Francisco Antero Cardoso Marques

Licenciado em Ciências de

Engenharia Electrotécnica e de Computadores

A Flexible Navigation System for Autonomous Robots Operating in Heterogeneous Environments

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: José António Barata de Oliveira,

Professor Auxiliar, FCT-UNL

Co-orientador: Pedro Figueiredo Santana,

Professor Auxiliar, ISCTE-IUL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2012

Copyright

A Flexible Navigation System for Autonomous Robots Operating in Heterogeneous Environments

Acknowledgements

I would like to express my gratitude to my dissertation supervisor Prof. José Barata for his invaluable support and for giving me the outstanding opportunity to work on the robotics domain. Also like to thank co-supervisor Prof. Pedro Santana for the motivation, knowledge and support and also for opening me the world of robotics. Also the fun and extremely noisy research team members André Lourenço, Carlos Sousa, Eduardo Pinto, Gonçalo Cândido, Giovanni di Orio, Pedro Gomes and Ricardo Mendonça.

IntRoSys, S.A for giving me the opportunity for working in the Introbot project, especially Magno Guedes and Pedro Deusdado from helping me on the innumerous times I needed to test the robot.

André Lourenço, Pedro Pinto and Rui Borrego for the countless hours passed studying in the last couple of years.

My family for giving me support, my grandparents who helped me realize that hard work is more important than talent. My uncles who introduced me to most of the music I heard during the writing of this dissertation. My sister Teresa and mother Né that helped me throughout all these years. Finally my girlfriend Catia for the patience and all these years spent together.

Abstract

This dissertation presents a flexible navigation system for autonomous robot operating in heterogeneous environments. In the proposed system, flexibility occurs at several levels of the navigation system. At the lowest level, proper locomotion modes are selected according to the local context, which includes handling dynamic footprints when computing traversability costs. This flexibility ensures that the kinematic and morphological constraints of the robot are adequately considered when navigating in demanding environments. At the highest level, proper motion planning strategies are selected according to the global context, namely, the expected topology of the environment. This flexibility allows the robot to trade-off, in a context-aware way, accuracy of the planned motions and computational cost. As a result, the complexity of the planner matches the complexity of the environment, which is key to enable a proper management of computational and energetic resources. Following a pragmatic strategy, the robot obtains its global context from off-line generated maps, which are becoming widely available. To validate this idea, a tool for semantic labelling of satellite imagery was developed. Online, the robot obtains its global context extracting the semantic label (e.g., urban environment) and distribution of expected obstacles associated to its current global position. The proposed system leverages on the well-known Robotics Operating System (ROS) framework for the implementation of the navigation system underpinnings. For extensive validation purposes, a physics-based 3D simulator was used. Moreover, the system was validated over 1 Km long experiments on a physical four-wheeled robot. The results obtained show the ability of the system to ensure safe navigation in heterogeneous environments. This is a result of the ability of the system to exploit context awareness to reconfigure itself when facing transitions among topologically different environment regions. The added level of robustness introduced with the proposed system is expected to foster the application of autonomous robots in socially relevant domains.

keywords: planning, navigation, semantic, flexible.

Resumo

Esta dissertação apresenta um sistema flexível de navegação destinado ao controlo de robots autónomos em ambientes heterogéneos. No sistema proposto, a flexibilidade abrange diversos níveis. Num primeiro nível, o modo de locomoção correcto é escolhido de acordo com o contexto local, compreendendo o cálculo do custo de cada trajetória onde se inclui o seu footprint dinâmico. Num nível mais elevado, as estratégias de planeamento são seleccionadas de acordo com o contexto global, nomeadamente a topologia esperada do ambiente. Assim, seguindo uma estratégia pragmática, o robot extrai o contexto global de mapas gerados offline e, posteriormente, extrai a etiqueta semântica e a distribuição dos obstáculos associados à sua posição global actual. A validação do sistema foi feita, num primeiro momento, através de um simulador 3D de ambientes físicos, e depois, através de diversas experiências com um robot físico onde foi percorrido aproximadamente 1 km de distância. Os resultados obtidos mostram que o sistema permite uma navegação autónoma segura em ambientes heterogéneos, o que se deve sobretudo à sua competência para se reconfigurar perante transições entre regiões topologicamente diferentes. É esperado que este nível adicional de robustez possibilite a aplicação de robots autónomos em domínios socialmente relevantes.

palavras-chave: planeamento, navegação, semântica, flexibilidade.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The motivation behind this dissertation is the development of a flexible navigation system for autonomous robots operating in heterogeneous environments.

Using this type of control system, field robots may perform diverse tasks such has surveillance, search and rescue, environmental monitoring, and forest preservation.

Autonomous robot navigation was always widely studied, (Brooks, 1986; Arkin, 1989; Elfes, 1989; Kuipers and Levitt, 1988), and has proven to be a complex and interesting subject. Complexity increases if robots are to navigate in unstructured and heterogeneous environments. Even though some field robots already have the capability to navigate in complex environments, may they be off-road (e.g., (Thrun et al., 2006; Crane et al., 2007; Miller et al., 2006; Behringer et al., 2005; Braid et al., 2006)) or urban (e.g., (Urmson et al., 2008; Montemerlo et al., 2008; Miller et al., 2008; Chen et al., 2008)), the ability to transition between them remains an open problem.

Furthermore, autonomous robots' navigation systems developed to date cannot operate with the same level of performance in heterogeneous surroundings, as if they were built specifically to operate in one of them. One solution to the transition problem may be the use of topological maps (e.g., (Thrun, 1998)) to limit immense complexity of using more accurate metric maps when transitioning areas. Another solution is to use multiple-hierarchical semantic information (e.g., (Galindo et al., 2005)) to improve the system's adeptness to the surrounding environment. Other is to use scene classification to identify the type of surroundings and adjust navigation accordingly (e.g., (Collier and Ramirez-Serrano, 2009)). Finally, the extraction of semantic from satellite imagery can be used produce cost maps and traversable calculation (e.g., (Silver et al., 2006)), thus, adapting to the terrain.

This dissertation approaches the problem differently by proposing a flexible control system capable of exploiting local environment semantic information to adjust the navigation system to the surrounding's context. This way the robot is able to navigate in urban environments, off-road environments, and indoors without human intervention. Moreover, the smooth transition between these environments is also ensured.

In this dissertation, the extraction of context information is performed at two levels. At the first

level, context is retrieved from offline labelled overhead satellite imagery. At the second level, context is extracted from the robot's local perception sources.

The proposed system has flexibility at different levels to ease the adaptation and transition between diverse environments. This new level of robustness provides the robot with the tools to adapt and chose proper strategies according to the context.

To validate the proposed system, field trials were conducted on New University of Lisbon campus, where a robot with four independently steered wheels navigated autonomously for approximately 1 km at 0.5 meters per second over an operating period of 30 minutes.

During these trials the robot never collided even when facing different dynamic objects. Moreover, it was capable of reaching all of the navigational goals while transiting between heterogeneous environments.

Finally, regarding the main objective it was able to transition between open space and narrow space environments.

In sum, the results provided enough experimental data to show that the system is: (1) capable of providing autonomous robots with safe navigation behaviour; (2) capable of robustly handling GPS signal dropouts; (3) able to deal with noisy perceptual data causing obstacles false alarms; and (4) able of successfully exploit semantic information for smooth transitions between environments.

1.1 Dissertation Outline

This dissertation is composed as follows:

Chapter 2 reviews the state-of-the-art for mobile robot navigation.

Chapter 3 displays the Introbot robot platform, software architecture and simulator setup.

Chapter 4 describes the proposed navigation system and its components.

Chapter 5 presents the experimental setup and the set of results obtained, their strength and weakness;

Chapter 6 aggregates a set of conclusions, main contributions of this dissertation, and further research opportunities on the subject.

Chapter 2

Related Work

The main objective of this chapter is to introduce the reader to the basic components, the whole multitude of methods and best practices that can be applied when building a mobile robot's navigation control system.

Initially, robot motion and path planning methods are presented in Section 2.1, which is followed by an explanation concerning robot localization and mapping techniques (see Section 2.2), and finally the usefulness of semantic information (see Section 2.3) as more complex and robust behaviours are required.

It will be made clear how some methods are better suited for structured environments such as indoor navigation or urban facilities while others perform better navigating in unstructured and dynamic environments such as all-terrain or troublesome scenarios.

2.1 Robot Motion and Path Planning

One of the fundamental functions of a mobile robot's navigation system is to safely avoid obstacles, and that is something that this first section aims to explain.

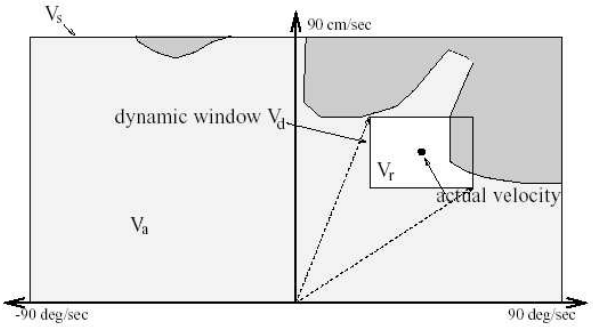
In subsection 2.1.1 obstacle avoidance and reactive techniques are addressed with special regard to their benefits and setbacks. Subsequently a description of global planning methods is exposed (see subsection 2.1.2), to bridge the most basic type of navigation constraints (e.g., obstacle avoidance) to the broader sense of the navigation's ultimate goal.

2.1.1 Local Planning

Local Planning holds the specific ability of controlling a mobile platform in order to achieve a certain target position. Specifically, the creation of trajectories representing feasible motions, ensuring

the robot's secure transition between its current position to a goal point in space. The goal can be either just a midway point of the whole path or simply a specific heading or even a final pose (position and heading). In the case of path following the local motion control increased granularity provides the means to correct global path errors, which would be unperceivable from the global planner coarser perspective. For the robot's local planner to behave in a reasonable fashion, the trajectory generation procedure needs to take into account the dynamics of the robot. The solution to this problem is not a trivial one, in fact, its complexity hinders a system from performing adequately in a whole variety of situations. This subsection presents some of the most relevant robot motion methods, namely the Dynamic Window Approach (Fox et al., 1997) and The Trajectory Roll Out (Gerkey and Konolige, 2008).

Dynamic Window Approach



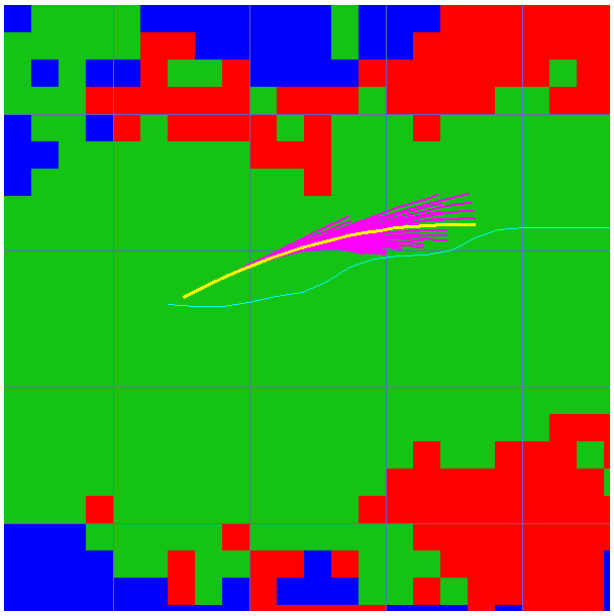
(a)

Figure 2.1: The dynamic window approach (a) The dynamic window approach where the black rectangle represents the window of possible velocity tuples (Fox et al., 1997)

In the Dynamic Window Approach (Fox et al., 1997) the main objective is to endow the mobile platform with appropriate reactive avoidance to handle the perilous situations the robot could find itself. The time factor is of crucial importance since it has to conciliate the dynamics of both the robot and the environment. This is attainable by taking into consideration the robot's motion dynamics to effectively reduce the search space of motion control possibilities, effectively lightening the computational load.

The search space reduction is carried out in three different steps: first the combination of linear and angular velocities are considered; secondly trajectories that make it impossible for the robot to stop safely when faced with an obstacle are discarded; finally a dynamic window is created from the robot's acceleration limits in the corresponding sample period, effectively confining the realm of analysed velocity tuples. Then an optimization function that combines linearly speed and distance to obstacles along the trajectory, is maximized to select the best trajectory. The major drawback of the Dynamic Window Approach is the complexity of the parameter tuning that when ill-configured can lead to erroneous behaviours.

Trajectory Roll Out



(a)

Figure 2.2: Trajectory roll out approach (a) The controller generates trajectories by sampling feasible velocities and simulating their application over a short time horizon. Generated trajectories are purple, the chosen trajectory is yellow, the desired global path is cyan, and obstacles are red (Gerkey and Konolige, 2008)

The Trajectory Roll Out (Gerkey and Konolige, 2008) takes the opposite direction that of the Dynamic Window Approach. In this method the search is limited to the space of feasible controls in contrast to feasible trajectories, confining the search to seek a good instead of the optimal control. The control space is normally defined as two-dimensional pairs of linear and angular velocities (e.g., surface robot) in which the sampled region is narrowed by the above-mentioned search policy. The simulation then predicts with a discrete time approximation of the robot's dynamics, this returns a path composed of 5 dimensional states. The trajectories simulated are then evaluated by a weighted cost (see Equation 2.1).

$$C(t) = \alpha Obs(t) + \beta Gdist(t) + \gamma Pdist(t) + \delta \frac{1}{x^2}(t) \tag{2.1}$$

In which there are three components, Obs that is the sum of the grid cell costs of the trajectory; Gdist is the estimated shortest distance to the goal; Pdist that is the estimated shortest distance to the path; X that is the translational part of the velocity command; $\alpha\ \beta\ \gamma$ are parameters to control the weight of each component.

By tuning these parameters it is possible to control the preferred trajectories of the robot. They could remain far from obstacles, be more goal driven, choose to remain the closest possible to the global planner path or prefer to move at higher speeds. Taking in account the normal perception limitations of mobile robots, where the majority of the perception sensors are situated at the front, a supervisory node is used to create a trajectory creation hierarchy. The forward motion velocities are

tried first then selecting the best of the legal trajectories, next if none of the forward velocities is valid, the in place rotations are sampled and if those are not valid also, the robot finally tries to backup. This supervisory node is used to promote progress if that is deem possible. As in the Dynamic Window Approach the acceleration limits may difficult the parameterization.

2.1.2 Global Planning Methods

The Local Navigation based architectures are adequate when dealing only with basic navigational needs, nevertheless they lack a broader sense of awareness. Hence, cannot assure an optimal path when traveling from point A to point B. Additionally this limitation can also take the robot into unforeseeable and troublesome situations, being the *cul-de-sac* one of the classical examples. Indeed this known limitation in conjunction with the greater availability of computational power fostered the adoption of global path planners. Global planning is indeed a fundamental part of most mobile robot navigation systems. Some of the most relevant path planners are overviewed in the following sections.

Dijkstra

Edsger Dijkstra developed the Dijkstra's algorithm (Dijkstra, 1959) in the mid-1950, a graph search based algorithm capable of finding the shortest path between two vertices (Graph Geodesic). The objective is when supplied with a certain start and goal vertices to find the path with lowest cost connecting them.

To find the optimal path all the vertices are initially marked as unvisited by placing their distance to the goal as infinity. Subsequently, the initial node is set as the current position and tentative distances to neighbouring nodes (part of the unvisited set) are calculated saving the shortest distance to each of them. Then, after considering all neighbours in the vicinity the current node is set as visited and removed from the unvisited set. Afterwards, a verification occurs to find if the goal is marked as visited or if the smallest distance in all of the unvisited list comprising nodes is infinity, if any of the above conditions is valid the search comes to a halt. However, if this is not the case, the unvisited node with the shortest tentative distance is marked current and the algorithm goes on.

Even though it is capable of finding the shortest path between two points, the Dijkstra's algorithm search is unfocused, with no hierarchy regarding its expansion. For this reason its inefficient compared with the subsequent algorithms.

A*

In 1968 Peter Hart, Nils Nilsson and Bertram Raphael extended the Dijsktra algorithm (Hart et al., 1968) by using heuristics to focus the search. Thus, addressing the algorithm's greatest pitfall. This is possible by using function ($f()$ that sums the goal distance from the starting node to the current

node ($g()$) to an heuristic ($h()$) that estimates the distance from the current node to the goal (equation 2.2).

$$f(x) = g(x) + h(x) \tag{2.2}$$

Normally an optimistic linear distance is used for the heuristic function so that its admissible, that means that it does not overestimate the distance to the goal. Starting in the initial node a *open set* is created organized in a priority queue, sorted by the lowest cost of the distance plus cost function. At each iteration the node with lowest cost is removed from the queue and the other nodes are updated and re-queued. This recursive behaviour is repeated until the goal node was the lower cost value of all the priority queue. Then, goal cost is only the path length, because at the goal the distance to goal is zero.

By focusing the search using heuristics the A * algorithm effectively diminishes the domain of search enabling it to cope with large search spaces, and so outperforming Dijsktra algorithm in that type of situations.

ARA*

The Anytime Repairing A* (Likhachev et al., 2003) algorithm is a response to real world problems than cannot be solved by traditional optimal planners. In conclusion anytime planners are the answer to situations where the response of the planner is more important than the optimality of the path. In some situations there is not an unlimited time frame to create an optimal path, so anytime planners arose to solve this issue. Providing a sub optimal answer in a short time and then continuously improving the solution is one way to solve the time frame demands.

One hypothesis to create a Anytime planner is to use A* search (Anytime A*) with inflated heuristics. To inflate the heuristics a parameter ϵ is multiplied with the normal heuristic function (always $\epsilon > 1$). This provides a bound on the desired sub-optimal path equal to the ϵ value. By continuously decreasing ϵ , a range of sub-optimal solutions is created, where the length of the solution is no larger than ϵ times the optimal path length.

Nonetheless, this approach searches start always from scratch, thus not capitalizing on previously gathered information, effectively wasting computational power. So in ARA*, A* is started with a large ϵ and run several times always lowering the value until an optimal path is returned, but at the same time each iteration uses previously search information making the algorithm faster than normal Anytime A*.

D* Lite

The possibility of profiting from data already gathered from previous searches was already mentioned in (Ramalingam and Reps, 1996). Where solutions to similar tasks where found faster that if

the search were always restarted. The D * lite algorithm (Koenig and Likhachev, 2002) also aims to reuse information gathered in previous searches and focusing the search by using heuristics methods as in A* (Hart et al., 1968).

The planner is designed to determine the shortest path between the vertex where the robot is located and the goal vertex by monitoring the vertex cost change whilst the robot is moving. Search direction is from goal to the robot position and the shortest path is travelled by minimizing the cost function and braking ties arbitrarily.

The reuse of previous search information is possible with the introduction of Right Hand Side rhs computation, that represents the value of a vertex parent plus the cost to travel to that vertex. The planner detects local inconsistencies by comparing this computation to the cost of that vertex, enabling it to update graph information while moving. In a normal situation the rhs and the node cost are equal and so locally consistent. Therefore, when the node cost is different from the rhs computation the cost changed dynamically. Inconsistency falls in two different types: under consistency where the node cost is lower than the rhs computation meaning that the vertex is now more expensive and the paths that pass through this node need to be recalculated; over consistency is when the node cost is greater than the rhs meaning that the path to this vertex is now less expensive and this information is used to manage the search open list.

When there are no inconsistencies the algorithm is the same as the A*. Nevertheless, when there are local inconsistencies those vertex are added to the open list. Subsequently, the algorithm is always checking for graph changes to add them to the open list refreshing key values when there is updated connection data and not only with new connections. This happens when the robot advances, heuristics decrease because the movement is towards the goal and so connections change and thus key values are recalculated.

2.2 Robot Localization and Mapping

In this section a brief overview of localization and mapping applied to mobile robotics is presented. Initially several localization methods are addressed followed by mapping techniques. Localization is an essential issue when working with mobile robots, the capability of estimating the robot's position in an environment is paramount for an effective autonomous navigation. In other words the ability of the robot to know its position in a reliable way is essential for a large array of mobile robotics applications.

There are two categories of localization the major difference being the demand of prior knowledge of the initial robot's position, while tracking techniques need this type of information global techniques do not require data concerning the initial localization. As a result the first approach is useless in case of a wrong initial estimate or if the robot becomes delocalized outside certain bounds. Whereas global techniques are capable of coping with no initial information about the robot's position. This means that they are able to handle the *kidnapped robot problem* where a robot is carried to an arbitrary location. The major trend in robot localization is the use of probabilistic methods, so three different methods are presented below, first Kalman Filtering (Kalman et al., 1960) secondly Markov Localization (Fox et al., 1998) and to conclude Monte Carlo Localization (Thrun et al., 2000). After

a brief overview of mapping techniques is provided in subsection 2.2.4, followed by the conjunction of localization and mapping a problem known as SLAM (Simultaneous Localization and Mapping) is explained in subsection 2.2.5.

2.2.1 Kalman Filtering

Kalman filtering (KF) is a case of probabilistic state estimation used in mobile robotics (Kalman et al., 1960). The Kalman filter contains two different models, the plant model and measurement model. In mobile robotics the plant model represents the relationship between motion controls and the robot position, where the measurement model is the data provided by the robot sensor package related to its current position. The KF uses a two-step predictor algorithm in a form of force feedback, first it projects the predicted state estimate and the error covariance from the earlier state obtaining the *a priori* estimates for the next state. The feedback occurs when a sensor measurement is received, it corrects the predicted state estimate by using the measurement to generate *a posteriori* state estimate. This cycle is repeated after each time step and measurement update.

The applicability of the traditional Kalman Filter to localization in mobile robotics is impaired by the its incapability of handling nonlinear systems, this is because mobile robots tend to be highly nonlinear. For that reason in mobile robotics variations of KF that can cope with nonlinear systems are normally used, like Extended Kalman Filter (EKF) (Jazwinski, 1970) or Unscented Kalman Filter (UKF) (Julier and Uhlmann, 2004). To handle with nonlinear systems the EKF linearizes all nonlinear transformations and replaces the Jacobian matrices for the linear transformations in the traditional Kalman Filter. This of course is only robust if the error propagation can be approximated by a linear function. However, the major drawback of using Kalman Filtering in mobile robotics is due to its unimodal nature so it cannot localize the robot from scratch. Furthermore, it can quickly diverge if not properly initialized, in fact its incapable of recovering from this type of failures.

That said Kalman Filters are successfully used in mobile robotic applications (Leonard and Durrant-Whyte, 1991, 1992; Schiele and Crowley, 1994).

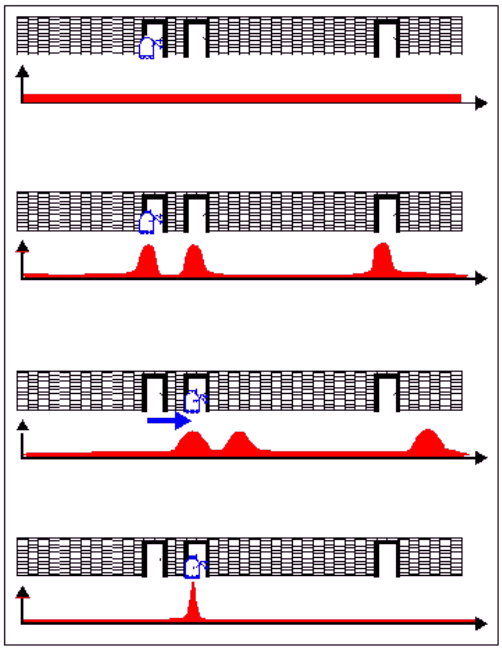
2.2.2 Markov Localization

Another probabilistic method used for mobile robot localization is Markov Localization (Fox et al., 1998). By maintaining a probability distribution over the space comprised by all possible robot locations, it weighs all possibilities and makes an mathematical assumption regarding the robot's position.

The key idea of Markov localization is to maintain a probability density over the space of all locations of a robot in its environment. This is true for both grid based maps (Burgard et al., 1996, 1998) and topological maps (Simmons and Koenig, 1995; Nourbakhsh et al., 1995). If a robot is placed in a location without prior knowledge a uniform distribution of probability is laid in all the positions. Then, using sensory information it gathers from the surrounding environment (i.e. for example range data) it can raise the probability for locations with similar range information. However, to encompass sensor noise (i.e. the sensor information might by plagued by noise) it does not lower

the probability in all the remaining positions. The next step in the localization method is the handling of robot motion. After moving and by considering the noise induced by the uncertainty intrinsic to robot motion, the probability is smoothed lowering the certainty of the robot's current location. Then, robot queries it sensors and updates the perception information, using this new data the probability of certain locations is raised. Leading to a new and more accurate belief. All of this steps are evident in Figure 2.3.

In essence Markov localization is a multi-modal discrete state space localization technique, which provides a robot with the capability of robustly locate itself in static environments that is has prior knowledge about. However, it is also possible to apply Markov Localization to dynamic environments, as it was showed by Fox's work (Fox et al., 1999) were this was achieved by the introduction of filtering techniques .



(a)

Figure 2.3: Markov localization (a) The basic idea of markov localization: A mobile robot during localization (Fox et al., 1998) In the top image the robot's position probability density (in red) is spread equally in all the positions. After the sensors of the robot detect that it is near a door, so the locations near a door have their probability raised. The robot moves (blue arrow) and the certainty of its position lowers. The sensor is queued and the robot is localized with a high probability.

2.2.3 Monte Carlo Localization

Monte Carlo Localization is another method that uses probability density but differs from the aforementioned by using a sampling-based representation. By using this type of representation it can address some of the pitfalls of the other methods previously mentioned. It is able to represent multi-modal representations and so it is a global localization technique, able to localize the robot from

scratch. Nevertheless, Markov Localization is also global localization technique but at the expense of large amounts of memory and computational burden. Furthermore, Monte Carlo Localization is more accurate than Markov Localization if the grid cell size is fixed. To implement Monte Carlo Localization a particle filter is used, in the next paragraph the particle update process is described.

First the next generation of particles is created from the previous generation and by sampling the probabilistic motion odometry model as a proposal distribution. Then, the particles are weighted according to a importance sampling principle related to sensor measurements.

This importance weighting method is the likeability of each particle according with the sensor information available. In other words the difference between the predicted measurement and the actual measurement. The next step is resampling, where particles are retrieved randomly but proportional to their importance weight. Making the particles with better probability more likely to live on, where the lower probability particles tend to disappear. The random nature of the resampling is an important part of the particle filter algorithm, because if only the particles with higher probabilities lived on there is a risk of finding only a local maximum. After this resampling all the particles have the same weight and the cycle returns to the first step. These approaches combine the advantages from the previous localization methods, are capable of multi-modal and globally localize the robot.

2.2.4 Mapping

In this subsection an outline of some mapping techniques is provided, with emphasis on topological mapping and merging topological with metric maps. Typically, topological maps are represented as graph states, in which nodes represent landmarks, places or regions and paths represent the trajectories that will take an autonomous agent from one region to another. In Remolina and Kuipers work (Remolina and Kuipers, 2004) an overview on how to merge sensory input, topological and local metric information, into a topological map is presented. Also Thrun (Thrun, 1998) stresses the efficiency of using topological representations when solving large scale planning opposed with the immense complexity of using more accurate metric maps with grid-base methods. Nevertheless, he also states that generating accurate and consistent topological maps is a difficult task, and so presents the hypothesis of generating topological maps on top of grid-based metric maps. This perspective tries to combine the accuracy off metric maps and the efficiency of using topological maps to planning problems. A similar approach is also defended by Kuipers et al. (2004) the use local metrical information for the small-scale space and topological maps for the large-scale space. While Konolige et al. (2011) proposes a method to build topological map using a graph SLAM system, local navigation metric maps are also used. The major advantage of all the aforementioned methods is the faster global planning provided by the use of a topological graph in opposition of a full metric map, at the same time maintaining a safe navigation behaviour by using metric maps for local navigation.

2.2.5 Simultaneous Localization and Mapping

All of the localization methods described so far have one common assumption that there is *a priori* knowledge of the robot's surrounding environment. Although, this is an acceptable assumption

in some circumstances (Thrun et al., 2000). In other situations where the environment is unknown, for instance a search and rescue operation this is not possible. Creating a problem because the localization errors are now incorporated in to the creation of the map representation, limiting the operation of mobile robots in such situations.

The incorporation of localization errors in the creation of the map is a technique known as Simultaneous Localization and Mapping (SLAM) and will be described in this subsection.

The ability to build maps is one of the essential capabilities of a mobile robot when faced with uncertain environments. However, it is a complex problem, it requires a conjunction of a robust localization and consistent mapping. This two premises normally are mutual dependent, to generate a trustworthy map a reliable location is needed and vice-versa. In essence uncertainty created by the robot motion when building a map forces the robot to also perform localization. Several solutions are available they can be based in Kalman Filter approaches (EKF-SLAM) or on probabilistic methods (particle filters) (Montemerlo et al., 2002, 2003; Grisetti et al., 2007), there are also differences in map representation while some use grids others use topological maps. All approaches deal with the same basic problem, they need to estimate online both the trajectory of the robot and the location of landmarks without *a priori* information. In this subsection a brief and simplified explanation of how these approaches function will be presented.

The solution using EKF suffers from the same problems described in the aforementioned 2.2.1 Kalman Filtering localization subsection, but it also profits from the its strong points. In this type of implementation the robot motion depicted by an EKF and landmark observations are used for the measurement updates. The EKF-SLAM updates all the landmarks and covariance matrixes at each observation measurement, entailing a great computational burden. Nevertheless, some optimized variations showed that they can deal efficiently with a large number of landmarks. Another important issue in this type of implementation is the strong dependency of sound association between observations and landmarks, if this association becomes incorrect the EKF-SLAM algorithm fails.

To solve some of the limitations of the EKF solution, some implementations shifted to particle filters, namely RAO-Blackwellized particle filters (Montemerlo et al., 2002, 2003; Grisetti et al., 2007). The particle filter is used to estimate the posterior of the map using the robot's motion, the estimation provided by sensor observations and odometry measurements. In a Rao-Blackwellized filter the computation is done first by estimating the trajectory of the robot and then generating a map associated with that motion. Amongst the most studied models, the FastSLAM model (Montemerlo et al., 2002, 2003) represents a robot trajectory hypothesis with a particle and each map landmark is independent. This means that each map (particle) is composed from independent landmarks represented by a Gaussian distribution. Every observation of a landmark is treated as an EKF measurement update of that trajectory, which is represented in by single particle.

Both SLAM implementations mentioned earlier are based on landmarks, but there are also algorithms that are grid based. Grisetti et al. (2007) optimize SLAM reducing the number of particles needed, this implementation takes advantage of the accuracy of laser based range sensors. Instead of using only the odometry model for the proposal distribution, the most recent sensor information is used in the creation of the next generation of particles. The algorithm capitalizes on the fact that laser sensor information leads to extremely peaked likelihood distributions. To optimize SLAM initially the particles are created using an odometry model proposal distribution. Then, the search is focused in

a meaningful area by using a scanmatcher algorithm starting from an initial pose guess, sampling is then done with points centred around the region of interest return by the scanmatcher. This leads to a new particle pose, based on this improved proposal distribution, the importance weights are updated, and each map (particle) is refreshed according to the pose and observation, followed by resampling. This focusing of the search produced by the use of scan-matching effectively diminishes number of particles needed, and so reduces computational load, in addition resampling is also optimized to reduce the risk of particle depletion.

SLAM adds the capability of operating in unknown terrain. This of course is only interesting in certain situations and environments. Mainly because of the computational burden associated, in other words when *a priori* knowledge is easily obtainable the advantages of SLAM are overpowered by the need to save computational power. That said SLAM is one of the most important mobile robotics achievements.

2.3 Semantic Information

By extracting semantic from the environment the realm of tasks performed by mobile robots can grow in complexity and robustness. In this subsection two different uses of semantic data extraction are described. First, scene classification used to exchange navigation modes between indoor and outdoor scenarios. Second, semantic data is extracted from overhead imagery to improve global navigation.

Scene Classification

Collier and Ramirez-Serrano (2009) uses image classification to determine the operating mode of the robot. By extracting features from image perception information and training a classifier using supervised learning, the system is able to adapt to two different types of environments, indoors and outdoors. A autonomous robot normally is designed to operate in only one type of environment and could fail when faced with different surroundings. Scene classification adds the capability of categorizing the surrounding context, enabling the robot to use this information in the decision making process.

Initially images from sensor streams are analysed and certain features are extracted: color Histograms, RGB, HSV, LUV, color spaces; orientation histograms computed form horizontal and vertical Sobel Images; color moments from HSV and LUV; curvature histograms computed form vertical and horizontal Sobel images. These features are then classified by using either artificial neural networks or support vector machines, the resulting classification is then used for the informed exchange of the robot's mapping system. In outdoors the robot uses a 2.5D map by integrating range data from stereo pair and pose estimates from GPS and IMU into a grid-based map. In the indoor mode an EKF-SLAM algorithm is used based on a 2D map representation.

The integration of scene classification into the robot's navigation system provides the means to operate in two different environments with diversified challenges. In other words the addition of

semantic data improves the scope and performance of the navigation system.

Long Range Navigation Using Overhead Data

The work of Silver et al. (2006), uses the extraction of Semantic data of heterogeneous overhead information to allow an autonomous robot to navigate through large complex environments. The feature extraction depends on the nature of the data. The system can extract data from different sources and also different sets from the same type of data. This includes image base features like: HSV images; Near-Infrared imagery and Normalized Difference Vegetation Index (NDVI); elevation base features where local relative elevation is used instead of absolute elevation; and, finally point cloud based features are used to describe the density of vegetation or other occluded space over the ground surface. A neural network is used for the classification, trained by a data set with labelled regions selected by a human user. Next Digital Elevation maps and LiDAR are processed and the ground surface is extracted. By doing this the remainder will probably only contain vegetation, rocks and other objects that are above the ground plane. A vehicle model is then used for mobility analysis to assess the traversability, this is achieved by placing the robot model in different set of poses on the ground points. Detecting positive and negative obstacles as well as computing ground clearance, and so retrieving the traversability cost of each map cell. Finally, a costmap is produced where traversal costs are computed independently from: the terrain classification; vehicle mobility analysis; and, a speed limit for each cell created by the human operator. Based on this costmap a feasible and accurate global path can be generated, adding the capability to an autonomous robot to navigate throughout large scale complex natural environments.

Chapter 3

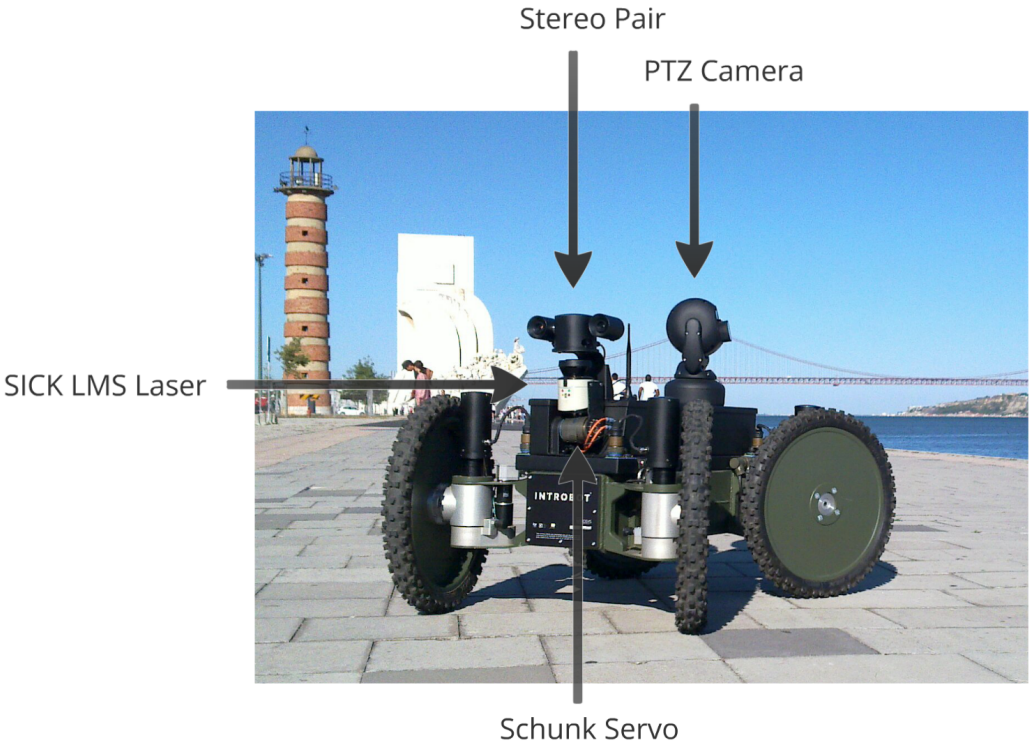
Real and Simulated Robot

In this chapter the autonomous vehicle used in the implementation of this thesis is presented. First, the real Introbot is presented in Section 3.1, where a brief description of the mechanical structure, hardware and perception is provided. Then, the contribution made by creating a simulation platform is presented in Section 3.2 based on the Gazebo simulator and is composed by a robot simulation model and a couple of simulation worlds. Finally, in Section 3.3 the Robot Operating System (ROS) based software architecture is presented, with a brief overview of the contributions.

3.1 The Introbot Robotic Platform

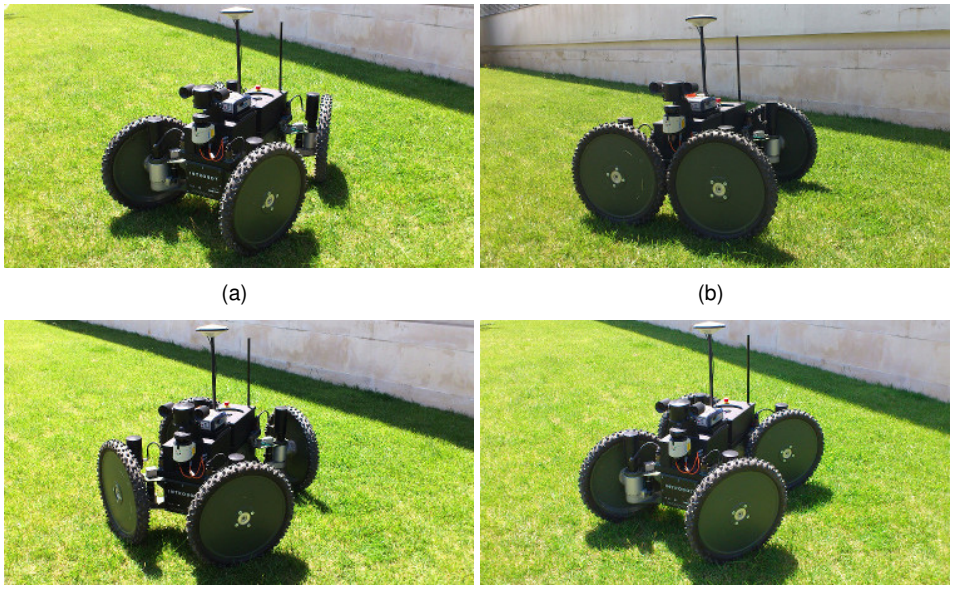
The Introbot is a versatile outdoor robotic platform with quasi-omnidirectional motion for real world applications such as surveillance, agriculture, environmental monitoring and other related domains. It was develop in a joint effort by SME Portuguese company IntRoSys, S.A.[1], LabMAg of the University of Lisbon[2] and Uninova CTS[3] in the context of the QREN project Introsys Robot[4].

[1] IntRoSys, S.A. `http://http://www.introsys.eu`
[2] LabMAg of the University of Lisbon `http://labmag.di.fc.ul.pt`
[3] Uninova - Centre of Technology and Systems `http://www.uninova.pt/cts/`
[4] Introsys Robot, project number 2008/002641 `http://www.introbot.pt`



(a)

Figure 3.1: Robot sensor package (a) Overview of perception sensor package



(a) (b)

(c) (d)

Figure 3.2: Introbot locomotion modes. (a) Double Ackerman. (b) Lateral. (c) Turning point. (d) Displacement.

The introbot was designed to operate in diverse kinds of terrains and situations allowing fully and semi-autonomous control intended to work on outdoor real world applications.

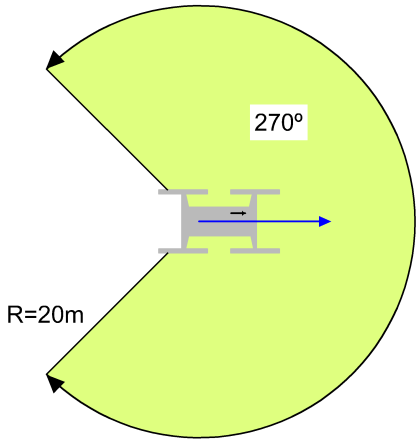
The operational versatility of the Introbot is possible because of a robust modular hardware architecture mainly composed by off-the-shelf components. The sensor package is composed by multiplicity of perceptional devices that include a 2-D laser scanner (see Figure 3.3), a ring of ultrasonic sensors, a stereoscopic vision sensor, and a set of monocular cameras (see Figure 3.1). Additionally, also an inertial measurement unit (IMU) with a built-in global positioning system (GPS) device is available to enlarge the realm of perception sources, in this case mostly suited for localization and navigation. Processing is assured by a 2.4 ghz 4 gigas ram Pentium(R) Core 2 duo CPU T4300 2.10GHz with 4 Gb of RAM, running a 32-bit Linux distribution Ubuntu 10.10 (Maverick Meerkat) industrial board for basic functions, and to perform more complicated tasks there also the possibility of adding additional processing.

To detect problems that could result in a lower operational readiness the robot possesses self-monitoring and diagnosis capabilities. This is accomplished by integrating several internal sensors responsible for gathering temperature readings, voltage and current drawn, and controlling the health of the battery.

Another important asset included in the Introbot design is the no-slip quasi-omnidirectional locomotion structure. Based on a 4x4 independent wheeled system, it can provide up to five types of movement (see Figure 3.2). This choice has multiple advantages, it reduces mechanical stress, energy consumption, and for localization and navigation purposes it avoids large odometry errors. This versatility is possible by coupling directly each motor to their corresponding wheel through a gear instead of chains or belts. Furthermore this reduces energy loss and more importantly the number of failure points. All of this in conjunction with the aggregated power of all motors, exceeding 1 kW maximum, allows the robot to navigate at up to 1.5ms^{-1} and overcome up to 45 degrees incline slopes.

Another issue often neglected in current robotic platforms is the ease of their transportation and storage. It has a medium sized footprint (800mm $\times$ 1500mm $\times$ 700mm) and weight, 100 Kg. Also, each wheel supports hand locking and manual clutch, and the longitudinal passive axis can also be locked in the central position. All of the above characteristics ease the manual transportation of the Introbot.

Hence all of these assets combined offer a highly capable robust mobile robot platform able of operating in all-terrain and all-weather situations.



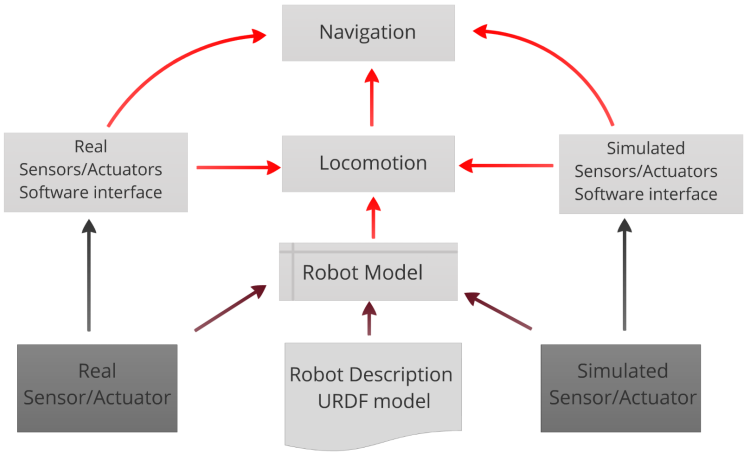
(a)

Figure 3.3: Robot laser scanner range coverage diagram (a) Top view of laser scanner range coverage where blue arrow represents the robot's orientation and the sensing area of the laser scanner is represented in green.

3.2 Introbot's simulated Model

The use of simulation was indispensable in the development of the system presented in this dissertation. Despite the clear differences to real world situations and restricted complexity, it permitted to surpass the limited amount of time and resources for testing using the Introbot. The parallel development of hardware components meant these opportunities were scare and so the capability of testing new software developments in a simulated environment facilitated the development cycle. Therefore, a simulation setup was created based on the ROS framework that provided both software and hardware in the loop capabilities. The simulation was based on the Gazebo simulator [5] for ROS. A robot model was created using the Unified Robot Description as well as simulated sensors and actuators. Additionally for a better simulation performance and fidelity there were also developed software simulations of the Introbot sensor and actuators interface (see Figure 3.4). In this section will be presented the URDF model creation, the sensor actuator controllers and interface software simulation as well as simulation worlds.

[5]Gazebo: `http://gazebosim.org`



(a)

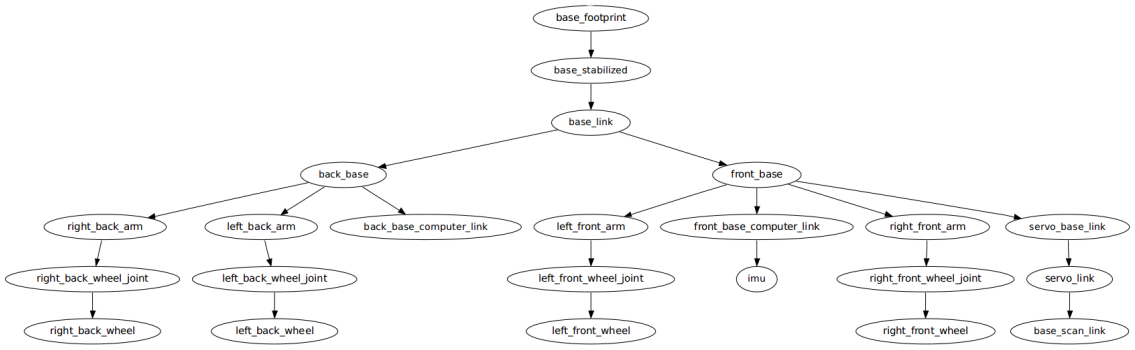
Figure 3.4: Overview of robot model architecture (a) Robot model architecture, by adding a simulated layer of sensor and actuators software interface, the same robot model and Unified Robot Description Format (URDF) description can be used for the simulated and real robot. In this figure the dark grey/red arrows represent low level sensor streams and the transforms tree messages of sensor/actuator respectively. The red arrows are higher level messages (ROS topics) exchanged by nodes or groups of nodes (stacks). The robot model is loaded into the ROS master and becomes available to all the nodes in the system such as the locomotion node or navigation stack.

To fully integrate a robot within the ROS framework a model representation has to be created, for this purpose ROS relies on a description language based on Extensible Markup Language (XML) called URDF. The language depicts models in a tree structure and can only be used on robots complaint with this type of representation. Nevertheless, the modelling process is not limited only to the description of mechanical properties, it also must contain additional information about sensor placements and part dimensions. In the next paragraph a summary of the inner workings of the URDF language is presented to improve the reader's understanding of its limitations and possibilities.

The URDF language is based on two major type of components, links and joints, a link is basically a part that has both visual and collision representations, whilst a joint simply unites two different links. A joint has a parent and a child link and it can be of flexible or inflexible nature. In inflexible joints only the origin in relation to the two links is defined, flexible joints however have another set of characteristics mainly concerning the kind of displacement they allow. There are five types of flexible joints: the revolute joint that rotates a certain angle around a defined axis; the continuous joint that is a special type of revolute joint that can rotate indefinitely; the prismatic joint that moves along an axis, not around it; the planar joint that has translational movement around two axes; and, finally the floating joint that is basically and unconstrained joint.

By incorporating and modelling the necessary components a movable mechanical model of the Introbot was created using this language (see Figure 3.5). Furthermore, sensors and actuators were added by using already created Gazebo plugins, this was the case of both the laser scanner and IMU (Inertial Measuring Unit). Moreover, additional care was taken in the parameterization to better mimic their real counterpart. However, the GPS simulation was not possible using standard plug-

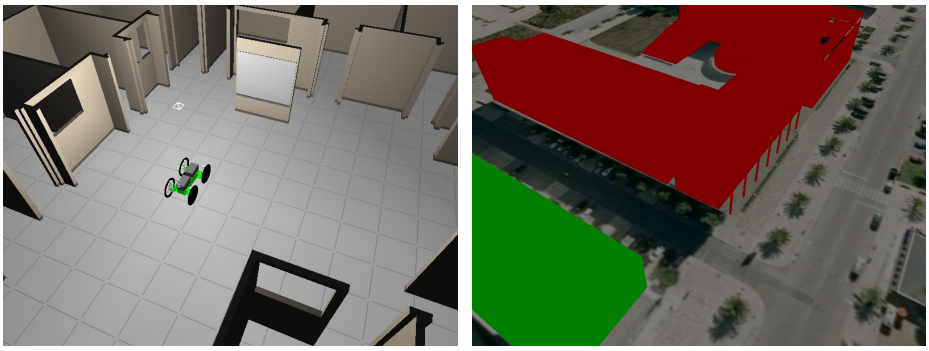
ins so a new one was developed using geographic_ info [6] ROS package. By exploiting a not fully implemented parameter in the description of a simulated Gazebo world, it was possible to create a new reliable GPS simulation. The parameter in question was used in conjunction with the provided ground truth positioning and the assumption that the X axis of the reference frame points north and the Y points east. Then, basically the ground truth position was added to the centre position to get a (Universal Transverse Mercator) UTM geo-referenced position. The UTM position was then converted to longitude latitude, additionally a Gaussian error was also added to get a more reliable GPS simulation. Actuators were also modelled using tools provided by ROS, a PID controller simulator was implemented for each wheel both for traction and steering. The controllers tuning process was carried out by using the Ziegler Nichols method (Ziegler and Nichols, 1942) to adjust the motor response to better mimic that of the real robot. Despite all of the similarities to the actual Introbot, it



(a)

Figure 3.5: Robot model transform tree (a) The robot model transform tree is the structure that maintains the relationship between coordinate frames in a tree structure, each ellipse represents a frame.

was also necessary to simulate the intermediate layer (controller nodes) that connected the low level PID controllers to the higher level kinematic control. This step added the possibility to transition from the simulated model to the real robot in a seamless and effortless way.

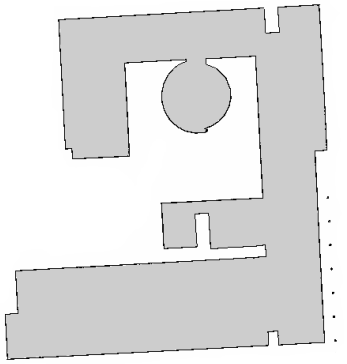


(a) (b)

Figure 3.6: Simulated Worlds (a) Snapshot of Indoor world. (b) Snapshot of Outdoor world both red and green represent $3D$ models of buildings .

[6] Geographic Info: `http://ros.org/wiki/geographic_info`

Finally to complete the simulation setup it was necessary to turn to the simulation of the environments the robot was designed to travel. This meant that two kind of settings were needed, first an indoor world and second an outdoor large scale ambiance. For the indoor simulation there was no need for modification and so the stock office world that comes with gazebo was used (see Figure 3.6(a)). This world represents the inside of an office building, several office furniture models and other objects were inserted during the simulation tests including tables, chairs etc.



(a)

Figure 3.7: Simulated mathematics department offline map (a) Image of generated map where grey stands for unknown space, white represents free space and black are lethal obstacles.

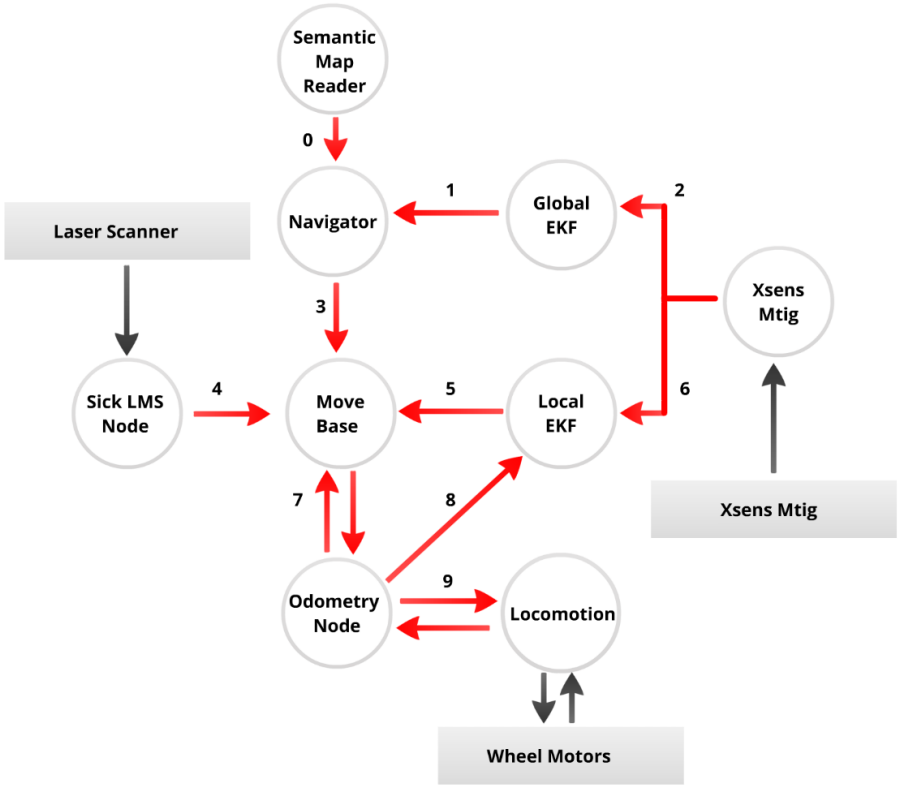
For the remaining tests another world was designed to simulate the grounds of the Faculty of Science and Technology of New university of Lisbon (Figure 3.6(b)). To achieve this effect first a satellite image with approximately 500 for 250 m of the selected area was obtained, this image was then overlaid in Gazebo on top of the simulation ground plane. To further increase realism existing 3D models of Faculty buildings were added, as well as some trees and diverse objects. This world was essential to test the transition between environments by enabling the realization of offline maps (see Figure 3.7) of certain key areas, that then were used for both single semantic zone navigation as well as the semantic zone transitions.

The ability to simulate this type of environments proved to be crucial in finding problem areas, moreover, it aided the testing new software developments and parameters. In sum the simulation capability of this system provided tools invaluable to both the basic functions (motion, kinematic control, sensor streams) as well as higher level capacities (navigation, perception algorithms).

3.3 Introbot's Software Architecture

The Introbot software architecture was develop based on ROS framework a general description of the framework is displayed below, followed by a more detailed account of the actual Introbot software architecture.

The ROS framework was created to deliver appropriate tools to manage the complexity of software needed to operate a robot, For example the large service robots have multiple processing units and sensors and high bandwidth demands because of the large amount of data exchanged between them. To resolve these issues the ROS framework already provides standard operating system services such as hardware abstraction, low-level device control, message-passing between processes and package management. Joining all of these functionalities the framework provides tools for distributed computing development, capable of running applications across multiple computers. The basic architecture is based on a peer-to-peer graph like topology, in which nodes process data asynchronously at the same time that they can receive, post or multiplex messages, which can carry any type of previously defined data such as sensor control, actuator state and planning messages.



(a)

Figure 3.8: Software Architecture (a) Overview of the software architecture, each number represents a message stream between nodes. The red arrows represent message passing between nodes, the grey arrows represent sensor/actuator data acquired by the node using the sensor/actuator API. (0) std_ msgs/UInt16 Semantic Zone, sensor_ msgs/NavSatFix (1) geometry_ msgs/PoseWithCovarianceStamped (2) sensor_ msgs/NavSatFix (3) move_ base_ msgs/MoveBaseActionGoal (4) sensor_ msgs/LaserScan (5) geometry_ msgs/PoseWithCovarianceStamped (6) sensor_ msgs/Imu (7) geometry_ msgs/Twist, nav_ msgs/Odometry (8) nav_ msgs/Odometry (9) sensor_ msgs/JointState, sensor_ msgs/JointState

Extra care was taken to make the Introbot software architecture fully compliant with the ROS syntax. By using the topics concept the nodes may publish and subscribe to the messages of their interest while, processing information in an asynchronous manner. This system is organized in five main groups (Figure 3.8), in the following paragraphs the groups the nodes and their interactions will be explained in greater detail.

- Low level control and sensor interfaces, in this group are included all the sensor/actuators low level interfaces.
- Locomotion control, this group is responsible for the locomotion of the robots composed by two nodes the kinematics/odometry node and Locomotion node. The odometry node receives a message that represents the desired linear an angular velocity (*geometry_ msgs/Twist*) and transforms it into kinematic valid wheel velocities and steering angles (*sensor_ msgs/JointState*), it also receives wheel encoder data stream (*sensor_ msgs/JointState*) and then publishes an estimated robot pose in form of a odometry message (*nav_ msgs/Odometry*) . The locomotion node receives wheel velocities and steering angles and sends this desired values to the motor controllers and also collects wheel encoder data in form of a ROS message (*sensor_ msgs/JointState*).
- Perception is composed by the Sick LMS Node and Xsens Mtig Node. The Sick LMS Node publishes the planar laser data stream in a ROS complaint message (*sensor_ msgs/LaserScan*). The Xsens Mtig Node publish two different types of messages and 3D orientation message (*sensor_ msgs/IMU*) and a GPS message (*sensor_ msgs/NavSatFix*)that are both used for localization.
- Navigation is composed of a group of nodes in charge of semantic label extraction, large scale navigation, path planning and obstacle avoidance behaviours (Navigator, Global EKF, Local EKF, Move Base). By receiving and fusing sensory information (*sensor_ msgs/LaserScan*, (*sensor msgs/IMU*), position and velocity estimates (*nav_ msgs/Odometry*) and desired goal poses (*move_ base_ msgs/MoveBaseActionGoal*), provided respectively by the perception nodes, odometry and Navigator, it publishes a path to the desired goal as well as the next angular and linear velocities (*geometry_ msgs/Twist*) to the locomotion control node. This is the main contribution of this thesis and so will be the subject of an analysis in greater depth in Chapter 4.

Chapter 4

Flexible Navigation System

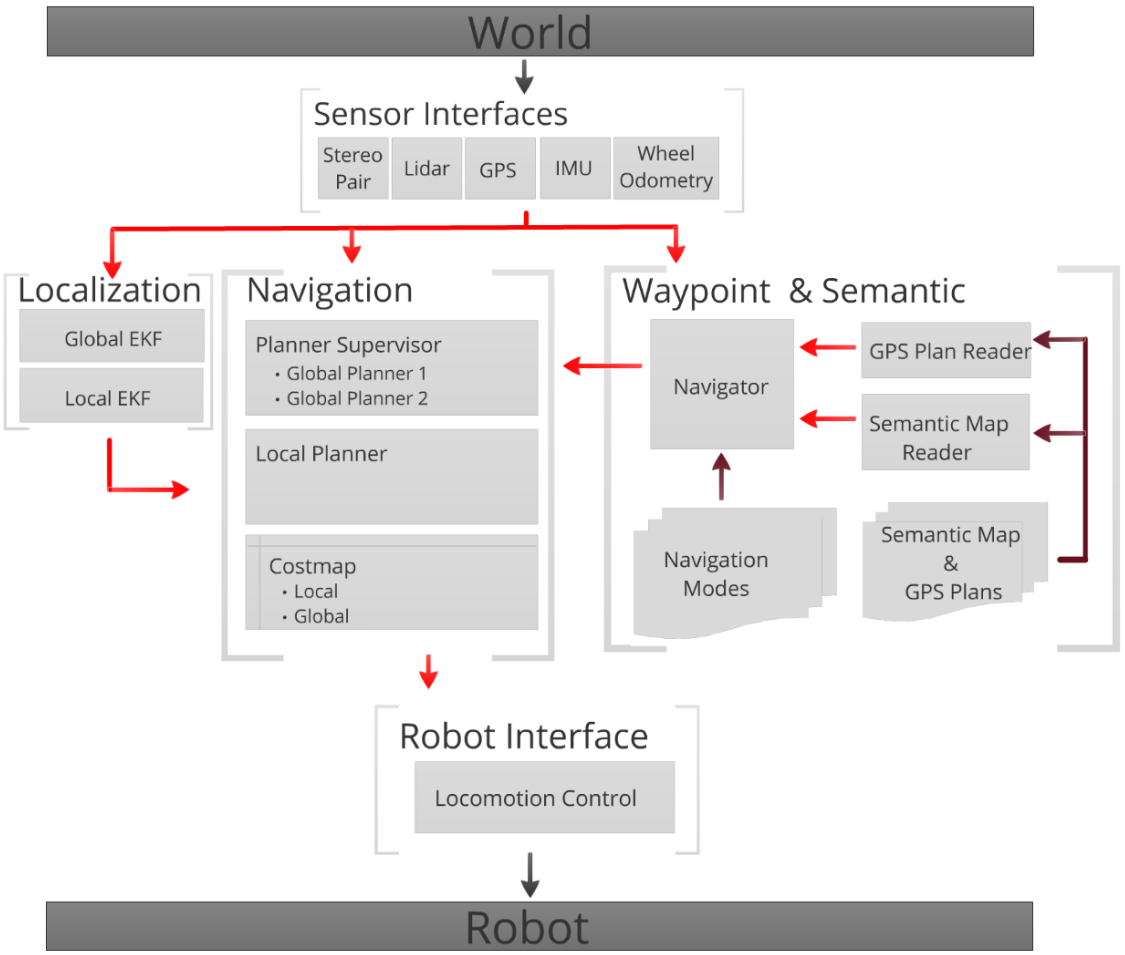
The major contribution of this thesis is the navigation module developed. To better cater the needs of the navigating in complex and heterogeneous settings, severe modifications were needed to the traditional architecture of an autonomous robot navigation system. This chapter presents an overview of the modifications and their interactions (see Figure 4.1) followed by an analysis of each component in greater detail.

As it is evident by the statements above any navigation architecture contains a large amount of data being exchanged between its components and also from and to the hardware of the platform. This is one of the reasons for using the ROS navigation stack as a starting point, because its framework already provides support for message passing and synchronization. Also the ROS navigation stack already has a functional basic architecture (Marder-Eppstein et al., 2010) that contains several of the traditional navigation components, such as:

- Occupancy grid;
- Global Planner;
- Local Planner;
- Recovery behaviours.

Despite the good results of most of the basic navigation architectures evidenced in some type of situations. The majority of basic architectures also have some limitations that restrict the scope of possible tasks. The most meaningful limitations to the type of system this thesis tries to create are enumerated below:

- Inability of dealing with large-scale environments;
- No capability of adjusting online to the environment;
- Limit to one online global planner;
- Limit to fixed footprint type robots;



(a)

Figure 4.1: Flexible navigation system architecture overview (a) The navigation system joins many different nodes and stacks, this figure shows an overview of the different nodes and interactions inside the navigation system. Each red arrow represents an interaction utilizing ROS topics, each bracket contains a category of nodes, the red arrows are ROS topics, the black arrows are low level sensor/actuator messages, the dark red arrows represent parameters.

- Only designed for differential drive and holonomic wheeled robots only;
- Developed for a square robot, so its performance will be best on robots that are nearly square or circular.

Is this type of pitfalls that the new developed navigation module tries to respond and resolve, enabling the robotic platform with the capability to navigate in complex and diverse of environments. A new set of modules, plugins and additional tools where designed to address these problems.

The system encompasses the following new functionalities. The first reads semantic map and retrieves the type of zone in which the robot is located (see subsection 4.1). Next a control center with a path design tool is able to draw geo-referenced waypoints paths and also extract diverse information from the robot (see subsection 4.2). To deal with the limitation regarding large-scale navigation there is a high level supervisory module (see subsection 4.3). Another area with supervisory skills is

the global planning, to enhance the planning proficiency a supervisory module can choose online between different global planners(subsection 4.4). Finally the modified local planner is capable of dealing with quasi-omnidirectional robots with dynamic footprints(subsection 4.5).

4.1 Semantic Labels and Environment Types

To enrich the situational awareness of the autonomous mobile platform there was a need of higher level information, in other words the limited scope of the perception tools available to the robot do not provide enough information, both in quality as well as in quantity to empower the robot to travel in a multitude of environments.

The approach taken in this thesis is to furnish *a priori* information by semantic labelling overhead satellite imagery. In this iteration of the stack only two different labels were used, one represents an open space while the other symbolizes a narrow space area. When talking about open spaces the label encompasses diverse environments with the common feature being as the name entails large open spaces, normally natural or rural ambiances. Narrow space areas can also represent a large range of settings, with the prevalent characteristics being the large amount of obstacles as well as the absence of GPS signal, these a normally urban settings both indoor or outdoor.

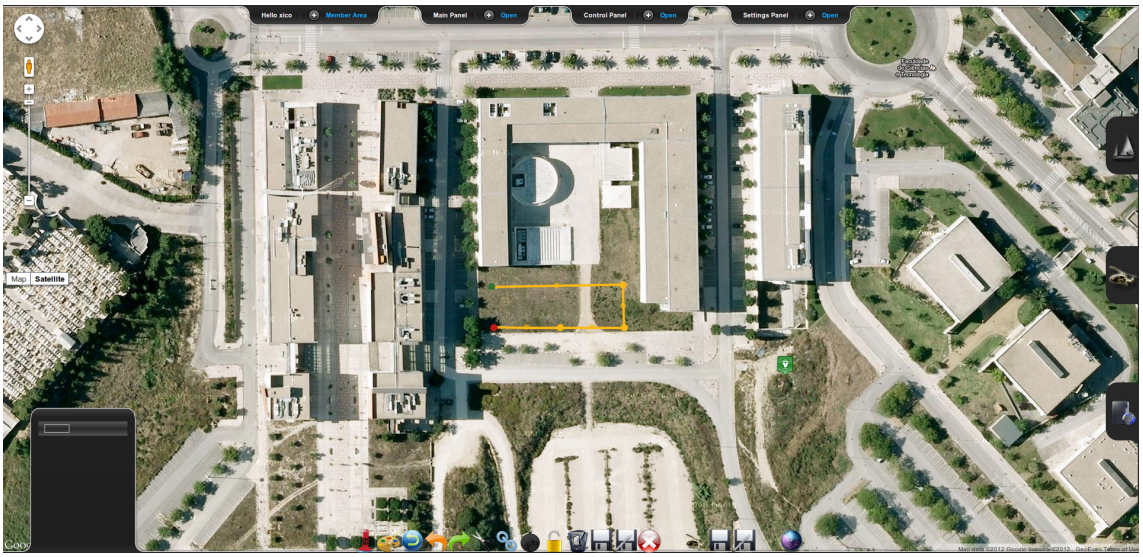
Using this information the system can by using its flexible nature adapt to the requirements and challenges of the aforementioned types of surroundings. The nature of these challenges as well as the steps the navigation stack takes to respond to them will become clear in the following subsections.

4.2 Control Center and Path Design Tool

For the control center and path design the choice was a Web User-Interface based on Google Maps API [1], designed to serve as a control center to monitor the position and infer high-level path and behaviour planning to the robot system of the Introbot platform. The Google API was a first stepping stone to the application construction, as it already had built in waypoint drawing and positioning over a map in a well adopted user-friendly environment. Above a layer of multiple features intended to smooth the path design, the application gathers a special group of traits:

- User register (login) system;
- Management of user paths files in accordance to known GPS standards (GPX [2] ,KLM[3]) verified with distributed xml schemas;
- Asynchronous Communication with the robots via Dropbox[4] synchronization;
- Live tracking of the current position (or path) of the robots on the map.

[1] Google Maps API `https://developers.google.com/maps/`
[2] GPX `http://www.topografix.com/gpx.asp`
[3] KLM`https://developers.google.com/kml/documentation/kmlreference`
[4] Dropbox `http://www.dropbox.com/`



(a)

Figure 4.2: Snapshot of path design tool (a) Snapshot of path creation using path design tool, orange represents a designed path, green circle is the path origin, red circle is the path end.

The application was constructed as a Java Server Page assembling Javascript (Google maps API and JQuery Library [5]) to both implement the path design over the Google Maps and ease the control panels in and out of the application, and AJAX to request any user-related feature, such as the logging in or the file submission. The unreliable nature of the communication with both outdoor robots over remote areas led to the decision of introducing also an asynchronous mean of communication to complement the ROS message service, a file like synchronization over the internet through Dropbox, had to be the solution, given its simplicity and practicality. The auto-planner option provided renders a simple and intuitive approach to path planning. An image of the region of interest defined by the two selected points is retrieved from Google Static Maps API and a path planner based on Field D* Algorithm (Ferguson and Stentz, 2007) provides a list of waypoints that are then marked back on the map as new path or subpath.

4.3 Waypoint Navigation

Large scale navigation is one of the necessities of autonomous robots. Thus, in the proposed system a node takes on the role of supervising the navigation stack enabling it to perform large scale navigation. This subsection is dedicated to explain the inner workings of this node.

Initially the node receives a plan created in the control center, containing an array of geo-referenced waypoints that form a path to be followed by the robot. Given such plan the waypoints are transformed

[5] JQuery `http://jqueryui.com/`

(a) (b)

Figure 4.3: Semantic mode waypoint to goal calculation method (a) Narrow space , first the distance from robot to map center (red square) is calculated (blue arrows), then distance from goal to map center also computed (orange arrows) and finally goal (green circle) can be sent to robot global navigation path planner (green arrows). (b) Open space, distance from robot to goal is calculated (blue arrows) and goal (green circle) is sent to robot global navigation path planner (green arrows).

to navigation goals, however, the goal transformation process is semantic zone dependent. In other words the waypoint to goal process has different steps according to robot's semantic location.

However, both transformations share a similar step, the geo-referenced positions are transformed from longitude and latitude to UTM coordinates. UTM coordinates represents the world in two-dimensional Cartesian coordinates, this is achieved by dividing the world in sixty zones and using a secant transverse Mercator projection in each one. Therefore, with the robot position and waypoint/goal position represented in UTM coordinates and in conjunction with the robot's heading its possible simply calculate the Euclidean distance between them, this distance is then fed to the robot navigation.

Whilst this suffices in the case of the Open Space where there are only this two positions and the reference frame is robot centric. In the Narrow Space semantic zone the goal has to be map referenced. To solve this issue the map centre position is also geo-referenced therefore possible to transform to UTM coordinates. Thus allowing the use of simple Euclidian distance, to transform a waypoint/goal from a geo-referenced waypoint to an offline map referenced position (see Figure 4.5).

Another issue dealt by the supervisory node is the control of the waypoint following behaviour. This is performed by maintaining a close watch of the robot's position in relation to the previous, current and next goals. Based on that waypoint information and in a set of rules provided it calculates to which of the waypoints it needs to plan next.

To provide flexibility in dealing with unfamiliar terrain, the following set of rules are included into the system. This is of the outmost importance mainly because of the limitations inherent to the construction of the waypoint path. Albeit an intuitive and simple way to create paths the use of offline satellite imagery has intrinsic problem. The images are usually outdated and obsolete that in conjunction the satellite photo localization errors. Therefore a waypoint apparently on an obstacle free space, can now be on top of manmade structure or a natural obstacle like a river stream or group of trees. For example, if the satellite takes a photo of a location during the winter, you see clear spaces that might be filled with vegetation in the spring. In the case of an urban setting there might be a new building on what was previously open space. With this in mind the next set of rules were created:

- If the robot is inside a user defined waypoint tolerance radius then it proceeds to the next

waypoint;

- If it's closer to the next waypoint than the current one then proceed to the next waypoint;
- If it's more than half distance to the next waypoint than the distance between the current waypoint and the next then proceed to next waypoint.

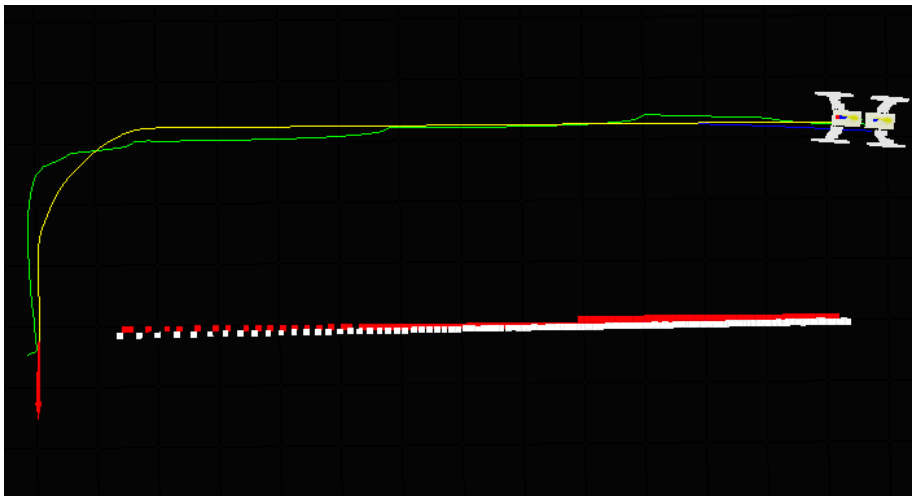
Another function of the supervisory node is the management of the navigation behaviours linked with each of the semantic zones (see Section 4.1). For each zone there is an associated navigation behaviour parameterizable by the user. This settings are then stored to a file respective to the corresponded semantic zone. When transitioning to the semantic zone the node loads a navigation parameterization file, that in some cases also includes launching and killing of nodes necessary for that type of navigation. For example when entering a narrow space zone, the localization is based on an offline map, so the navigator node launches the Monte Carlo Localization node. Subsequently, loads the offline map associated with that region and also the global and local planner parameters adequate to that zone.

The localization and transition between semantic zones is also in the realm of the capabilities assigned to the navigator node. In other words based on the info provided by the semantic map reader, the navigator node has to take the decision when to transition for one mode to another. The control mechanism conducts a permanent verification if the current navigation mode is the same as the correspondent semantic label provided by the semantic map reader. If that is not the case the desiredness of changing to a new navigation mode increases and if it reaches a previously set value a semantic transition occurs. However, if the desired navigation mode is equal to the current mode then its desiredness to change drops to zero maintaining the same mode. This prevents sudden jumps in navigation modes when in a boundary zone, thus only if the new mode is desired continually will a transition be trigger, if not the current mode is preferred.

4.4 Global Path Planning

The start of a navigation process is normally the creation of a global plan, its function is to provide a feasible path to a desired goal. One of the objectives behind the development of this navigation stack was to adapt all of its parts to deal with different surroundings. Hence, in the case of global planning, this could mean that a certain global planner can be better suited for a type of environment than another. For this reason a global planner supervisor was created to permit the online exchange of global planners. When active the node is always waiting for a new goal, in this case it is also waiting for the information of what planner to use. This permits the use of a multitude of planners and play with their advantages and disadvantages without ever needing to stop the robot. While in some situations this could not provide a performance enhancement in others it could mean the difference between being able to find a feasible plan or not (e.g. doorway traversing) .

In this iteration of the software there are two global planners available, the Nafvn Planner that is a Dijkstra algorithm ((Dijkstra, 1959)) based planner and also the SBPL Lattice planner that is a motion primitive based planner ((Likhachev and Ferguson, 2009)). In Figure 4.4 the same goal is pursued by two different types of planners. If a smoother plan is needed the SBPL planner could be used but



(a)

Figure 4.4: (a) Same goal (red arrow) with path generated from different planners, the path of Sbpl lattice is represented in yellow and the Navfn plan is displayed in green, obstacles are displayed in red and laser scanner hits in white

at the expense of computation time, if the computation was the key issue the option should rely in the faster Navfn planner. In sum the global planner supervisory node adds another layer of flexibility to the whole navigation system by allowing the online exchange of global planner algorithm.

4.5 Local Trajectory generation

For the local trajectory generation an instance of the trajectory roll out (Gerkey and Konolige, 2008) is used, where as previously explained in Chapter (2) the idea is to sample the robot control space, then simulate what would be the position of the robot if the sampled velocity was applied for a short period of time. The velocity control space is divided into three components dx represents forward motion, dy is the side motion component, and $d\theta$ represents the turn in place velocity. Each of this sampled velocities will be carried over the entire forward simulation period given the acceleration limits of the robot and generate a trajectory that will be scored given a certain metric. Then, the one with the lowest cost will be picked to be transmitted to the robot base in (dx,dy,$d\theta$) format.

One limitation of the original roll out algorithm is that it does not encompass the requirements of robots with quasi-omnidirectional locomotion and dynamic footprints. To cater to this needs several modifications were made to the original software that will be summarized in the paragraphs below.

First a set of robot model specific parameters were created to deal with the motion constraints. The Introbot platform has a set of specific locomotion characteristics as explained in Section 3.1 of Chapter 3. The Introbot's capability of changing locomotion type creates a problem to the trajectory

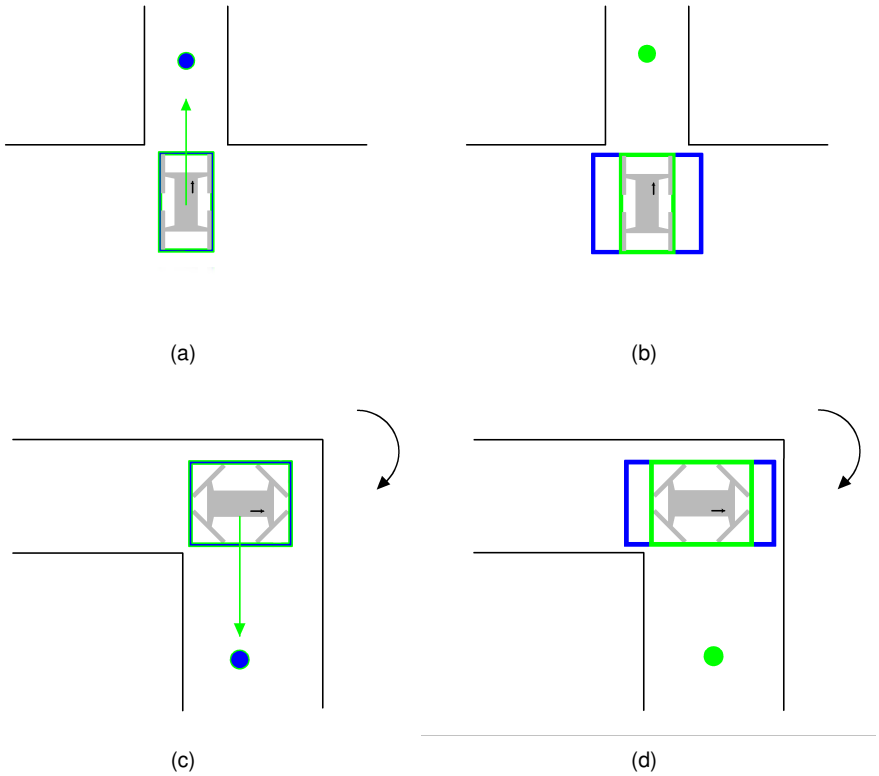
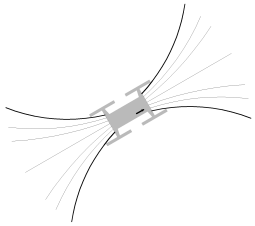


Figure 4.5: Two examples of the advantages of dynamic footprint versus a static footprint. In the first case (Figures (a) and (b)) the next goal is in a narrow hallway. In the second case (Figures (c) and (d)) a sharp ninety degree turn is needed to reach the goal, that only can be achieved by rotating in place. (a) Successful case, where the real footprint (green rectangle) is the same as the trajectory creation footprint (blue rectangle), this means the robot can reach the proposed goal(green circle). (b) Failure case, the trajectory footprint (blue rectangle) encompasses all the possible locomotion modes and so cannot pass through the narrow hallway even though the real footprint in that locomotion mode is narrower than the hallway (green rectangle). (c) Successful case, the overlap of the real (green rectangle) and trajectory creation (blue rectangle) enables the robot to turn in place to the goal (green circle). (d) Failure case, the increased footprint used for trajectory creation makes it impossible to turn in place even in there is clearly enough space with the real footprint (green rectangle).

creation, because some of the sampled velocities are impossible to the robot. For example the robot has maximum turn radius that it can perform in Ackermann or double Ackermann mode (see Figure 4.6), therefore below this radius the trajectories cannot be performed by the physical robot base. To answer that constraint a maximum turn radius parameter was introduced limiting the generation of trajectories. Another problem arises because of open wheeled nature of the robot, even in the same locomotion mode the footprint changes and that has to be taken in account when verifying obstacle collision, or it would severely impair the manoeuvrability of the robot in certain situations.

On the other hand if the footprint has to be sufficient to enclose all the wheel positions attainable by the robot, it would create difficulties when transversing narrow spaces or rotating in place. To solve this limitation each velocity sample was assigned an associated footprint to be used in the obstacle collision verification.

Another particular characteristic of the Introbot platform is that to perform certain trajectories it has to change locomotion mode. Consequently, this locomotion change has an associated cost both in energy and time that needs to be optimized. It can also has a different side effect of making the robot indecisive between locomotion modes. This happens when trajectories from different locomotion



(a)

Figure 4.6: Robot's possible trajectories (a)Each line represents one possible trajectory, the darkest line represents the maximum turn radius achievable by the robot, the black arrow shows the robot's heading.

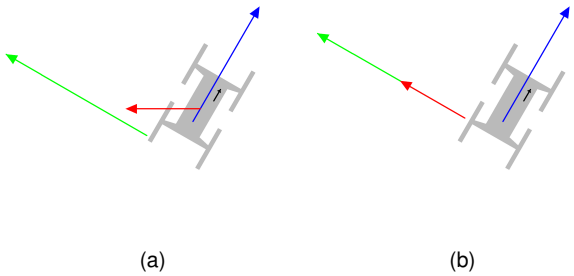
modes have a very similar cost, to solve this a couple of rules and parameters were introduced. The chosen approach to this problem was to penalize the diverse locomotion modes in certain situations, those penalizations are enumerated below.

Like the original roll out algorithm forward motion trajectories are favoured but now the plan alignment is also considered. If the robot pose is aligned with the plan heading the in-place rotations and escape velocities are penalized and so the robot tends to perform forward trajectories if they are possible. If the robot's heading becomes misaligned with the plan for more than a 45 degree angle and it is possible to rotate to the plan, the forward motion trajectories are penalized and so the robot favours turn in place rotations. Only when aligned in a 10 degree delta are the forward motion trajectories penalizations lifted, introducing a hysteresis to filter the indecisions on the boundary situations.

(a) (b)

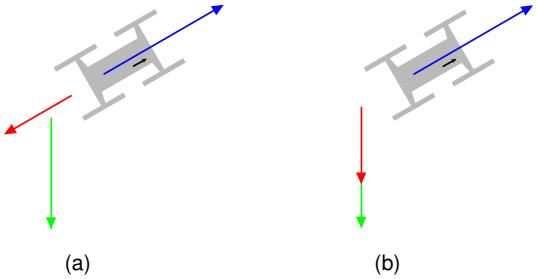
Figure 4.7: First situation of plan heading calculation, green arrow is the generated global plan, blue represents the robot's real heading and red arrow is the calculated heading. (a) Original Algorithm. (b) New Algorithm.

Another type of penalization was introduced to force the robot when rotating to the plan to rotate in direction of the shortest angle. However, the aforementioned rotation penalization is only possible because of modifications to the plan heading calculation. The original implementation calculates heading difference by finding the first free clear line of sight between the robot and a point on the path. While this gives an estimate of the heading difference it can also lead to erroneous behaviours as can be seen in Figure 4.7(a). In that particular case the heading difference will be 90 degrees,



(a) (b)

Figure 4.8: Second situation of plan heading calculation, green arrow is the generated global plan, blue represents the robot's real heading and red arrow is the calculated heading. (a) Original Algorithm. (b) New Algorithm.

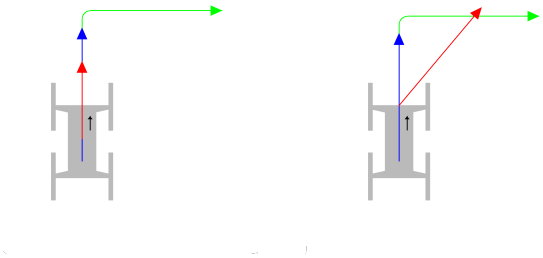


(a) (b)

Figure 4.9: Third situation of plan heading calculation, green arrow is the generated global plan, blue represents the robot's real heading and red arrow is the calculated heading. (a) Original Algorithm. (b) New Algorithm.

but in reality the robot is aligned with the plan. Despite that the scoring function of the algorithm will probably still generate a forward trajectory to intercept the plan a couple of centimetres ahead. Nevertheless this will also fail in certain situations, such as the ones represented in 4.8 and 4.9. Where the robot should rotate in place to align to the plan, but the heading difference estimated is larger than the real heading difference between the robot and the plan. Consequently, this means that there is no way to know what is the actual difference of the alignment to the plan and the correct direction.

To solve this problem the plan direction was calculated by picking two points of the plan separated by parameterized distance. Subsequently by finding this direction and comparing it with the heading of the robot it is possible to find a more accurate heading difference. The results of this modification can be observed in Figure 4.7(b), 4.8(b) and 4.9(b) this returns a more accurate heading difference. However it can also fail when there is sudden changes of direction in plan heading Figure 4.10(a) but even in this situations it provides a more accurate result than the previous method (Figure 4.10(b)).



(a) (b)

Figure 4.10: Failures cases of plan heading calculation, green arrow is the generated global plan, blue represents the robot's real heading and red arrow is the calculated heading. (a) Original Algorithm. (b) New Algorithm.

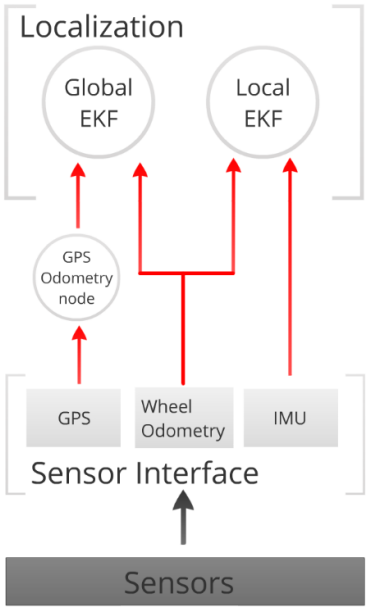
4.6 Localization

Localization is a key part of any navigation system, each of the semantic zones have different challenges regarding localization. In the Open Space areas normally robot has open sky and so the GPS signal is a good method of localization, however in Narrow Space areas normally there will be man-made structures or natural obstacles that will affect the GPS signal reliability. Thus in this case using other localization sources is more effective. For this reason the localization methods in this navigation system are semantic zone specific.

Two types of localization are relevant and used, the first a global geo-referenced localization and the other a local vehicle-centric coordinate frame. The first is only used for tracking the absolute position in relation to the semantic zones and geo-referenced waypoints. While the second references the obstacles in relation to the robot center, this approach keeps the absolute positioning errors from affecting the local navigation.

The implementation uses two different Extended Kalman Filters (Figure 4.11). In the ROS implementation each EKF is capable combining measurements from three different sources. First a 2D source that contains the position and orientation of the robot in the ground plane and the covariance on this pose. Second a 3D source that provides only orientation information, Roll, Pitch in absolute angles and Yaw is interpreted as a relative angle. Finally 3D source that represents the full position and orientation of the robot and the covariance on this pose.

The local localization EKF returns the position in a robot centric frame and has two different sources, the wheel odometry as a 2D source and also the IMU as a 3D orientation only source. While in the global or geo-referenced localization a second EKF is used, wheel odometry is again the 2D source and a GPS based odometry is used as the 3D source but only with 2D measurements. In the GPS based odometry latitude and longitude are converted to Universal Transverse Mercator (UTM) coordinates and the error of the GPS sensor estimate is used as the covariance of each measurement. Therefore by using both wheel odometry and GPS based odometry the EKF returns a more trustworthy geo-referenced position using the wheel odometry to filter both GPS signal blackouts and GPS position jumps.



(a)

Figure 4.11: EKF input sources (a) Global and Local EKF input Sources, where the grey/red arrows represent low level messages and ROS topics.

Table 4.1: Three frames are always present in the localization TF tree in all of the semantic zones, the only difference is who publishes the translation and orientation transform between them.

Semantic zone	Geo-referenced Position	Offline Map to Odometry Transform	Odometry to Base transform
Indoor Narrow Space	None	AMCL localization	Local EKF
Outdoor Narrow Space	Global EKF	AMCL localization	Local EKF
Open Space	Global EKF	Static publisher	Local EKF

Another type of localization is introduced when entering a narrow space area, in this case there is another intermediate localization frame is introduced. This frame is based on a global static offline map, and the localization in this map is provided by the AMCL

For simplifying the transition mechanism in the other semantic zones this frame also exists but it is the same as the robot-centric frame. When in an outdoor situation the map localization is based on AMCL. This is useful because in this type of the environment the GPS signal rapidly deteriorates and so using the AMCL localization will greatly improve the localization performance. In an indoor setting there no need of geo-referencing the position and so the global EKF position isn't used, only the static map localization remains relevant.

Chapter 5

Experimental Results

This chapter presents the experimental setup and parameterisation used to assert the robustness and efficiency of the proposed model, as well as the obtained results. The failures cases of the proposed model and the discussion of the results, are described in Section 5.2.

5.1 Experimental Setup

The proposed model nodes where implemented entirely in the C++ programming language and it was made fully compliant with the Robotics Operating System (ROS)[1] (Quigley et al., 2009).The system was tested in a Intel(R) Core i7-2670QM CPU @ 2.20GHz with 6 Gb of RAM, running a 64-bit Linux distribution Ubuntu 11.10 (Oneiric Ocelot).

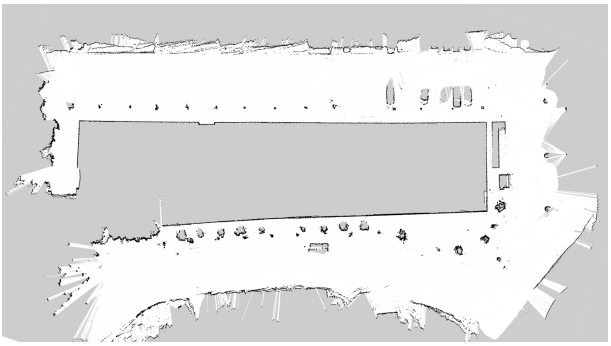
5.2 Results

This section presents an overview of the experimental results obtained in the final field trials, that were performed in two separate days, 24 of August 2012 and 30 of August 2012. The tests were carried out in the facilities of the New University of Lisbon organized in four different scenarios in which the robot travelled a total of 845 m at the average speed of 0.6 m/s (while in motion) and 30 minutes of operating time. In the first of the two days the robot was faced with the challenge of navigating in narrow space areas both outdoor and indoor. The subsequent test day an open space GPS waypoint following behaviour was performed as well as a transition between operating modes.

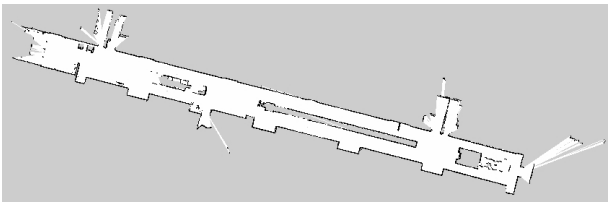
[1] ROS: `http://www.ros.org`

5.2.1 Narrow Spaces

This experiment aims to demonstrate the navigation module capabilities of handling navigation in narrow constrained spaces. The narrow space experiments were based on the building of the Electrical Engineering Department of the New University of Lisbon, the building is four stories high and approximately 80 m long per 22 m wide. Indoors, the building exhibits the layout of a typical office. the outside it is a generic urban setting with roads, sidewalks vegetation and parking spaces. Offline maps of the building were generated by tele-operating the robot (see Figure 5.1). These maps are provided to the navigation stack in order to have an a priori of the obstacles distribution across the environment.



(a)



(b)

Figure 5.1: Occupancy grid maps obtained from teleoperating the robot around the environment, using ROS SLAM gmapping node. Where grey stands for unknown space, white represents free space and black are lethal obstacles. (a) The outdoor map of the building. (b) The first floor of the building.

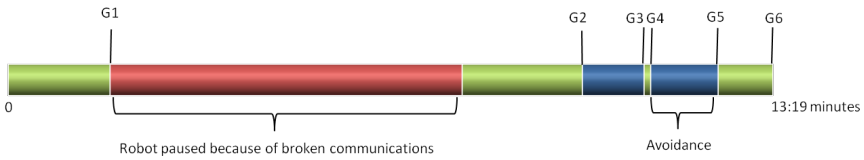
Indoor

The test area was the first floor of the building which is composed of narrow hallways and not traversable locations such as bottlenecks, stairwells and a ramp. The floor was filled with office furniture such as a few chairs and tables, several cabinets, vending machines, announcement boards and cardboard boxes. This testing ground was chosen because of the challenge that is navigating successfully in an environment, in which most of the hallways are only slightly bigger than the robot footprint and with dynamic objects that can cause instant obstructions.

In this test, the robot navigation stack was feed with a series of goals, provided by a human controller. The robot was required to transverse the narrow hallways without external intervention and to deal with the sudden emergence of dynamic objects, such as people (see Figure 5.4). In

(a) (b) (c)

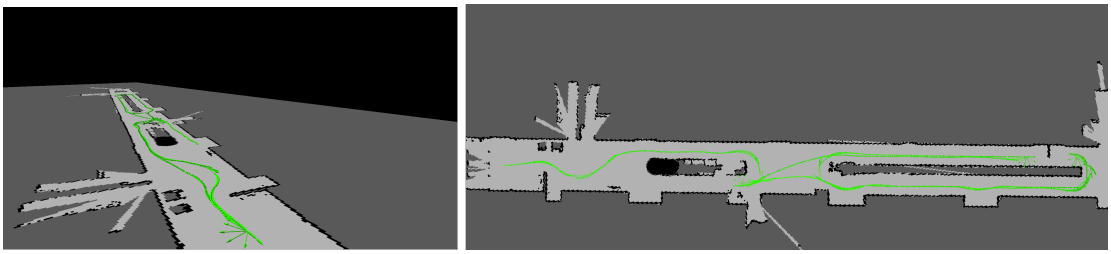
Figure 5.2: Indoor location. (a) Elevator door and narrow passages. (b) Bottleneck example. (c) Narrow hallway.



(a)

Figure 5.3: A timeline of the transition run where blue represents key moments, red situations where the robot was paused and green for a normal behaviour

total, there were six different goals fed to the robot, with five of them being successfully reached. The whole run took fifteen minutes, that were travelled at an average speed of 0.43 m/s. The goals were selected to test the capability of manoeuvring within such a narrow space, promoted several challenging turns including narrow gaps as well as promoting encounters with dynamic objects. In the next few paragraphs a brief summary of the run itself is displayed with brief descriptions of notable events that occurred.



(a) (b)

Figure 5.4: Path of the robot during the indoor test. Green represents the robot's path. (a) 3D perspective of trajectory. (b) 2D perspective of trajectory.

The first goal required the robot to move through a narrow passage and a long hallway, culminating in a sharp turn to face the bottleneck imposed by a protruding wall. As it is visible in Figure 5.5, on the first confined passage, the cabinet on the right cannot be fully perceived by the laser range

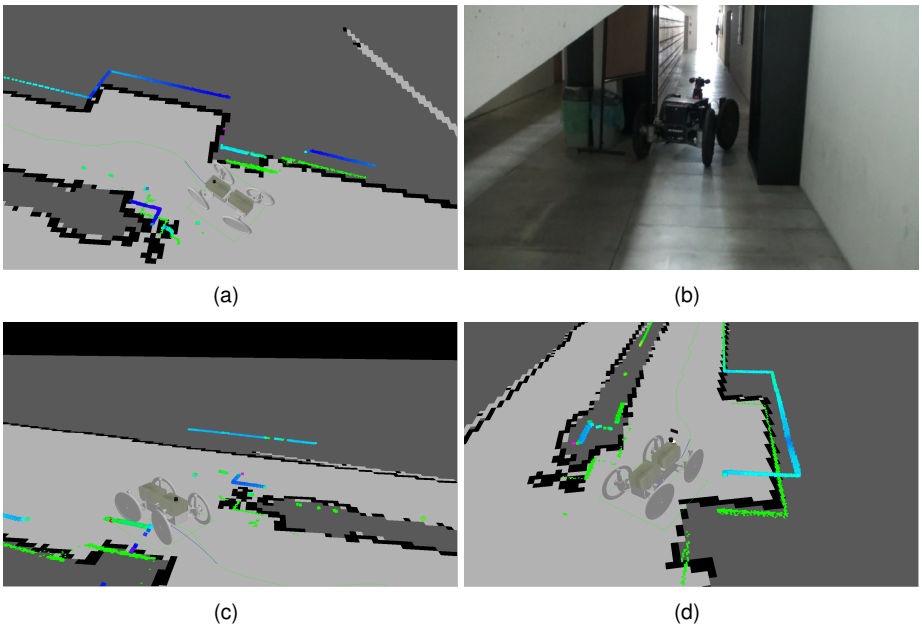


Figure 5.5: Robot going through the first turn. (a) Data replay of first turn approach. (b) Snapshot of first turn approach. (c) Data replay showing full cabinet. (d) Cabinet does not get fully perceived. Red global obstacles, green local obstacles, RGB laser scanner hits

finder. Despite the successful navigation, this situation unveiled a major problem, that could lead the robot to collide with certain objects that possessed physical qualities that absorb the laser beam, rather than reflecting it.

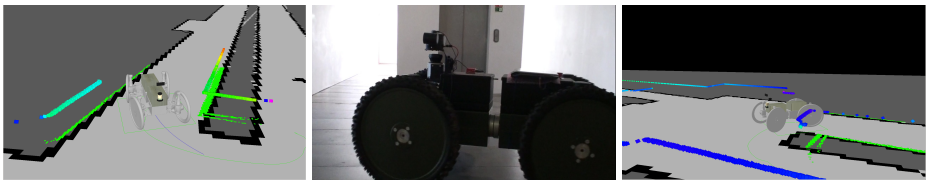
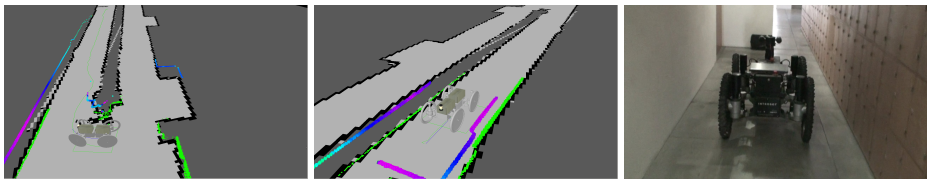


Figure 5.6: Robot dealing with u shaped turn. (a) Data replay of first part of u shaped turn. (b) Snapshot of robot turning. (c) Data replay of robot poorly localized. Red/green represent global/local obstacles, RGB marks are the laser scanner hits

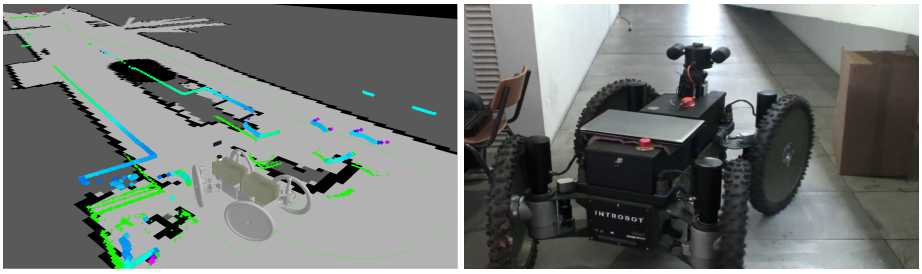
The next fast moving sharp u-shaped turn required the robot to be as close as possible to the row of cabinets and finally rest in the other side of them. The fast turn was to be a challenge to the robot's localisation stack, as it can be seen in Figure 5.6. Nevertheless, the robot reached the goal without engaging in oscillating behaviour.

The second goal sent the robot to rest on the other side of the cul-de-sac it was now facing. In this case, the route required another series of sharp turns and long straight in an even narrower hallway. Like before, there were no problems in navigating through a harder but similar route (see Figure 5.7).



(a) (b) (c)

Figure 5.7: Snapshot of the route to the third goal. (a) Robot in a u shaped turn. (b) Robot resting at the goal. (c) Narrow hallway. Red/green represent global/local obstacles, RGB marks are the laser scanner hits



(a) (b)

Figure 5.8: Bad localization caused robot to get stuck. (a) Robot not properly localized. (b) Frame of the same moment. Red/green represent global/local obstacles, RGB marks are the laser scanner hits

The next route showed to be more demanding and led to the only failure reported. The robot had to reach the front door of the building, which means that it had to choose between the two hallways. The robot initially chose to go through the hallway where the human operator was seated in a doorway. As soon as the saw the operator it picked the alternative hallway to follow.

The operator removed the chair he was resting his laptop to free enough space for the robot to rotate. As a result, some false positives (poorly cleared map regions) in the robot path emerged. Figure 5.9. A few time steps latter, the false positives disappeared and the managed to navigate smoothly to the goal. This situation was the major setback of the whole experiment, the too optimistic choice of path led the robot to get stuck not only because of the path itself but also because of the localisation problems that shifted the robot too close to the surrounding objects (see Figure 5.8). Despite this, the robot was never in danger of colliding either with the operator or the static objects, displaying a very safe behaviour.

Then the previous goal was resent, the robot selected the clearest hallway and proceeded to move towards the front door of the building, to be noted that this was the narrowest hallway of the whole mapped building. As it was moving on the corridor a person appeared from the stairwell situated on the right and moved onto the robot desired path.

That forced replanning of the desired route and plan through a narrow gap left between the wall and the person. The maneuverer was done swiftly and efficiently and prove no major challenge to the robot. The constant replanning showed a major asset in this particular case because as soon as

(a) (b)

Figure 5.9: False positives prevented robot from moving even if it clearly has space. Red/green represent global/local obstacles, RGB marks are the laser scanner hits

the new object appeared and moved, the planner reacted and made a new safe and possible plan. The next goal made the robot return to its starting place, the route taken was similar but shorter and faster because of the absence of the previous obstruction.

In sum, these experiments show that the robot's navigation stack provides the robot with sufficient skills for a robust operation indoors. Albeit structured and somewhat predictable, the sheer size of the robot in relation with its surroundings makes it highly demanding. The localisation of the robot was robust and never seem to drift too much of the externally perceived ground truth position. Also, the robot showed to be capable of avoiding obstacles in a safe and robust way.

Despite the success, there were a few difficulties that arose from these field trials. The dependability on the laser scanner as the only range sensing modality limits the navigation when in the presence of object with certain reflective physical characteristics, or simply because these are below the laser's planar field of view. A solution to this problem would be to exploit sensor fusion. For instance, a binocular vision system, a 3D laser scanner, and a time-of-flight camera could be used to extend the perceptual capabilities of the robot.

Clearing false positives, especially if these are in the immediate robot vicinities, also showed to pose some challenges. These false positives can cause the robot to mis-perceive the environment to the point navigation is impaired. The results showed that this problem is solved if the robot has enough time to accumulate evidence through its sensors. Nevertheless, the literature encompasses solutions for a faster clearing process, which can be exploited in future work. Communication dropouts were felt in some situations during the experiments. In key situations, for instance when the operator is trying to send a new goal to the robot, this temporary failure may hinder the overall usability of system.

Outdoor

The scenario off the narrow space outdoor field trial has the outside of the aforementioned building, the environment was picked has it displays a normal urban setting with roads, sidewalks, vegetation, parking spaces, cars, and pedestrians. The building is surrounded by trees and a grass patch in one side, in the front there is a staircase and a ramp, there are several parking spaces in the other

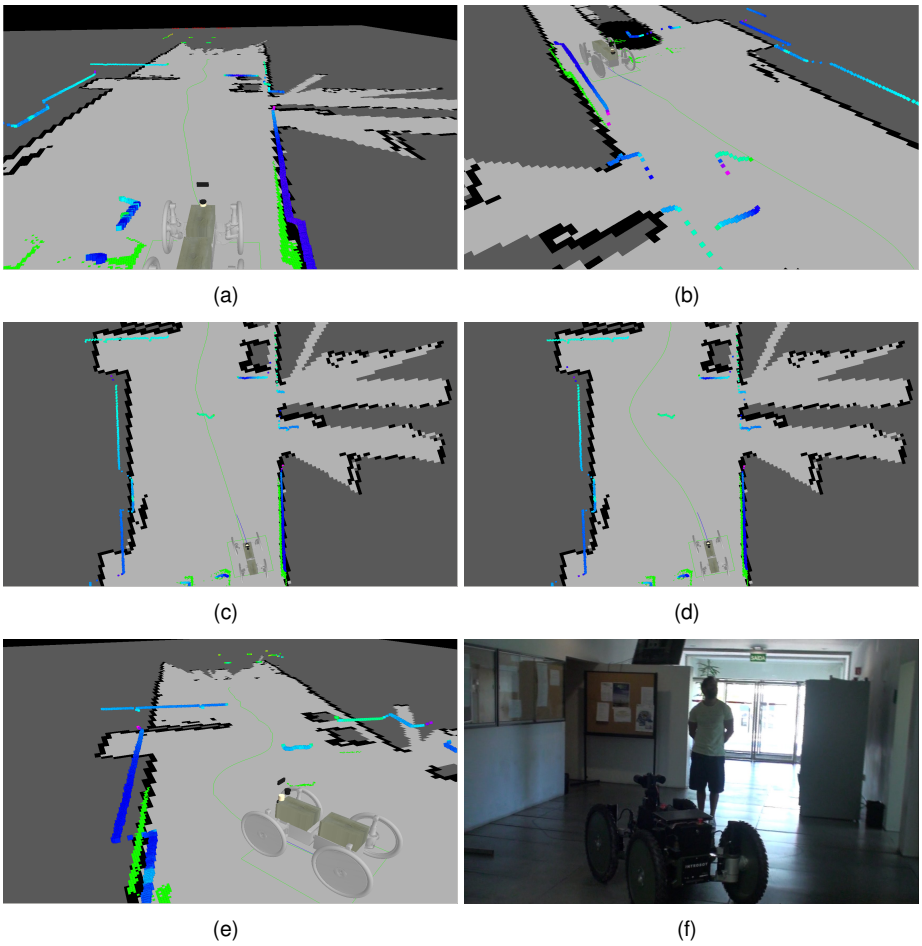


Figure 5.10: Avoidance of a dynamic object. (a) Robot has a clear path to goal. (b) A person appears from stairwell on the right side of the robot. (c) Object in the middle of path to goal. (d) New plan to go around obstruction. (e) Robot already started the avoidance behaviour. (f) robot on way to goal after replanning. Red/green represent global/local obstacles, RGB marks are the laser scanner hits

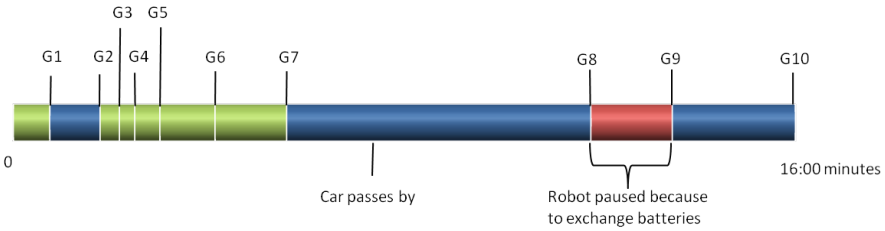
side, on the back of the building there is a dirt patch with debris and some light vegetation.

The terrain brings a new set of challenges, the most important is the non-planar ground that influences the detection of obstacles, a small hill that is transversable by the robot could be classified as an obstacle. Other difficulty that should be considered is that even if in theory the surroundings are more spacious for the robot to move there are other constraints that have to be considered, fast moving dynamic objects like cars should be avoided at all cost, so the robot should navigate primarily in the sidewalk around the building. Vegetation also brings difficulties as it can cause false positives, a small bush or a hanging tree branch could cause to get stuck or to not find a viable plan. In this section the events of the outdoor run are described as well as some snapshots of the most critical moments.

In the outdoor test the goals were mostly chosen at random but at same time some care was taken to provide challenging goals like going completely around the building, ten different goals were sent to robot all but one were achieved, localization problems also caused one of them to be askew, the robot took 14 m and 35 s moved at$0.48\,\mathrm{ms}^{-1}$ average speed covering 396 m.

(a) (b)

Figure 5.11: Outdoor location. (a) Side of the building. (b) Larger view of the building.



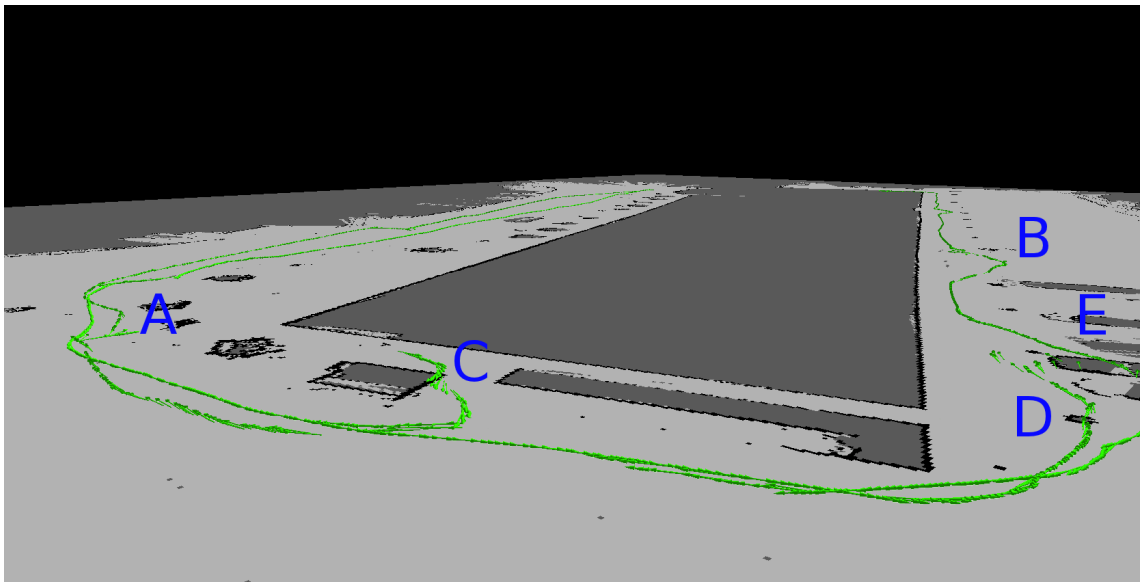
(a)

Figure 5.12: A timeline of the outdoor run where blue represents key moments, red situations where the robot was paused and green for a normal behaviour

The robot was initially placed below the balcony of edifice and requested to move five meters in front of the structure (see Figure 5.14). Just as it was in the starting position the ferromagnetic materials used to build the structure that generate a strong magnetic field influenced seriously the magnetometer, leading to a heading drift that caused an incorrect localization. In spite of this, the robot tackled the challenge of exiting such a constrained space successfully (see Figure 5.14) and without any risk of collision.

The next goal was near the building in-between the parking spaces and the wall. When the robot began to move a pedestrian came into to sight, and proceeded moving alongside the robot (see Figure 5.15) crossing the robot desired path who reacted promptly and adjusted is route.

The robot proceeded and was required to do a ninety degree turn, the bad localization influenced the capability of taking such turn, and so the robot under steered moving into the parking spaces believing that it was near the wall (see Figure 5.16). After the localization was corrected new goals at the front of the building were sent, after the robot reached all of them. Then the max speed was modified to $1.2\,\mathrm{ms}^{-1}$(see Figure 5.17) to test the behaviour at higher speeds.



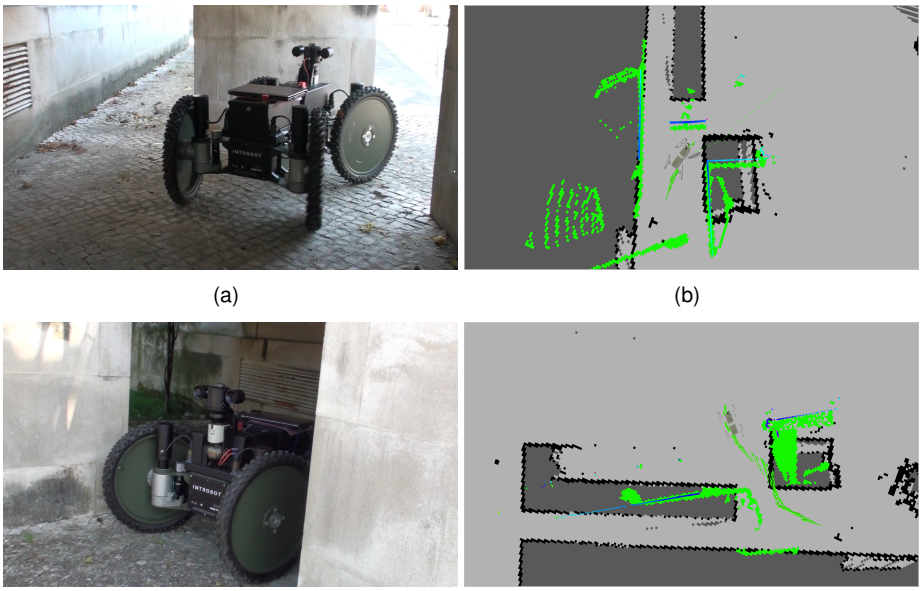
(a)

Figure 5.13: Robot's path (green plot) executed during the narrow-space field trial, overlaid on the offline generated map. Light grey, dark grey, and black represent free space, unknown space, and obstacles, respectively. Capital letters represent key situations in the course of the test. Situations A-B: avoidance of dynamic obstacles. Situations C-D: due to a slight mis-localisation of the robot, the path overlays the obstacles present in the off-line generated map. As the local map is updated the robot is still able to avoid the obstacles safely. Situation E: obstacles present in the off-line generated map but absent in execution time. This explains why the robot's path apparently crosses these obstacles.

The goal was now in the back of the edifice, even at such speed the robot did not collide and avoided pedestrians in a safe controlled way. The only problem in this part of the test was the speed it carried while climbing the sidewalk (see Figure 5.18(a)), this issue could be resolved by inducing cost to climbing the sidewalk, for that a different sensor is needed because 2d laser scanner cannot give the necessary information for that type of algorithm and so it is out of the scope of this thesis. Midway on the route to goal the top speed was reduced and cruised to its next goal (see Figure 5.18(b)).

The subsequent goal was on the other side of the building (see Figure 5.19) the planner made feasible plan very quickly, but with an inflation radius too pessimistic the path sometimes led to robot to move alongside the sidewalk rather than on top of it. At that time a car passed by but with no consequences (see Figure 5.20(a) and Figure 5.20(b)), the robot was constantly approached by people walking and stepping on his way (see Figure 5.20(c) and Figure 5.20(d)). All this was ultimately not to be a problem, the only time the robot stopped was to change locomotion modes to avoid an obstacle that appeared directly in front of it.

When the robot reached the other side of the building it got stuck (see Figure 5.21(a)) when climbing the sidewalk. That was probably caused by its batteries that were almost completely drained. The robot was paused, the batteries exchanged, and the inflation parameter tweaked, that solved the problem. The robot proceeded on its way this time always traveling on the sidewalk rather than alongside it (see Figure 5.21(b)).



(a) (b)

(c) (d)

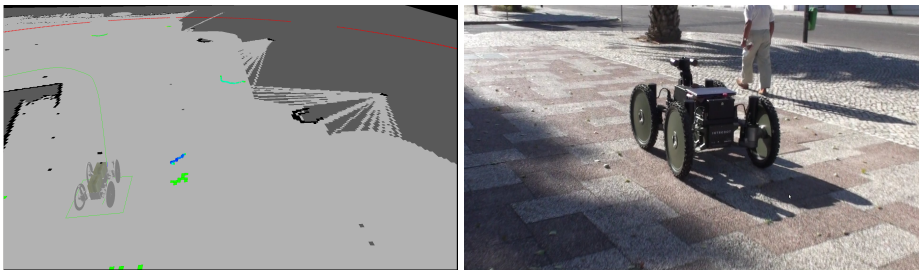
Figure 5.14: Snapshot of the route to the initial goal. (a) Robot initiating the maneuverer. (b) Initial bad localization caused by heading drift. (c) Robot moving through the entrance. (d) Final position. Red/green represent global/local obstacles, RGB marks are the laser scanner hits

The localization estimate given by the human operator on the restart was not accurate enough and that caused a translation error that led the robot to become inaccurately localized in the next sharp turn, even with that error the robot achieved its goal.

The robot had some issues during this field trial, magnetometer drift was in some measure responsible for some of the setbacks felt by the location mechanism, when approaching structures with ferromagnetic materials the heading steadily drifted to align with the magnetic field caused by them. This caused some difficulties in certain parts of the route because of the proximity to the structures and the slow speed of the robot in those locations. One of the ways to diminish this effect was to disconnect the magnetometer function of the EKF filter of the IMU, this would have degraded the estimate of the absolute heading but could have yield better results. Another problem was the difficulty off making an accurate first estimate of the position of the robot, in one case that led to bad localization from then on. This was in part fault of the small spread of initial particles both in translation and in heading, in the following experiments this issue was taken in consideration. Besides the issues addressed above the run had very positives points, the robot travelled without any collision in a safe and controlled manner and was capable of reaching every goal sent to it. The localization was robust when properly initialized regardless of the dynamic environment with passing cars, moving vegetation, and several pedestrians.

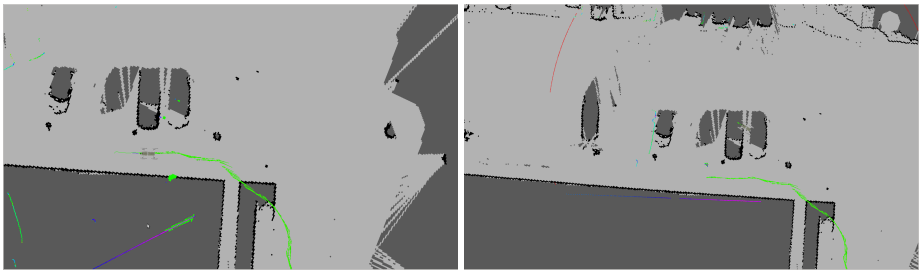
5.2.2 Open Spaces

In the open space operating mode the robot is requested to follow a path generated by the offline path design tool described in chapter 4. The characteristics needed for the experiment were simple,



(a) (b)

Figure 5.15: Snapshot of the robot moving alongside a pedestrian. Red global obstacles, green local obstacles, RGB laser scanner hits

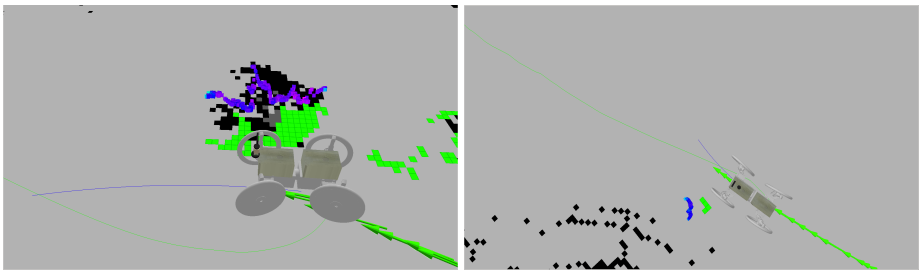


(a) (b)

Figure 5.16: Example of bad localization. (a) Initial bad localization caused by heading drift. (b) Final position. Red/green represent global/local obstacles, RGB marks are the laser scanner hits

clear view of the sky and a large open space was chosen to better mimic a typical off road setting. The area chosen is a rough dirt clearing with small elevations and with about 6300m^2, filled with small concrete objects, large patches of ground and medium height vegetation and also some dismantled steel fences.

The path to the robot to follow was made with the GPS path design tool and passed to the robot, its objective was to simulate a real world application of surveillance or recon in an off road environment. As it is noticeable in Figure 5.24 the path followed by the robot despite highly successful has some differences to the ideal route. The unexpected detours were caused by different factors. One is related to the mismatch between the environment state obtained from the google maps aerial imagery (see Figure 5.24) and the actual one. That had to be anticipated in a dynamic a setting like an off road dirt patch with vegetation growth and in this case man made debris. With this in mind the parameters of the GPS following rules previously explained in chapter 4, where tuned to be distrustful in regard to the surroundings with an goal achieved radius of ten meters, that explains the reason why some of the waypoints were discarded because they resided inside this radius from one another. Another major reason for the path deviation is the existence of false positives manly caused by vegetation and terrain roughness. By not being the objective of this thesis the amount of false positives presented the opportunity of creating challenging situations to the navigation stack, that said the false positive issue will be analysed with greater depth in a next paragraph.



(a) (b)

Figure 5.17: High speed avoidance. (a) Snapshot of high speed avoidance. (b) Snapshot of another high speed avoidance. Red/green represent global/local obstacles, RGB marks are the laser scanner hits

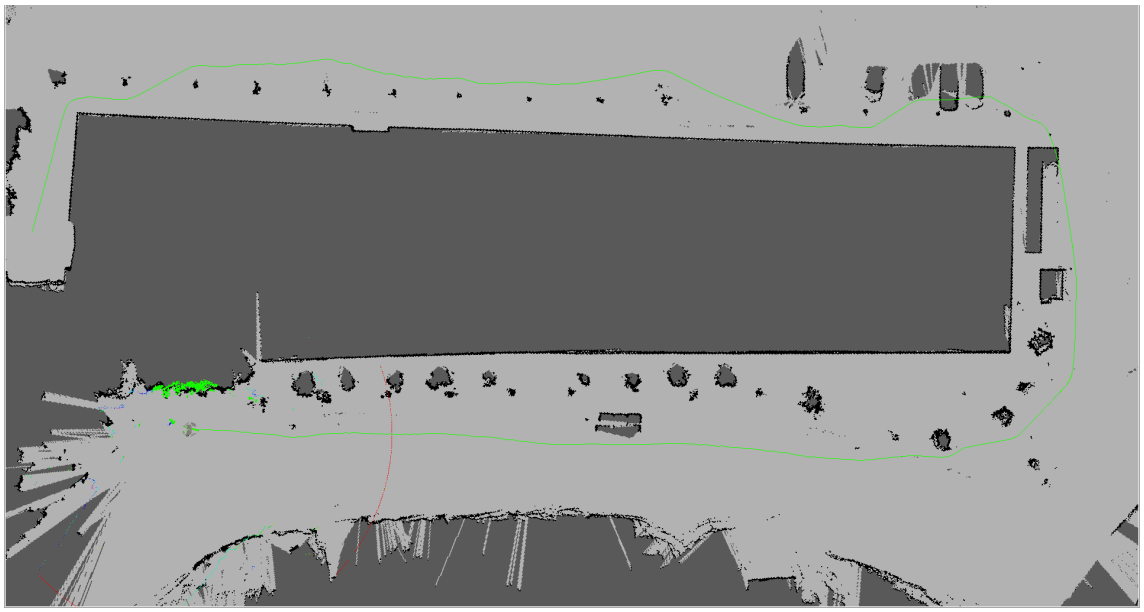
(a) (b)

Figure 5.18: Robot near the third goal. (a) Robot climbing the sidewalk. (b) Robot reaching the third goal. Red/green represent global/local obstacles, RGB marks are the laser scanner hits.

The robot took six minutes and 23 s to complete the 221 m of the course traveling at $0.7\,\mathrm{ms}^{-1}$ average speed while moving, some of the noticeable events are depicted in this subsection. The first waypoints were located in an unobstructed area of the terrain with almost no obstacles only with some low ground hugging plants and terrain irregularity that generated only a small amount of false positives (see Figure 5.25), this enabled the robot to cover this area quickly.

The next waypoints were located on top of a large cluster of vegetation (see Figure 5.26), that was a challenge to cross because even though clearly transversal they were detected as obstacles and because of the non-static nature of the vegetation forced constant replanning and the robot exhibited an oscillating behaviour. After successfully dealing with this problematic zone the robot was forced to avoid a dynamic obstacle that moved extremely close (see Figure 5.28) this made the robot slow down and this part of the route was travelled at $0.3\,\mathrm{ms}^{-1}$ that is substantially slower than the $0.72\,\mathrm{ms}^{-1}$ average of the whole run, this was caused by the sudden approach of the dynamic object and the proximity to the robot, that preferring not to back up had to stop and rotate towards open space. After reaching several goals and discarding some because of the close proximity between them, the robot avoided another dynamic object and then moved towards a more confined region that included a large patch of vegetation and some piled up and dismantled steel fences (see Figure 5.27), finding a gap the robot moved towards it and reached its final destination.

The run was successful only plagued by the vast amount of false positives and IMU drift, all this



(a)

Figure 5.19: The path generated by the global planner is represented by the green plot.

problems could be resolved by adding different sensors that could provide more reliable information or, in some cases, using more advanced perceptual algorithms. Neither of the issues prevented the robot to complete its run and with a good average speed for its parameterization, the max speed allowed to the navigation stack was 1 meter per second that is close to the 0.72 m/s achieved, proving that the solution presented was robust and will reach the objectives proposed to it in most situations.

5.2.3 Transition

For the transition between open space/narrow space the outside of the building of the Electrical Engineering Department was selected. The surrounding environment provided the necessary diversity to be a good testing ground for achieving the experiment objectives. The route included a long straight in front of the Mathematics Department that was the GPS part of the path and after crossing a road the robot entered in the Narrow Space area (see Figure 5.29), finishing below the balcony of the building of the Electrical Engineering Department. The focus where is not the challenges of terrain itself but the transition from one operating mode to the other in a seamless way. In the path design tool there is no information of the semantic area of each point, it is the navigator module that contains a semantic map of the area and distributes the goals accordingly. This test is divided in three different parts the first is the GPS part, second the transition itself and finally the Narrow Space.

In the GPS part the robot was requested to follow waypoints that conducted it to in a straight line alongside the front of the building (Figure 5.32), as in the previously GPS test described above, there was a heading drift caused by the fusion filter of the IMU but as before the robot demonstrated capable of following the waypoints until it reached the transition area. The robot during this stretch of the course encountered dynamic objects but none crossed its way, only slowing down when descend-

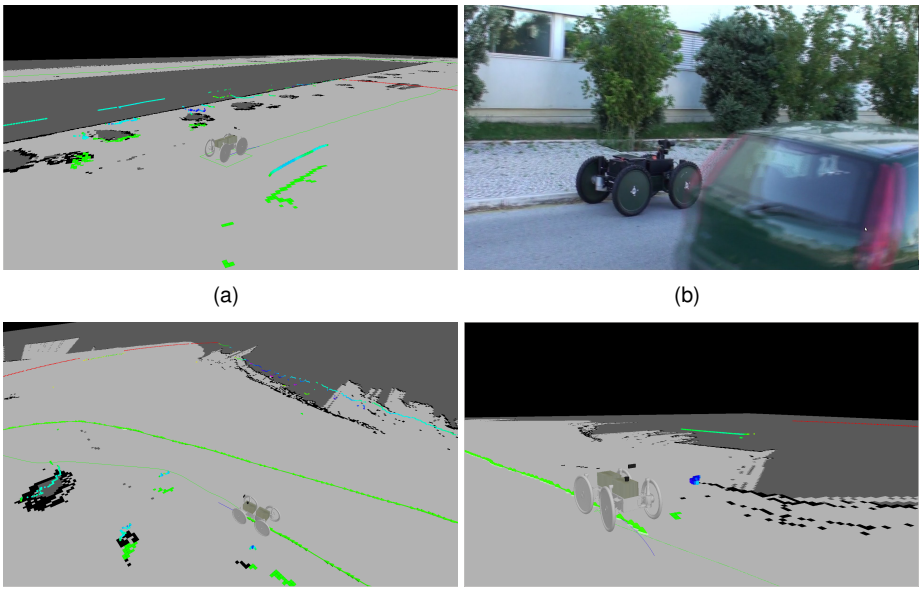


Figure 5.20: Snapshots of events on the way to sixth goal. (a) Marking moving car as an obstacle. (b) Snapshot of car passing by. (c) Robot harassed by pedestrian. (d) Robot avoiding pedestrian. Red/green represent global/local obstacles, RGB marks are the laser scanner hits.

ing from the sidewalk and momentarily detecting the ground and marking it as an obstacle (Figure 5.33(a)).

After crossing the road the robot entered in the Narrow Space Area, here the robot started the transition phase, where it started the localization methods and changed some of the navigation parameters. One of the most important stages of the transition is the first estimate of the robot position (see Figure 5.34(a)) in relation to the offline map. Using the experience gained in the Narrow Space Outdoor experiment, where there were localization problems at certain point, the localization error estimate was enlarged not to encompass only the GPS error estimate but to be a couple of meters larger as well as a bigger array of heading estimates. The Figure 5.35 shows the evolution of the robot localization, showing the scattered initial particles and the filter convergence has the robot moved to the desired goal.

In a couple of meters the robot was properly localized in relation to the offline map, which enabled the robot to move to the next goal, moving close to the ramp on its right the robot slowed down to make the final maneuverer.

The transition experiment was the major objective of the experiments, after the basic function in each operating mode was tested, there was the need to test the capability of the system to adjust to its environment. The robot was capable of changing operating modes based on a semantic map, in this case from an open space to narrow space, in both the behaviour is considered a success the robot never it an obstacle and followed the path to the finish.

(a) (b)

Figure 5.21: Snapshot of the final goal run. (a) Robot stuck from lack of batteries. (b) Robot navigating safely on the sidewalk.

(a) (b)

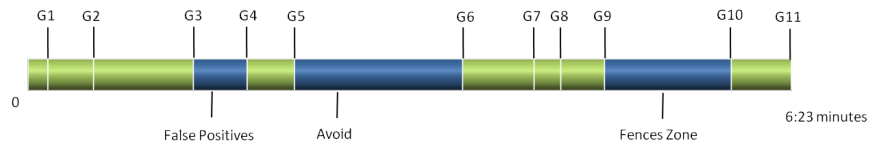
Figure 5.22: GPS location overview

5.3 Discussion

The results obtained throughout the several experiments highlight a set of key factors that must be addressed for a proper autonomous robot navigation system.

With the original rollout implementation the navigation performance was limited in certain situations, mainly due to the system's inability of incorporating different locomotion modes and footprints. Furthermore, when faced with narrow spaces, this limitation lead to the untraversability of some areas. By making the rollout module context aware in what regards the handling of diverse locomotion and footprint modes, it was possible to improve overall robot navigation in the most challenging situations faced across the field trials. That is, the robot never collided even when facing different dynamic objects in the field trials, thus showing the ability of the proposed system to ensure safe navigation. Hence, the new added awareness regarding diverse locomotion modes and footprints seems to attenuate the limitations of the previous navigation system.

The use of a single sensory modality, a laser scanner, poses serious challenges that were evident in the ineffectiveness of detecting some of the cabinets in the narrow space experiment. Still, the system showed the ability to improve navigation as new sensory information enabled a more comprehensive perception over the environment. Moreover, the system has been devised in order to enable an easy integration of additional sensory inputs, such as stereo cameras. This ability is key



(a)

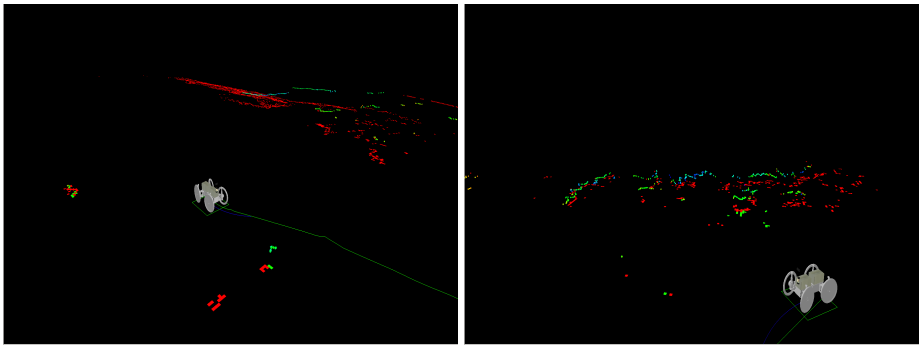
Figure 5.23: A timeline of the transition run where blue represents key moments, red situations where the robot was paused and green for a normal behaviour

to enable the range of domains to which the robot can be applied.

Most of the components used on the Introbot platform are Commercial Off-The-Shelf (COTS), consequently, they entail some constraints at their parameterisation level. These limitations were most evident in the XSENS IMU, which did not permit control over the fusion parameters of its built-in EKF. Without a proper parameterization, a few reliability problems emerged in heading estimation, which impaired to some extent the robot's performance. Remarkably, despite all these issues, the navigation system was capable of taking the robot towards all given waypoints throughout the field experiments, thus showing that the robustness of the aggregate overcomes the limitations of the individual components. Finally, it is important to mention that adding more local semantic information would enrich the robot navigation behaviour. This was particularly evident in the narrow space outdoor test, where the robot would have benefited from being aware of the presence of the sidewalk and the crosswalk. To exploit this added level of semantic information, additional perceptual mechanisms would have to be added. The base system is prepared to receive these add-ons as they become available.

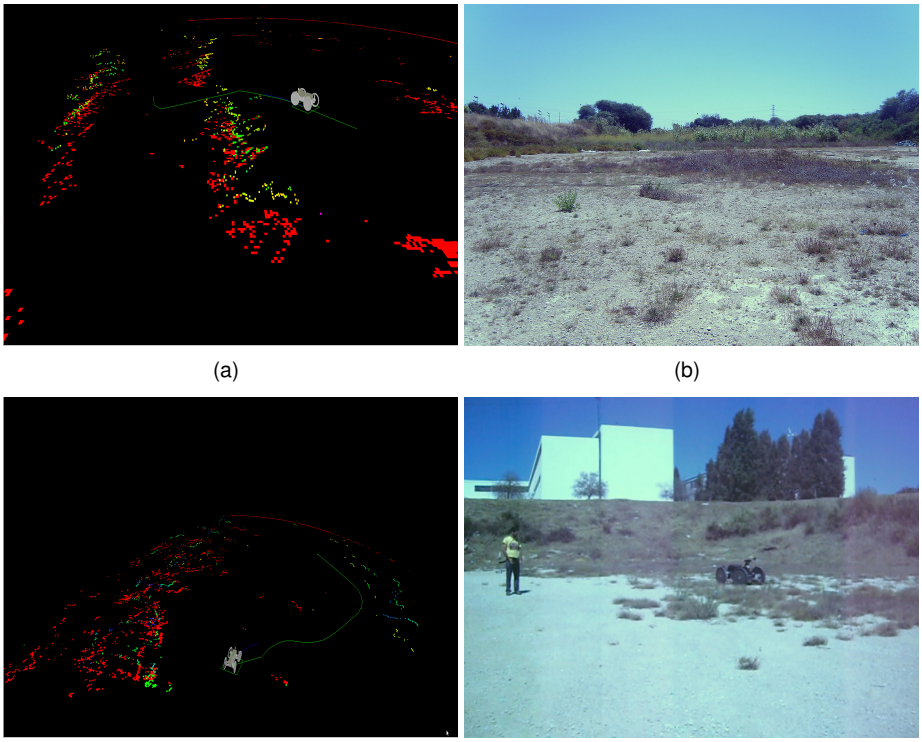
(a)

Figure 5.24: Offline waypoint path (red plot) and actual robot's path (blue plot) executed during the open-space field trial, overlaid on the satellite aerial imagery. Capital letters represent key situations in the course of the test. Situation A: path deviation due to obstacle false positives generated by dense vegetation. Situation B: avoidance of a dynamic obstacle. Situation C: waypoints missed due to the presence of dense vegetation inducing obstacle false positives. Situation D: path deviation due to presence of dismantled steel fences in the desired path.



(a) (b)

Figure 5.25: Snapshots of initial goals. Red/green represent global/local obstacles, RGB marks are the laser scanner hits.



(a) (b)

(c) (d)

Figure 5.26: False positives caused by terrain roughness and vegetation. Red/green represent global/local obstacles, RGB marks are the laser scanner hits.

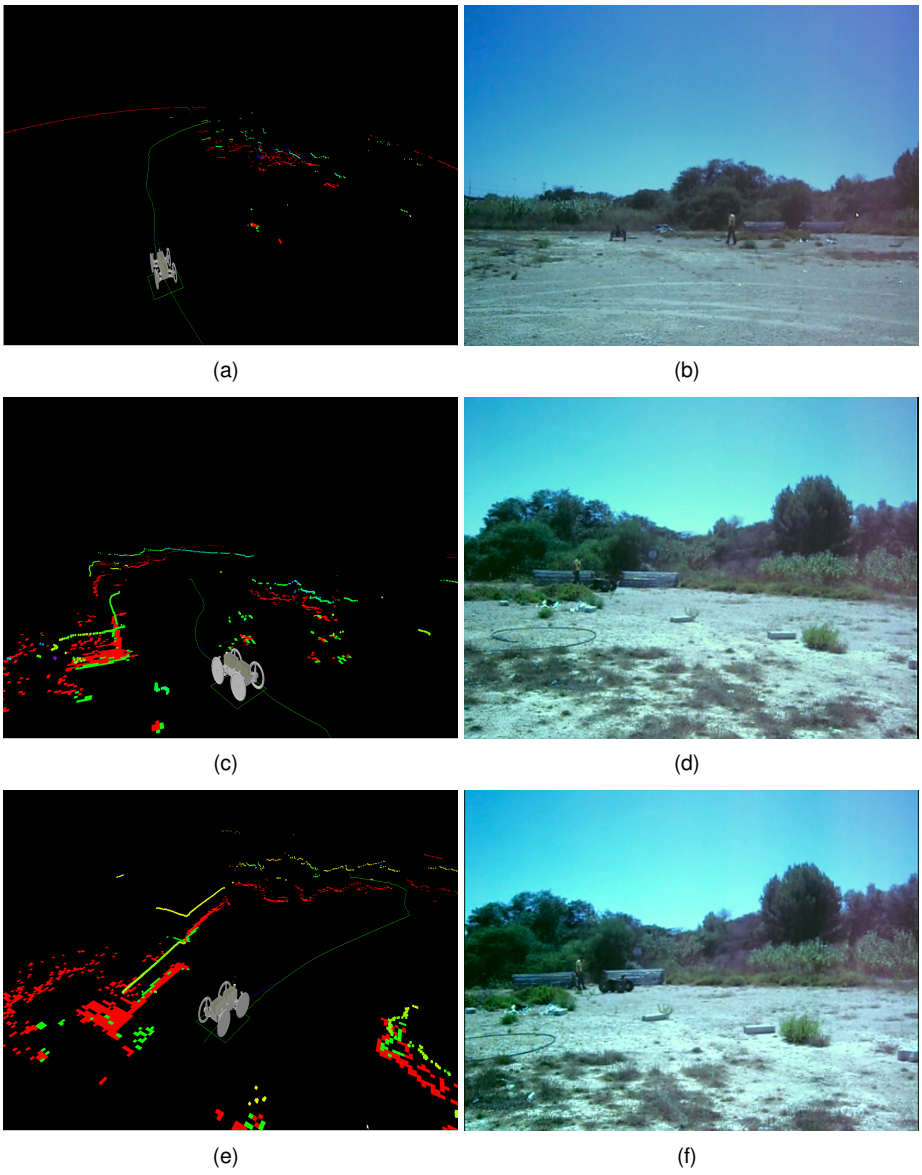


Figure 5.27: Avoidance of a dynamic object. (a) Robot navigating to goal inside steel fences area. (b) Snapshot of robot moving towards the steel fences area. (c) Robot surrounded by obstacle. (d) Snapshot of robot inside steel fences area (e) Robot already planning to another waypoint. (f) Snapshot of robot reappearing Red/green represent global/local obstacles, RGB marks are the laser scanner hits.

(a) (b)

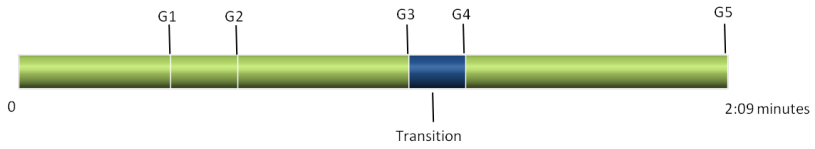
Figure 5.28: Snapshot of avoidance situation where a pedestrian moved extremely close to the robot, making it slow down before safely avoid the obstacle. Red/green represent global/local obstacles, RGB marks are the laser scanner hits.

(a)

Figure 5.29: Semantic map used in the experiment. Where blue represents a narrow space area, green an untraversable area, red a path following area

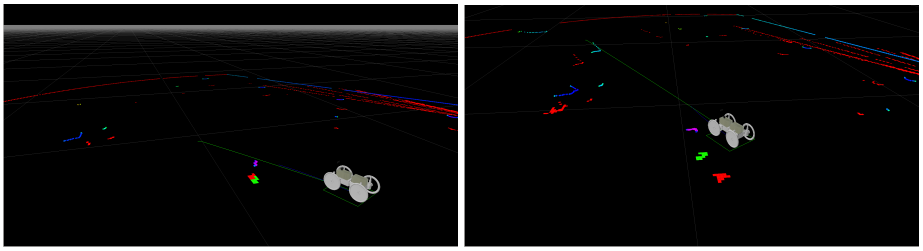
(a)

Figure 5.30: Transition experiment route. (a) Offline waypoint path (red plot) and actual robot's path (blue plot) executed throughout the trial, overlaid on satellite imagery. Letter A represents the robot's transition from open space to narrow space. The blue overlay represents the area of the environment labelled as narrow space. The remainder of the map is labelled as open space.



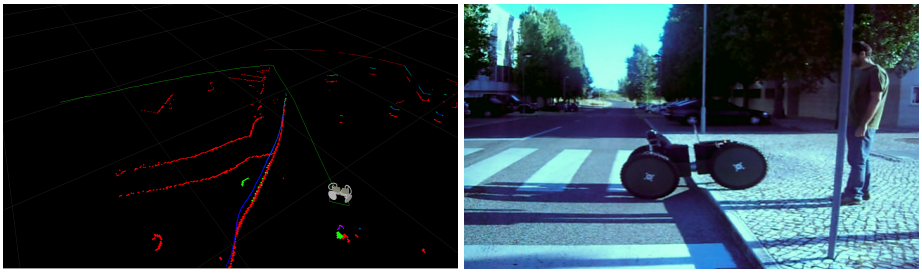
(a)

Figure 5.31: A timeline of the transition run where blue represents key moments, red situations where the robot was paused and green for a normal behaviour



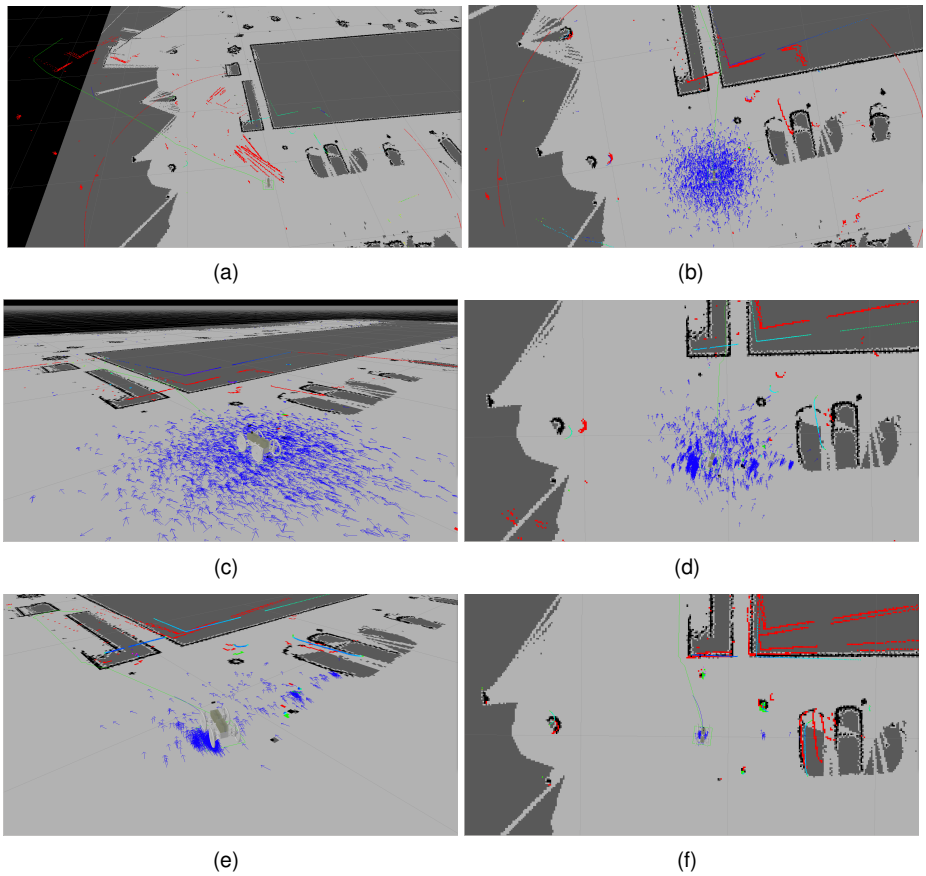
(a) (b)

Figure 5.32: Snapshot of GPS part of the transition run. Red/green represent global/local obstacles, RGB marks are the laser scanner hits.



(a) (b)

Figure 5.33: Robot slows down because of ground detection false positives. (a) Ground marked as an obstacle. (b) Snapshot of the robot position descending the sidewalk. Red/green represent global/local obstacles, RGB marks are the laser scanner hits.

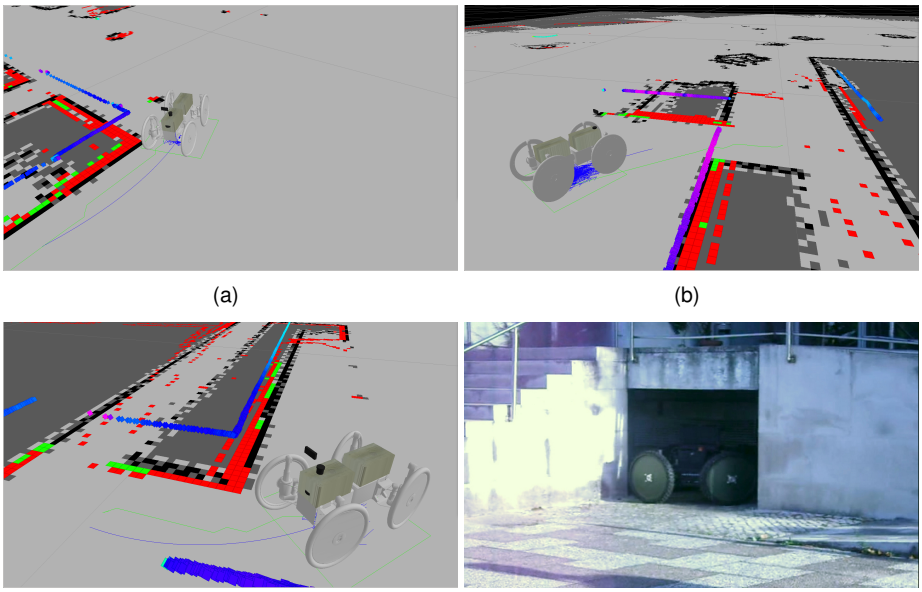


(a) (b)

(c) (d)

(e) (f)

Figure 5.34: Snapshots of particle dispersion evolution in the localization of the robot. Red/green represent global/local obstacles, RGB marks are the laser scanner hits, blue arrows represent AMCL particles in both position and heading



(a) (b)

(c) (d)

Figure 5.35: Snapshots of final maneuverer. Red/green represent global/local obstacles, RGB marks are the laser scanner hits.

Chapter 6

Conclusions and Future Work

This chapter presents a summary of the achieved results in this dissertation, and also gives some directions to potential future research.

6.1 Conclusions

A flexible navigation system designed for autonomous navigation in heterogeneous environments was presented. The navigation system's software architecture is based on the well-known Robotics Operating System (ROS), composed by a group of nodes that were designed to ensure flexibility at all levels of the control system. This flexibility means that the robot can adapt its motion, localization, and planning modules based on local environment semantic information. To help on the validation and development of the proposed navigation system, a simulation setup based on Gazebo was also developed. The simulation setup includes a model of the physical robot used for the experiments results and models of the environments where the field trials took place. The simulation setup is an important part of the system as it permits to surpass the limited amount of time and resources for testing.

Particular care has been given to a typical problem of outdoor navigation, i.e., GPS dropouts. To handle these occurrences, the system picks one of the two following solutions: (1) GPS dropouts occur the system provides two solutions: geo-reference maps created offline for accurate navigation in GPS-denied areas, so that these maps can be correlated with the global navigation goal; (2) use an EKF to compensate GPS with wheel odometry, which is particularly useful for (offline) unmapped open space areas.

Finally, the geo-referenced global localization and paths enable the system with large scale navigation providing flexibility in dealing with unfamiliar terrain.

Results obtained with a robot with four independently steered wheels, on three diverse environments; show that the system is able to provide the robot with safe navigation capabilities in heterogeneous environments. Moreover, the system allows a smooth transition between heterogeneous

environments. The results aforementioned are possible due to the context aware switch and parameterization of the several systems components. To enrich the context awareness of the autonomous robot, high level information was made available to the control system. Concretely, the robot was provided with a priori information via semantic labelling of overhead satellite imagery. This link to semantic information enabled the system to adapt online to each of the evaluated environments, thus enabling seamless transitions. On the other hand, the semantic map contains all the semantic information regarding the environments allowing the geo-referenced waypoints to be set without care to what will be their semantic label.

Most of the limitations observed during the field trials were caused by limitations in the sensory repertoire and not so much on the proposed model itself. Namely, the existence of several false positives in the detection of obstacles in almost all of the field trials leading to a non optimal behaviour and the unreliable estimation of the IMU heading component that impaired the performance of the robot. Nevertheless, these problems did not prevent the navigation system to find out solutions to attain the navigation goals across all field trials.

6.2 Future Work

Though successful, the proposed system can still benefit from additional improvements and extensions, including:

- The implementation of an added level of semantic information similar to work of Collier and Ramirez-Serrano (2009), in which scene classification based on visual cues is used to generalize and predict environments and, in turn, selecting the proper navigation mode, accordingly;
- The parallelization of some of the navigation algorithms, such as trajectory creation and global planners, which would reduce overall computational load;
- The introduction of an odometry source from which motion could be estimated by tracking visual cues between pairs of frames provided by a binocular vision sensor Nistér et al. (2004), would improve the reliability of localization;
- To further increase the robustness of the proposed model's localization capabilities with real-time fusion of GPS/INS information with map-referenced visual features within a particle filter framework (Miller et al., 2011);
- To use EKF-based tracking to detect and follow dynamic objects, which would permit the system to navigate in crowed environments;
- To expand the sensor repertoire with additional laser scanners so that richer information can be extracted from the surroundings and, as a result, some of the problems induced by the presence of obstacles false positives could be mitigated.

Bibliography

Arkin, R. (1989). Motor schema based mobile robot navigation. *The International Journal of Robotics Research*, 8(4):92–112.

Behringer, R., Travis, W., Daily, R., Bevly, D., Kubinger, W., Herzner, W., and Fehlberg, V. (2005). Rascal-an autonomous ground vehicle for desert driving in the darpa grand challenge 2005. In *Proceedings of the 2005 IEEE Intelligent Transportation Systems (ITS)*, pages 644–649. IEEE.

Braid, D., Broggi, A., and Schmiedel, G. (2006). The terramax autonomous vehicle concludes the 2005 darpa grand challenge. In *Proceedings of the 2006 IEEE Intelligent Vehicles Symposium (IV)*, pages 534–539. IEEE.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.

Burgard, W., Derr, A., Fox, D., and Cremers, A. (1998). Integrating global position estimation and position tracking for mobile robots: the dynamic markov localization approach. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 730–735. IEEE.

Burgard, W., Fox, D., Hennig, D., and Schmidt, T. (1996). Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the National Conference on Artificial Intelligence*, pages 896–901.

Chen, Y., Sundareswaran, V., Anderson, C., Broggi, A., Grisleri, P., Porta, P., Zani, P., and Beck, J. (2008). Terramax: Team oshkosh urban robot. *Journal of Field Robotics*, 25(10):841–860.

Collier, J. and Ramirez-Serrano, A. (2009). Environment classification for indoor/outdoor robotic mapping. In *Proceedings of the 2009 Canadian Conference on Computer and Robot Vision (CRV)*, pages 276–283. IEEE.

Crane, C., Armstrong, D., Touchton, R., Galluzzo, T., Solanki, S., Lee, J., Kent, D., Ahmed, M., Montane, R., Ridgeway, S., et al. (2007). Team cimar's navigator: An unmanned ground vehicle for the 2005 darpa grand challenge. *The 2005 DARPA Grand Challenge*, pages 311–347.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57.

Ferguson, D. and Stentz, A. (2007). Field d*: An interpolation-based path planner and replanner. *Robotics Research*, pages 239–253.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *Robotics & Automation Magazine, IEEE*, 4(1):23–33.

Fox, D., Burgard, W., and Thrun, S. (1998). Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3):195–207.

Fox, D., Burgard, W., and Thrun, S. (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427.

Galindo, C., Saffiotti, A., Coradeschi, S., Buschka, P., Fernandez-Madrigal, J., and González, J. (2005). Multi-hierarchical semantic maps for mobile robotics. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2278–2283. IEEE.

Gerkey, B. and Konolige, K. (2008). Planning and control in unstructured terrain. In *Proceedings of 2008 Workshop on Path Planning on Costmaps (ICRA)*.

Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Jazwinski, A. (1970). *Stochastic processes and filtering theory*, volume 63. Academic press.

Julier, S. and Uhlmann, J. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Kalman, R. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

Koenig, S. and Likhachev, M. (2002). Dˆ* lite. In *Proceedings of the National Conference on Artificial Intelligence*, pages 476–483. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Konolige, K., Marder-Eppstein, E., and Marthi, B. (2011). Navigation in hybrid metric-topological maps. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3041–3047. IEEE.

Kuipers, B. and Levitt, T. (1988). Navigation and mapping in large scale space. *AI magazine*, 9(2):25.

Kuipers, B., Modayil, J., Beeson, P., MacMahon, M., and Savelli, F. (2004). Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4845–4851. IEEE.

Leonard, J. and Durrant-Whyte, H. (1991). Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376–382.

Leonard, J. and Durrant-Whyte, H. (1992). *Directed sonar sensing for mobile robot navigation*, volume 448. Kluwer Academic Publishers Dordrecht.

Likhachev, M. and Ferguson, D. (2009). Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945.

Likhachev, M., Gordon, G., and Thrun, S. (2003). ARA*: Anytime A* with provable bounds on sub-optimality. *Advances in Neural Information Processing Systems (NIPS)*, 16.

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., and Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 300–307. IEEE.

Miller, I., Campbell, M., and Huttenlocher, D. (2011). Map-aided localization in sparse global positioning system environments using vision and particle filtering. *Journal of Field Robotics*, 28(5):619–643.

Miller, I., Campbell, M., Huttenlocher, D., Kline, F., Nathan, A., Lupashin, S., Catlin, J., Schimpf, B., Moran, P., Zych, N., et al. (2008). Team cornell's skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics*, 25(8):493–527.

Miller, I., Lupashin, S., Zych, N., Moran, P., Schimpf, B., Nathan, A., and Garcia, E. (2006). Cornell university's 2005 darpa grand challenge entry. *Journal of Field Robotics*, 23(8):625–652.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., et al. (2008). Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al. (2002). Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the National conference on Artificial Intelligence*, pages 593–598. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al. (2003). Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the 2003 International Joint Conference on Artificial Intelligence (IJCAL)*, volume 18, pages 1151–1156. Lawrence Erlbaum Associates LTD.

Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–652. IEEE.

Nourbakhsh, I., Powers, R., and Birchfield, S. (1995). Dervish an office-navigating robot. *AI magazine*, 16(2):53.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. In *Proceedings of the 2009 ICRA Open-Source Software Workshop*.

Ramalingam, G. and Reps, T. (1996). An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, 21(2):267–305.

Remolina, E. and Kuipers, B. (2004). Towards a general theory of topological maps. *Artificial Intelligence*, 152(1):47–104.

Schiele, B. and Crowley, J. (1994). A comparison of position estimation techniques using occupancy grids. *Robotics and autonomous systems*, 12(3):163–171.

Silver, D., Sofman, B., Vandapel, N., Bagnell, J., and Stentz, A. (2006). Experimental analysis of overhead data processing to support long range navigation. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2443–2450. IEEE.

Simmons, R. and Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. In *Proceedings of 1995 International Joint Conference on Artificial Intelligence (IJCAL)*, volume 14, pages 1080–1087. Lawrence Erlbaum Associates LTD.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Haehnel, D., Rosenberg, C., Roy, N., et al. (2000). Probabilistic algorithms and the interactive museum tour-guide robot minerva. *The International Journal of Robotics Research*, 19(11):972–999.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006). Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C., et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466.

Ziegler, J. and Nichols, N. (1942). Optimum settings for automatic controllers. *Transaction. ASME*, 64(11).