gamma}(h\rho_1 - h_m)^2\right), \tag{18}$$

such that

$$\begin{aligned} \frac{dV}{dt} &= -a_m e^2(t) + \frac{1}{\gamma}(h\rho_2 + a - a_m)\left(\frac{d\rho_2}{dt} - \gamma\dot{\varphi}(t)e(t)\right) \\ &+ \frac{1}{\gamma}(h\rho_1 - h_m)\left(\frac{d\rho_1}{dt} + \gamma\dot{\varphi}^*(t)e(t)\right). \end{aligned} \tag{19}$$

With the rules of adapting the parameters given by [10, 38]

$$\frac{d\rho_1}{dt} = -\gamma\dot{\varphi}^*(t)e(t), \tag{20}$$

$$\frac{d\rho_2}{dt} = \gamma\dot{\varphi}(t)e(t), \tag{21}$$

the function $V(e,\rho_1,\rho_2)$ satisfies the Lyapunov criteria

$$\frac{dV}{dt} = -a_m e^2(t) < 0. \tag{22}$$

The outputs of this layer are the control signals, u_k, for each motor to keep the robot's right and left wheels at the desired angular velocities, $\dot{\varphi}_r^*$ and $\dot{\varphi}_l^*$. These can be obtained from $\mathbf{Vq}$ by

$$\begin{bmatrix} \dot{\varphi}_r^* \\ \dot{\varphi}_l^* \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{23}$$

Thus, it is expected that the system follows the reference model with the control laws given by [10, 38]

$$\begin{aligned} u_{r,k} &= \rho_{1,k}\dot{\varphi}_{r,k}^* - \rho_{2,k}\dot{\varphi}_{r,k}, \\ u_{l,k} &= \rho_{1,k}\dot{\varphi}_{l,k}^* - \rho_{2,k}\dot{\varphi}_{l,k}, \end{aligned} \tag{24}$$

in which ρ_1 and ρ_2 are the parameters to be adapted for each wheel according to the following adaptation law

$$\begin{aligned} \rho_{1,k} &= \rho_{1,k-1} - T_v\gamma\dot{\varphi}_k^*(\dot{\varphi}_k - \dot{\varphi}_{m,k}), \\ \rho_{2,k} &= \rho_{2,k-1} - T_v\gamma\dot{\varphi}_k(\dot{\varphi}_k - \dot{\varphi}_{m,k}), \end{aligned} \tag{25}$$

in which $T_v\gamma$ is the adaptation gain.

5.4 Vision Module

The RoboMagellan competition rules establish that the robot must make its way through an outdoor course going to various GPS centered orange cones as waypoints. There are no restrictions to the time of the day a round takes place. To identify and track the target cone for a smoother approximation, a vision module is part of the architecture. Its goal is to minimize the distance estimation error, track time, and be robust to changes in lighting conditions [8]. The maximum distance update time was defined as 100 ms in accordance to the motion module.

The target cone's shape and color were used for its identification in the environment. To this purpose, we adapted the combined single linkage and centroid linkage region growing technique [25] to obtain segments with the target's color with reduced computational cost, while maintaining robustness to lighting and background conditions.

To improve segmentation quality, as a preprocessing step, we applied a sequence of erosion, median and dilatation filters, smoothing segments surfaces and removing bright spots, making the target segment more homogeneous. Applying the median filter prior to the dilatation filter improved bright spots removal in comparison to a median and opening filter combination. We then change the image to the CIELAB color space [27], which is more invariant to lighting conditions.

To segment possible target cones in the image, we first define how similar each pixel is to the target cone's color, according to algorithm 1, in which a return value of 1 is the most similar. We will refer to this value as color similarity for the rest of the paper. The internal

Algorithm 1 Color Similarity

```
1: function ColorSimilarity ($L, a, b$)
2:     $S_L \leftarrow ChannelSimilarity(L, L\_min, L\_min\_optimal, L\_max\_optimal, L\_max)$
3:     $S_a \leftarrow ChannelSimilarity(a, a\_min, a\_min\_optimal, a\_max\_optimal, a\_max)$
4:     $S_b \leftarrow ChannelSimilarity(b, b\_min, b\_min\_optimal, b\_max\_optimal, b\_max)$
       return $S_L \cdot S_a \cdot S_b$

1: function ChannelSimilarity($val, min, min\_optimal, max\_optimal, max$)
2:     if $val < min$ then
           return 0
3:     else if $val < min\_optimal$ then
           return $\frac{val - min}{min\_optimal - min}$
4:     else if $val < max\_optimal$ then
           return 1
5:     else if $val < max$ then
           return $1 - \frac{val - max\_optimal}{max - max\_optimal}$
6:     else
           return 0
```

constants of the algorithm were manually tuned to fit the target cones color. To avoid the cost of repeatedly computing this, each possible similarity value is computed off-line and stored in a map data structure, to be quickly retrieved when necessary during execution.

Afterwards, we sweep the image and whenever we find a pixel with color similarity above a threshold value t_h we obtain a segment starting from that point applying a region growing technique. Empirically, the value $t_h = 0.8$ provided better segmentation for an acceptable computational cost, and was used in all experiments.

In the region growing, a pixel is added to the growing segment if it is adjacent to a pixel in the growing segment, the euclidean distance between these pixels colors is below a threshold t_e and if its color similarity is above a value t_g. Also empirically, we found that the values $t_e = 0.025$ and $t_g = 0.5$ allowed us to obtain the target segment without significant shape loss. This color similarity approach replaces the centroid criteria linkage in [25], avoiding the computational cost of online updates to the centroid value without significant shape loss to the segment. Each pixel added to the segment has its similarity value changed to 0 to indicate it has already been processed.

For each segment obtained, we calculate its principal components (x, y) and align them with the x and y axes. We then regress two lines to the segment right and left limits, given the target's simple trapezoidal form, the segment shape is verified by angle formed between those lines plus the height/width ratio of the segment. If the shape criteria is accepted, we try and measure the targets position in the robot's body frame coordinates.

To get the target's distance to the robot, we first obtain an a priori distance

$$d_{prior} = \frac{H_{target}}{\tan(S_{fv})} \tag{26}$$

in which H_{target} is the targets height in meters and S_{fv} is the segments field of view, obtained by multiplying the segment height in pixels by the field of view of a pixel.

Taking several of these a priori measures in known distances, we obtained a quadratic regression that related these a priori measures to a posteriori measures with better precision. Here we apply another quality criteria: if the distance is above a set maximum distance value, we reject that measure, as its accuracy has reduced confidence. We defined that maximum distance as 15 meters.

The targets angle of view is obtained by interpolating the segments center pixels column in the image horizontal field of view; x and y positions are obtained using trigonometric relations with the targets a posteriori distance and angle of view. The first (x,y) value initializes an extended Kalman filter tracker, using the robot's odometry in the prediction stage. The prediction stage equations are

$$\begin{aligned}
x_{B,k+1|k} &= x_{B,k|k} + T\dot{x}_{B,k|k},\\
\dot{x}_{B,k+1|k} &= -\sin(\omega_z T)\,\omega_z x_{B,k|k} - \sin(\omega_z T)\,\omega_z C\\
&\quad - \cos(\omega_z T)\,\omega_z y_{B,k|k} - \cos(\omega_z T)\,\frac{V_l + V_r}{2},\\
y_{B,k+1|k} &= y_{B,k|k} + T\dot{y}_{B,k|k},\\
\dot{y}_{B,k+1|k} &= \cos(\omega_z T)\,\omega_z x_{B,k|k} - \sin(\omega_z T)\,\omega_z y_{B,k|k}\\
&\quad + \cos(\omega_z T)\,\omega_z C - \sin(\omega_z T)\,\frac{V_l + V_r}{2},
\end{aligned} \tag{27}$$

in which C is the distance from the camera to the center of the robot, V_l and V_r are the left and right wheel speeds, respectively, and T is the elapsed time between predictions.

Using the inverse procedure to get x,y position from a segment, we obtain the predicted bounding box for the target in the image. Using the predicted x,y position variances, we define $x_{min}, x_{max}, y_{min}, y_{max}$ values, which we use to define a larger bounding box, limiting the positions in which we expect the target to be found in the image. This second bounding box we use as region of interest in which we apply the vision algorithm measuring the targets position.

As a way of controlling the vision algorithm efficiency, we defined a maximum image region size to be processed by the algorithm and resize regions of interest bigger than that size to it. To get our desired 10 Hz we defined that maximum size to 15000 pixels, which translated to the target being roughly 2 meters away from the robot. The big difference between this value and the maximum accepted distance shows that we could easily increase the algorithm frame rate if necessary.

Before applying the Kalman filter update step, we evaluate the quality of the vision measurement. We do so by calculating the Mahalanobis' distance between the predicted position and the measurement position. If the distance is greater than 5.991, the 95% in the χ_2^2 table, the measure is discarded, otherwise we apply the update step of the Kalman filter. If five successive measures are discarded this way, the tracking is abandoned, and the vision algorithm starts anew, searching the entire camera frame.

The x and y position values obtained after the update step of the Kalman filter are then transformed to a distance and an angle value, which are sent to be used by the motion module. The detailed algorithm can be found in [8].

5.5 Route Planner Module

The route planner utilizes a bi-dimensional cost map generated by a geo-referenced image obtained from satellite images. To provide some a priori knowledge about possible obstacles like fences, buildings, and others, these elements are manually marked in the map and the remaining parts being considered as free space. The resulting cost map is given as input to the Rapidly-Exploring Random Tree (RRT) algorithm [31]. We exploited standard RRT planning strategies with local planner given by line segments which were tracked by the real-time adaptive controller.

The cost map is transformed into a configuration space $\mathscr{C}$, as illustrated in Fig. 11 where the obstacles are represented in black. The algorithm creates a route by randomly exploring a tree structure representing the space. The search tree grows as new positions are added, with obstacles forming leaf nodes, until a position close enough to the goal is reached.

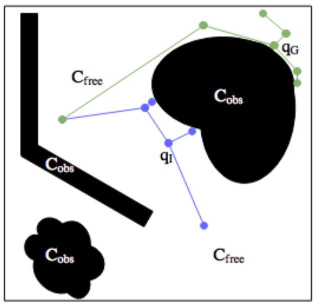

Fig. 11 $\mathscr{C}$ configuration space, represented as a continuous cost map.

Although this not an optimal algorithm, it provides good enough solutions very quickly, which is a better approach to work in a dynamic environment. The choice for a simpler solution yet quick solution was due to the nature of the autonomous navigation task in outdoors, that is, the large uncertainties within the environment. Implementations using B-splines, C-splines and even advanced geometrical Interpolation strategies could smooth the trajectory to the robot kinematics yet they would still be highly dependent on the adaptive control solution in real-world applications. For better comprehension of the interpolation complexity in this scenario, and possible improvements in planning (in exchange of additional computational complexity), the authors refer the reader to the excel work of Allmendinger et al. [2].

6 Experimental Results

This section presents a series of real-robot experiments in unconstrained outdoor environments. The tasks presented herein aimed at validating the proposed robot architecture in such complex scenario and to evaluate the performance of different modules from the system framework. This section is organized by trials and quantitative assessment from different modules. Lastly, we also included a simulated task to analyze the performance of the closed-loop system stemming from module modifications in a controlled simulated environment.

6.1 Localization module

The localization module was evaluated by performing tests in which the robot was remotely controlled to perform closed paths, measuring the distances between the start and finish points to assess the algorithms. Due to unforeseen communication delays from the GPS receiver, which degraded the efficiency of the GPS data in real-time, only the attitude filter was active and the velocity and position values were estimated using odometry. The filter was able to compensate the lack of GPS data, as shown in Fig. 12 — that is, the shape and distance travelled after the filtering is similar to the real ones travelled by the robot, the final point being close to the origin, the expected finish point.

Fig. 12 shows a comparison between the EKF and the UKF approaches for one of the tests, in which we may observe that the UKF performed slightly better than the EKF, with the final point being closer to the origin. Further testing for comparing of the computational cost in real-time applications is undergoing.

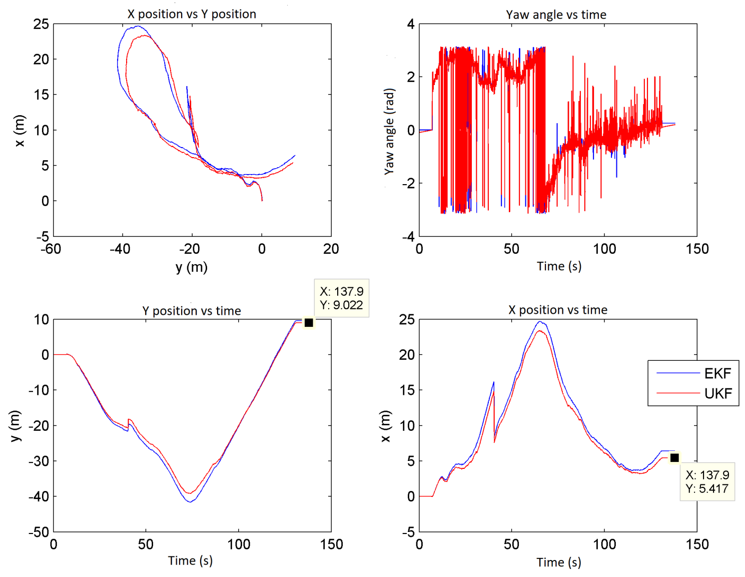

Fig. 12 EKF and UKF for attitude filtering in a closed path

6.2 Vision Module

The robustness of the vision module when dealing with unstructured environments was evaluated by performing several target identification tests in different conditions. Ambient illumination is a key factor, so we tested the algorithms in different situations, such as night time, as illustrated in Fig. 13. Despite the unfavorable conditions, the module successfully identified the target.

The target's surroundings might also pose a difficulty, so we experimented in conditions where the target might be camouflaged, as shown in Fig. 14. The module also identified the target successfully in this case.

To assess target tracking, we performed a short experiment in which a hard turn is forced (manually, via remote control) while the robot is tracking the target. The code was adapted so that if the algorithm failed to detect the target within the region of interest, a second attempt would be performed on the entire image, giving us graphic information of where the object should have been found. Fig. 15 shows the estimated trajectory for this experiment, where pink circles indicate the detection considering the whole image. Fig. 16 shows the

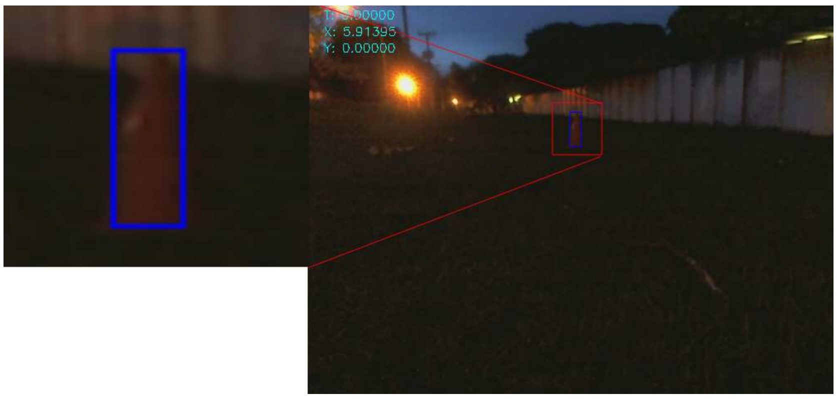

Fig. 13 Frame with target identification after sunset. Zoomed in on top left corner.

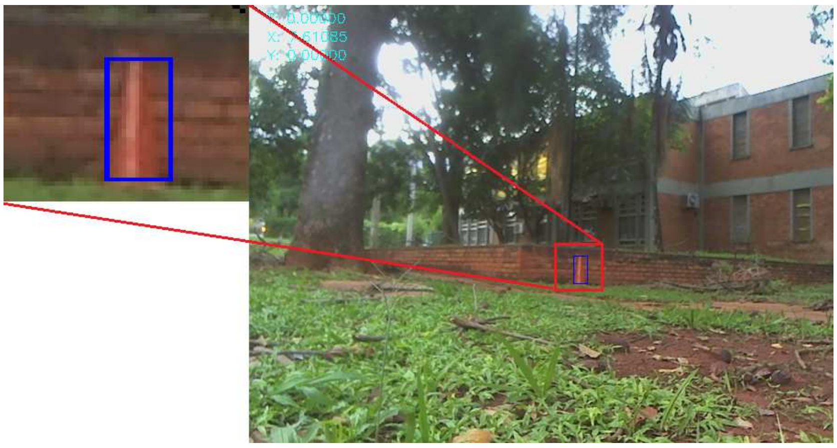

Fig. 14 Frame with target identification. Zoomed in on top left corner.

Mahalanobis distance calculated for this, where the red line indicates the distance threshold defined in subsection 5.4. The red and pink circles in Fig. 15 are the points for which the Mahalanobis distance was above the defined threshold. As may be observed in those images, after five frames without valid measures, the tracking error is identified and corrected. This experiment shows the importance of the Mahalanobis distance test in detecting errors in the tracking and guaranteeing that the target is being correctly tracked.

Using our standard algorithm we performed a new batch of experiments. Fig. 17 presents the results for one of these experiments, illustrating how the filter smooths the measurements and improves the final accuracy of the odometry.

6.3 Motion and Trajectory Control Module

To evaluate Bruce's propulsion systems, it was suspended so that the wheels did not touch the ground and a reference speed was provided for them.

Fig. 18 depicts the time-response and convergence of the measured robot's wheels velocities (in black) compared to the reference velocity input (in gray). It is clear that the speed

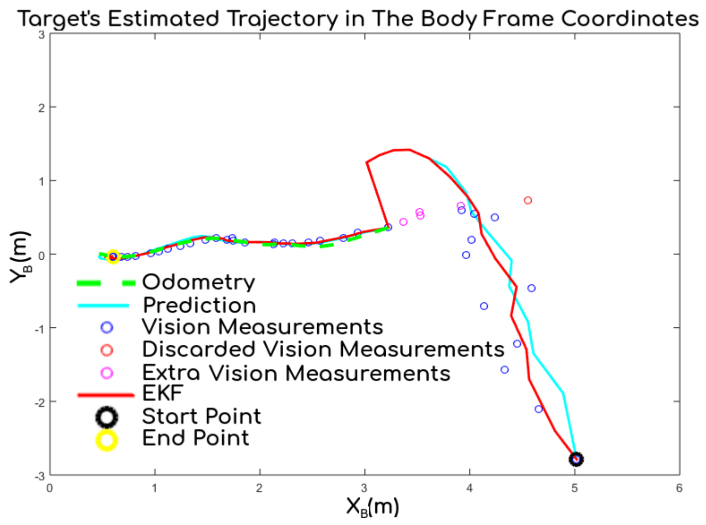

Fig. 15 Target's estimated trajectory by the vision module with tracking correction

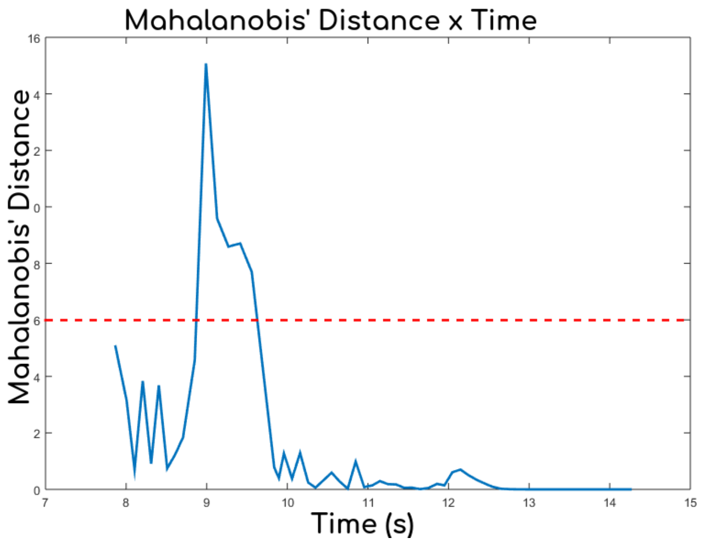

Fig. 16 Mahalanobis' distance for the experiment

controller presented negligible error in steady state. The transient state, on the other hand, was slow and slightly oscillatory, as expected for the chosen control methodology. The delay presented in Fig. 18 reflects the internal robot dynamics and the response time of the closed-loop system—in other words, it is the mobile robot closed-loop inherent delay required to achieve a given velocity from an acceleration input.

The errors between the wheel speeds and their references is shown in Fig. 19. Since the robot is kept at the reference speed during the steady state, these results show that the MRAC speed controller provides a satisfactory performance.

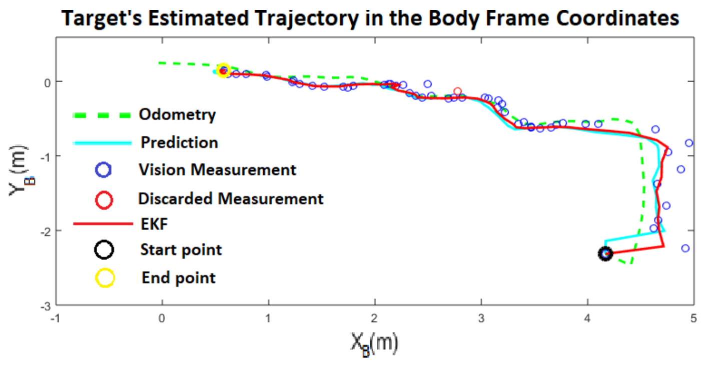

Fig. 17 Target's estimated trajectory by the vision module.

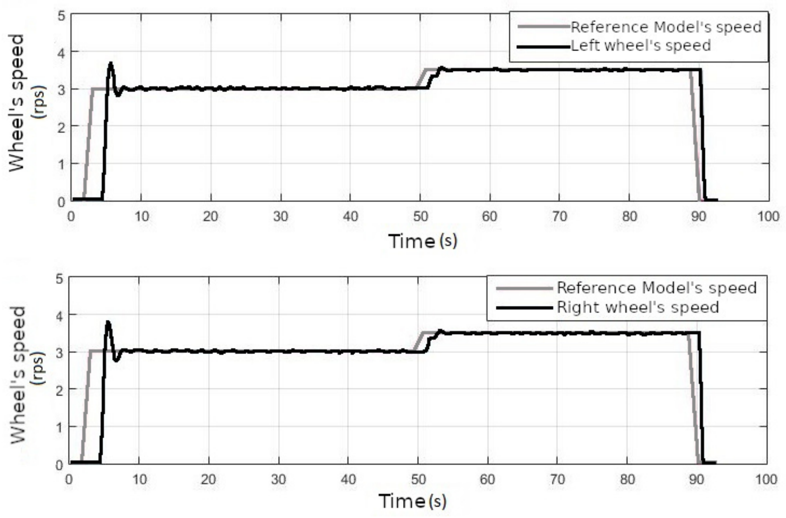

Fig. 18 Velocity of the robot's wheels with respect to the trajectory controller based on MRAC

6.4 Route Following

To evaluate the "Long approach" state, we suspended the robot and used only its odometry to estimate its position, as shown in Fig. 20. The black areas are obstacles, the route planner's path is magenta, the virtual robot's course calculated by the motion planner is in blue and the real robot's estimated path is red.

In the experiment, Bruce followed the first seven segments as planned, yet uncertainties in the terrain pushed the system out of the prescribed trajectory in 8th segment. The system then planned a new route (segments 9 and 10) to adapt to this situation and successfully reach the goal—even with additional uncertainties in the final segments. Note also that the non-smoothness in the transition between segments, particularly, between 3 and 6, refers to the planning strategy based on standard RRT. It is however noticeable the effectiveness of the adaptive controller in completing the curved trajectory satisfying the robot non-holonomic constraints.

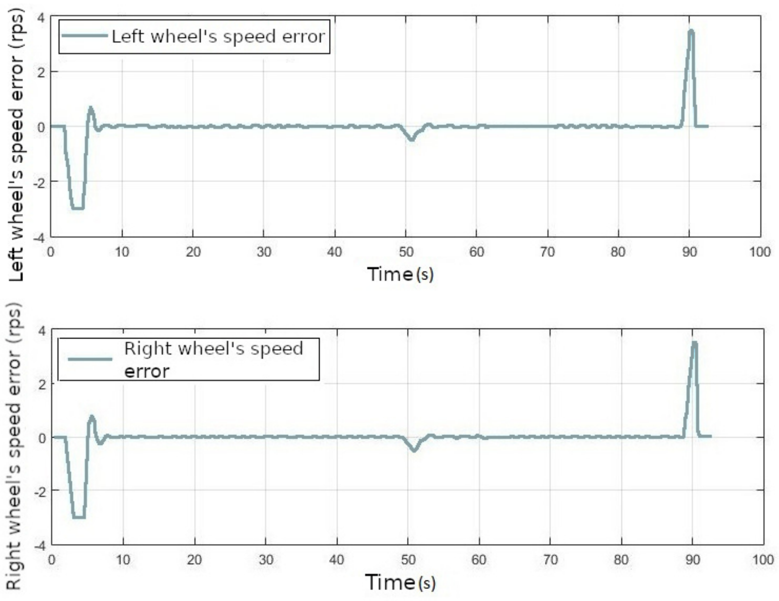

Fig. 19 Errors between the right and left wheel velocities and their respective references.

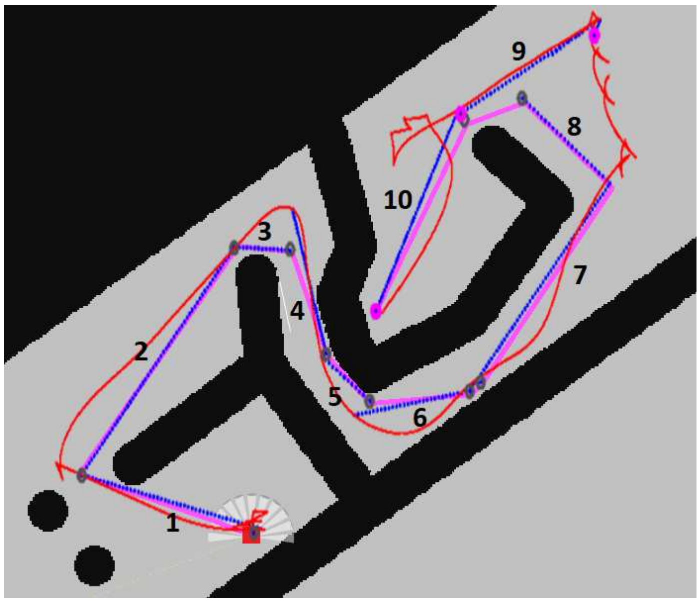

Fig. 20 Long Approach Test Results.

6.5 Simulation test results

The proposed framework's greatest advantage is its flexibility in handling the architecture. To test this, we changed the executive level and evaluated the functional and organizational levels' behavior.

To this end, we simulated the RoboMagellan environment as closely as possible, as shown in Fig. 21. From the georeferenced map of the competition's ground (top left) we built the robot's cost map (top right) and the simulation environment in Gazebo (bottom). We used the Pioneer 3-AT in the simulation, whose structure is very similar to Bruce's. Since they are both nonholonomic differential robots.

The test consisted of finding and reaching three different target cones. The routes planned for each target are shown in Fig. 22, with the first cone at the top left, the second at top right, and the third in the bottom part of the figure.

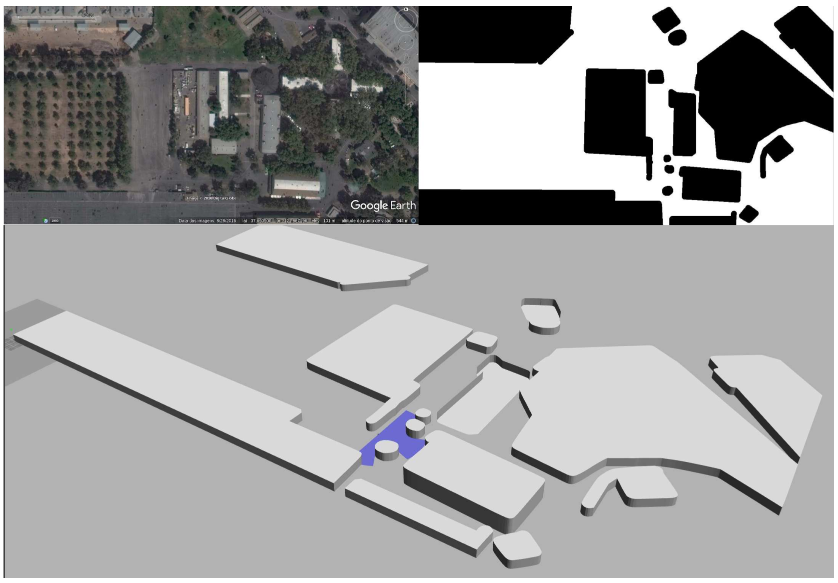

Fig. 21 The map utilized in the simulation.

The robot successfully reached the first two target cones, but failed to complete the challenge due to a collision with a wall. In the simulated tests, the obstacle avoidance modules were turned off and the robot used only its odometry to navigate. Since a simple reactive behavior could have prevented this failure, this test demonstrates a flaw of a purely deliberative system.

Fig. 23 presents the path calculated by the route planner (in red) and the actual path executed (in blue). The trajectory control module had to make adjustments in the first segment because the path calculated did not consider the robot as nonholonomic. Once the robot corrected its orientation, at the end of the first segment in the reference, the rest of the plan is followed perfectly.

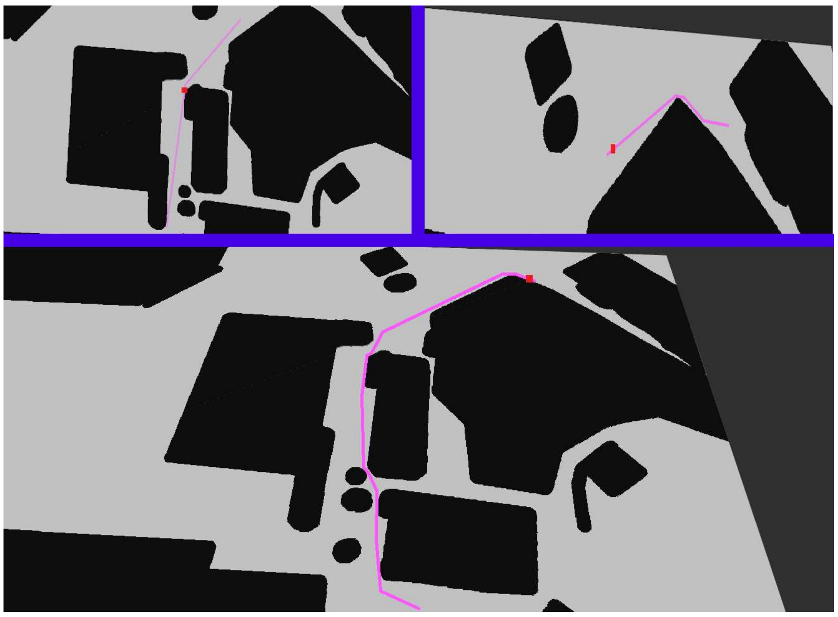

Fig. 22 The planned paths between the 3 traffic cones.

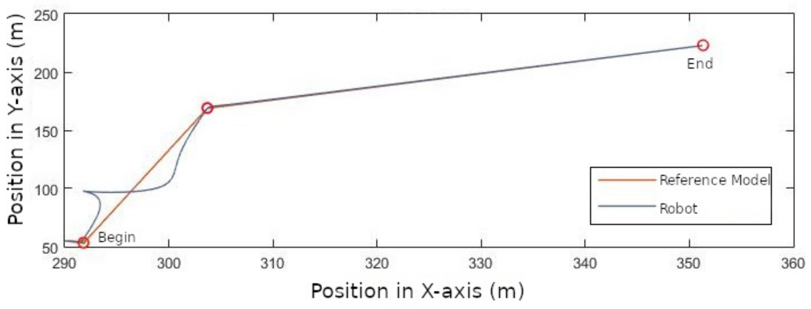

Fig. 23 Robot following the reference model.

7 CONCLUSIONS

Mobile robot competitions propose new challenges every year and these contribute with the advances in the field of robotics. To tackle the RoboMagellan competition, in which the participants must locate and reach specific targets, we built a robot and proposed a framework for quickly developing system architectures, upon which we created a system capable of autonomously navigate outdoor environments.

The framework was tested in real and simulated environments, showing that the proposed system is able to successfully handle the changes in the environment conditions and elements. The modular approach taken enabled different configurations of software and hardware components to be quickly tested, and provides flexibility for future developments.

Several improvements are currently under investigation. The route planner module will incorporate the robot's orientation and the knowledge that it is nonholonomic. New modules in the executive level, to handle different sensors and actuators available, will be tested and different algorithms for the functional levels are being investigated. Finally, we plan to test the framework in other robots to further test its flexibility.

References

1. Alami, R., Chatila, R., Fleury, S., Ghallab, M., Ingrand, F.: An architecture for autonomy. The International Journal of Robotics Research **17**(4), 315–337 (1998)
2. Allmendinger, F., Eddine, S.C., Corves, B.: Coordinate-invariant rigid-body interpolation on a parametric c1 dual quaternion curve. Mechanism and Machine Theory **121**, 731 – 744 (2018). DOI https://doi.org/10.1016/j.mechmachtheory.2017.11.023
3. Atsuzawa, K., Nilwong, S., Hossain, D., Kaneko, S., Capi, G.: Robot navigation in outdoor environments using odometry and convolutional neural network. In: IEEJ International Workshop on Sensing, Actuation, Motion Control, and Optimization (SAMCON) (2019)
4. Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D., et al.: Odin: Team victortango's entry in the DARPA urban challenge. Journal of field Robotics **25**(8), 467–492 (2008)
5. Baklouti, E., Amor, N.B., Jallouli, M.: Reactive control architecture for mobile robot autonomous navigation. Robotics and Autonomous Systems **89**, 9–14 (2017)
6. Bar-Shalom, Y., Campo, L.: The effect of the common process noise on the two-sensor fused-track covariance. IEEE Transactions on Aerospace and Electronic Systems **AES-22**(6), 803–805 (1986). DOI 10.1109/TAES.1986.310815
7. Barshan, B., Durrant-Whyte, H.F.: Inertial navigation systems for mobile robots. IEEE Transactions on Robotics and Automation **11**(3), 328–342 (1995)
8. Bauchspiess, R.: Sistema de visão para rastreamento de objetos alvo para robô móvel. Undergraduate thesis, Universidade de Brasília, Brasilia, Brazil (2017). URL `http://bdm.unb.br/handle/10483/36`
9. Black, H.D.: A passive system for determining the attitude of a satellite. AIAA Journal **2**, 1350–1351 (1964)
10. Borges, G.A.: Um sistema Óptico de reconhecimento de trajetórias para veículos automáticos. Master's thesis, Universidade Federal da Paraíba (1998)
11. Borges, G.A., Lima, A.M., Deep, G.S.: Design of an output feedback trajectory controller for an automated guided vehicle. In: XIII Congresso Brasileiro de Automática (2000)
12. Breuer, T., Macedo, G.R.G., Hartanto, R., Hochgeschwender, N., Holz, D., Hegger, F., Jin, Z., Müller, C., Paulus, J., Reckhaus, M., et al.: Johnny: An autonomous service robot for domestic environments. Journal of intelligent & robotic systems **66**(1-2), 245–272 (2012)
13. de Brito, C.G.: Desenvolvimento de um sistema de localização para robôs móveis baseado em filtragem bayesiana não-linear. Undergraduate thesis, Universidade de Brasília, Brasilia, Brazil (2017). URL `http://bdm.unb.br/handle/10483/19285`
14. Bó, A.P.L.: Desenvolvimento de um sistema de localizacão 3d para aplicacão em robôs aéreos. Master's thesis, Universidade de Brasilia, Brasilia, Brazil (2007)
15. Cacitti, A., Zapata, R.: Reactive behaviours of mobile manipulators based on the dvz approach. In: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, vol. 1, pp. 680–685. IEEE (2001)
16. Chavez, J.R.M.: Zona virtual deformável com filtro de partículas no rastreamento de obstáculos em robótica móvel. Mestrado em sistemas mecatrônicos, Universidade de Brasília (2014)
17. Chenavier, F., Crowley, J.L.: Position estimation for a mobile robot using vision and odometry. In: Proceedings 1992 IEEE International Conference on Robotics and Automation, pp. 2588–2593 vol.3 (1992)
18. Crassidis, J.L.: Sigma-point Kalman filtering for integrated GPS and inertial navigation. IEEE Transactions on Aerospace and Electronic Systems **42**(2), 750–756 (2006)
19. Cristóforis, P.D., Nitsche, M., Krajník, T., Pire, T., Mejail, M.: Hybrid vision-based navigation for mobile robots in mixed indoor/outdoor environments. Pattern Recognition Letters **53**, 118 – 128 (2015). DOI https://doi.org/10.1016/j.patrec.2014.10.010. URL `http://www.sciencedirect.com/science/article/pii/S0167865514003274`
20. Faisal, M., Hedjar, R., Sulaiman, M.A., Al-Mutib, K.: Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment. International Journal of Advanced Robotic Systems **10**(1), 37 (2013). URL `https://doi.org/10.5772/54427`

21. Fiack, L., Cuperlier, N., Miramond, B.: Embedded and real-time architecture for bio-inspired vision-based robot navigation. Journal of Real-Time Image Processing **10**(4), 699–722 (2015)
22. Fukao, T., Nakagawa, H., Adachi, N.: Adaptive tracking control of a nonholonomic mobile robot. IEEE Transactions on Robotics and Automation **16**(5) (2000)
23. Fukao, T., Nakagawa, H., Adachi, N.: Adaptive tracking control of a nonholonomic mobile robot. IEEE Transactions on Robotics and Automation **16**(5), 609–615 (2000)
24. Gustafsson, F., Hendeby, G.: Some Relations Between Extended and Unscented Kalman Filters. IEEE Transactions on Signal Processing **60**(2), 545–555 (2012). DOI 10.1109/TSP.2011.2172431
25. Haralick, R.M., Shapiro, L.G.: Image segmentation techniches. Computer vision, graphics, and image processing (1985)
26. Huan-cheng, Z., Miao-liang, Z.: Self-organized architecture for outdoor mobile robot navigation. Journal of Zhejiang University-SCIENCE A **6**(6), 583–590 (2005)
27. Hunt, R.W.G., Pointer, M.R.: Measuring Colour, fourth edn. John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England (2011)
28. Iagnemma, K., Buehler, M.: Editorial for Journal of Field Robotics—special issue on the DARPA grand challenge. Journal of Field Robotics **23**(9), 655–656 (2006)
29. Jwo, D.J., Weng, T.P.: An adaptive sensor fusion method with applications in integrated navigation. Journal of Navigation **61**(4), 705–721 (2008). DOI 10.1017/S0373463308004827
30. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The international journal of robotics research **30**(7), 846–894 (2011)
31. Lavalle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Tech. rep., Iowa State University (1998)
32. Leonard, J.J., Durrant-Whyte, H.F.: Mobile robot localization by tracking geometric beacons. IEEE Transactions on Robotics and Automation **7**(3), 376–382 (1991)
33. Luo, R.: Multi sensor integration and fusion in intelligent systems. IEEE Transactions on Systems, Man and Cybernetics **19**(5), 901–931 (1989). DOI 10.1109/21.44007
34. Menegaz, H.M.T., Ishihara, J.Y., Borges, G.A., Vargas, A.N.: A Systematization of the Unscented Kalman Filter Theory. IEEE Transactions on Automatic Control **60**(10), 2583–2598 (2015)
35. Meng, W., Liu, E., Han, S.: A novel collaborative navigation architecture based on decentralized and distributed ad-hoc networks. In: 2012 IEEE International Conference on Communications (ICC), pp. 606–610. IEEE (2012)
36. Mutambara, A.G.: Decentralized estimation and control for multisensor systems. Routledge (1998)
37. de Oliveira, R.W.S.M.: Uma arquitetura de navegação para robôs móveis. Undergraduate thesis, Universidade de Brasília, Brasilia, Brazil (2017). URL `http://bdm.unb.br/handle/10483/19224`
38. Porto, L.H.S.: Controle de movimento de um robô não-holonômico com tração diferencial. Undergraduate thesis, Universidade de Brasília, Brasilia, Brazil (2017). URL `http://bdm.unb.br/handle/10483/19239`
39. Schiffer, S., Ferrein, A., Lakemeyer, G.: Caesar: an intelligent domestic service robot. Intelligent Service Robotics **5**(4), 259–273 (2012)
40. Silva Porto, L.H., Werberich da Silva Moreira de Oliveira, R., Bauchspiess, R., Brito, C., da Cruz Figueredo, L.F., Araujo Borges, G., Novaes Ramos, G.: An autonomous mobile robot architecture for outdoor competitions. In: 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), pp. 141–146 (2018). DOI 10.1109/LARS/SBR/WRE.2018.00034
41. Stentz, A.: Optimal and efficient path planning for partially known environments. In: Intelligent Unmanned Ground Vehicles, pp. 203–220. Springer (1997)
42. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., et al.: Stanley: The robot that won the DARPA Grand Challenge. Journal of field Robotics **23**(9), 661–692 (2006)
43. Valls, M.d.l.I., Hendrikx, H.F.C., Reijgwart, V., Meier, F.V., Sa, I., Dubé, R., Gawel, A.R., Bürki, M., Siegwart, R.: Design of an autonomous racecar: Perception, state estimation and system integration. arXiv preprint arXiv:1804.03252 (2018)
44. Valls, M.I., Hendrikx, H.F.C., Reijgwart, V.J.F., Meier, F.V., Sa, I., Dubé, R., Gawel, A., Bürki, M., Siegwart, R.: Design of an autonomous racecar: Perception, state estimation and system integration. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 2048–2055 (2018). DOI 10.1109/ICRA.2018.8462829
45. Ziegler, J., Werling, M., Schroder, J.: Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In: 2008 IEEE Intelligent Vehicles Symposium, pp. 787–791 (2008). DOI 10.1109/IVS.2008.4621302