Mª Teresa Lorente Cebrián

Towards autonomous robotic systems: seamless localization and trajectory planning in dynamic environments

ISSN 2254-7606

© Universidad de Zaragoza
Servicio de Publicaciones

ISSN 2254-7606

Tesis Doctoral

TOWARDS AUTONOMOUS ROBOTIC SYSTEMS: SEAMLESS LOCALIZATION AND TRAJECTORY PLANNING IN DYNAMIC ENVIRONMENTS

Autor

Mª Teresa Lorente Cebrián

Director/es

MONTANO GELLA, LUIS

UNIVERSIDAD DE ZARAGOZA
Informática e Ingeniería de Sistemas

2018

PhD Thesis

Towards autonomous robotic systems: seamless localization and trajectory planning in dynamic environments

María Teresa Lorente Cebrián
October 2018

Supervisor: Luis Montano Gella

Grupo de Robótica, Percepción y Tiempo Real (RoPeRT)
Instituto de Investigación en Ingeniería de Aragón (I3A)
Departamento de Informática e Ingeniería de Sistemas (DIIS)
Escuela de Ingeniería y Arquitectura (EINA)
Universidad de Zaragoza (UZ)

Project framework

The work of this thesis has been developed within the Robotics, Perception and Real Time Group of the Instituto de Investigación en Ingeniería de Aragón of Universidad de Zaragoza. The framework of the thesis comes established by a scholarship from Vicerrectorado de Política Científica of Universidad de Zaragoza (PIFUZ-2012-B-TEC-002) as well as several research and technological transfer projects, listed next:

- TITAM_ie: Tecnologías Inteligentes para el Transporte Autónomo de Mercancías en Interiores y Exteriores (IDI-20110855), 2011-2013.
  The main objective of this project is to develop technologies for the automated transport of loads in unstructured environments, both indoors and outdoors, with the capacity to plan routes, secure and robust.

- TELOMAN: TEams of robots for LOgistics, MAintenance and eNvironment monitoring (DPI 2012-32100), 2013-2015.
  The project involves deployment and actuation techniques of a multi-robot team. It is necessary to address problems of task planning and allocation, coordinated execution of the navigation, perception of the environment from each of the team members and maintaining communication between all system components – robots, infrastructure, bridges, monitoring equipment, etc. This project addresses new goals and challenges for research and their application in real, large and complex scenarios.

- ROBOCHALLENGE: Navegación y Despliegue de Robots en Entornos Desafiantes (DPI2016-76676-R), 2016-2019.
  This project addresses theoretical and experimental research in the field of intervention and exploration of challenging environments by means of aerial and ground robots. A challenging environment is one where, due to different reasons, current robotic techniques fail or do not work properly and continuously. This failure is due to the characteristics of the environment, such as lack of coverage GNSS (any environment confined), absence of visual or geometric discriminant characteristics (tunnels, pipes), dynamic environments (presence of workers or other vehicles), or large (any real environment). Experimental environments considered are confined and structured (tunnels, galleries), or unstructured (tunnels under construction, underground areas in buildings). Realistic demonstrators are developed to evaluate the applicability of the techniques developed in these environments. The

adaptive integration of the information from the sensors allows the environment understanding for the robot localization and navigation during the whole mission.

- RTART: Robotic Testbed in an ARt and Technology center (H2020-645220-RAWFIE), 2016-2018.
  The aim of the project is to add the required steps for sharing a unique Art and Technology center, ETOPIA, which is available for ground robotic platforms experimentation. Five different scenarios will be available within ETOPIA: the large museum entrance, an exhibition hall, a large gallery and connected corridors, a residential area and an outdoor terrace. Monitoring tools are provided, as well as prior maps and assistance with the experimentation.

Resumen

Evolucionar hacia una sociedad más automatizada y robotizada en la que podamos convivir con sistemas robóticos que desempeñen tareas poco atractivas o peligrosas para el ser humano, supone plantearnos, entre otras cuestiones, qué soluciones existen actualmente y cuáles son las mejoras a incorporar a las mismas. La mayoría de aplicaciones ya desarrolladas son soluciones robustas y adecuadas para el fin que se diseñan. Sin embargo, muchas de las técnicas implantadas podrían funcionar de manera más eficiente o bien adaptarse a otras necesidades. Asimismo, en la mayoría de aplicaciones robóticas adquiere importancia el contexto en el que desempeñan su función. Hay entornos estructurados y fáciles de modelar, mientras que otros apenas presentan características utilizables para obtener información de los mismos.

Esta tesis se centra en dos de las funciones básicas que debe tener cualquier sistema robótico autónomo para desplazarse de forma robusta en cualquier tipo de entorno: la localización y el cálculo de trayectorias seguras. Además, los escenarios en los que se desea poner en práctica la investigación son complejos: un parque industrial con zonas cuyas características de entorno (usualmente geométricas) son utilizadas para que un robot se localice, varían; y entornos altamente ocupados por otros agentes móviles, como el vestíbulo de un teatro, en los que se debe considerar las características dinámicas de los demás para calcular un movimiento que sea seguro tanto para el robot como para los demás agentes.

La información que se puede percibir de los escenarios con ambientes no homogéneos, por ejemplo de interior y exterior, suele ser de características diferentes. Cuando la información que se dispone del entorno proviene de sensores diferentes hay que definir un método que integre las medidas para tener una estimación de la localización del robot en todo momento. El tema de la localización se ha investigado intensamente y existen soluciones robustas en interior y exterior, pero no tanto en zonas mixtas. En las zonas de transición interior-exterior y viceversa es necesario utilizar sensores que funcionan correctamente en ambas zonas, realizando una integración sensorial durante la transición para evitar discontinuidades en la localización o incluso que el robot se pierda. De esta manera la navegación autónoma, dependiente de la correcta localización, funcionará sin discontinuidades ni movimientos bruscos.

En entornos dinámicos es esencial definir una forma de representar la información que refleje su naturaleza cambiante. Por ello, se han definido en la literatura diferentes modelos que representan el dinamismo del entorno, y que permiten desarrollar una pla-

nificación de trayectorias directamente sobre las variables que controlan el movimiento del robot, en nuestro caso, las velocidades angular y lineal para un robot diferencial. Los planificadores de trayectorias y navegadores diseñados para entornos estáticos no funcionan correctamente en escenarios dinámicos, ya que son puramente reactivos. Es necesario tener en cuenta la predicción del movimiento de los obstáculos móviles para planificar trayectorias seguras sin colisión.

Los temas abordados y las contribuciones aportadas en esta tesis son:

- Diseño de un sistema de localización continua en entornos de interior y exterior, poniendo especial interés en la fusión de las medidas obtenidas de diferentes sensores durante las transiciones interior-exterior, aspecto poco abordado en la literatura. De esta manera se obtiene una estimación acotada de la localización durante toda la navegación del robot. Además, la localización se integra con una técnica reactiva de navegación, construyendo un sistema completo de navegación. El sistema integrado se ha evaluado en un escenario real de un parque industrial, para una aplicación logística en la que las transiciones interior-exterior y viceversa suponían un problema fundamental a resolver.

- Definición de un modelo para representar el entorno dinámico del robot, llamado *Dynamic Obstacle Velocity-Time Space* (*DOVTS*). En este modelo aparecen representadas las velocidades permitidas y prohibidas para que el robot evite las colisiones con los obstáculos de alrededor. Este modelo puede ser utilizado por algoritmos de navegación ya existentes, y sirve de base para las nuevas técnicas de navegación desarrolladas en la tesis y explicadas en los siguientes puntos.

- Desarrollo de una técnica de planificación y navegación basada en el modelo *DOVTS*. En este modelo se identifica un conjunto de situaciones relativas entre el robot y los obstáculos. A cada situación se asocia una estrategia de navegación, que considera la seguridad del robot para evitar colisiones, a la vez que intenta minimizar el tiempo al objetivo.

- Implementación de una técnica de planificación y navegación basada en el modelo *DOVTS*, que utiliza explícitamente la información del tiempo para la planificación del movimiento. Se desarrolla un algoritmo *A*-like* que planifica los movimientos de los siguientes instantes, incrementando la maniobrabilidad del robot para la evitación de obstáculos respecto al método del anterior punto, a costa de un mayor tiempo de cómputo. Se analizan las diferencias en el comportamiento global del robot con respecto a la técnica anterior.

Los diferentes aspectos que se han investigado en esta tesis tratan de avanzar en el objetivo de conseguir robots autónomos que puedan adaptarse a nuestra vida cotidiana en escenarios que son típicamente dinámicos de una forma natural y segura.

Abstract

Progressing towards an automated and robotic society where humans could live together with any robotic system deploying tedious or dangerous tasks, entails wondering, among other things, what currently solutions are and what potential improvements are to enhance them. Most applications already present in our lifes are robust and suitable solutions for their purpose. However, many of the implemented techniques could provide a more efficient performance or respond to other needs. Likewise, the context where the robots operate acquires great importance in most robotic applications. Some environments are well structured and easily represented, whereas others barely contain practical characteristics to gain information from them.

This thesis focuses on two of the basic functionalities that any autonomous robotic system should have to move in a robust way: localization and computation of safe trajectories. In addition, the scenarios that motivate the research are complex: an industrial park with zones where the characteristics (usually geometric) used to localize the robot vary; and environments highly occupied by other moving entities, like the hall of a theater, where the dynamic characteristics of the other agents should be considered to compute a safe motion for the robot and the other agents.

Scenarios with heterogeneous zones, like indoors and outdoors, lead to the need of a method that incorporates the perceived measurements from the sensors to estimate the localization of the robot all the time. Localization issue has been extensively studied in the literature and many robust solutions both indoors and outdoors have been proposed, but not in mixed areas. Indoor-outdoor transition areas and vice versa, require the use of sensors that work properly in both areas independently, integrating sensory data during transitions in order to not having discontinuities or even getting the robot lost. Thus, autonomous navigation, reliant on correct localization, can be properly achieved without discontinuities or jerky motions.

In dynamic environments, having a methodology that allows representing the changing nature of the environment results essential. Several models have been proposed in the literature to reflect the dynamism of the environment, and they allow to compute trajectory planning directly from control variables of the robot motion, in our case, the linear and angular velocities for a differential-drive robot. The solutions proposed for static environments lead to unsatisfactory behaviors in dynamic environments because they are purely reactive. Then, it is necessary to consider the prediction of the obstacle motion to plan safe trajectories without collision.

The issues addressed and contributions made in this thesis are:

- Design of a seamless localization system indoors and outdoors, with special focus on the fusing process of measurements perceived from several sensors during indoor-outdoor transitions, an issue slightly considered in the literature. This way, a bounded localization estimation is obtained during navigation. In addition, the localization module is integrated with a reactive navigation technique, thus developing a complete navigation system. The whole system has been evaluated in a real scenario, an industrial park for a logistics application, in which indoor-outdoor transitions and vice versa mean a challenging problem to solve.

- Definition of a model to represent the dynamic environment of the robot, named *Dynamic Obstacle Velocity-Time Space* (*DOVTS*). This model contains the free and forbidden velocities for the robot to avoid collision with the obstacles around. It can be used by any existing navigation algorithm, and underlies the new techniques developed in the thesis which are outlined in the following points.

- Development of a planning and navigation technique based on the *DOVTS* model. A set of relative situations between the robot and the obstacles are identified from the model. A navigation strategy is associated to each situation, considering the robot safety, as well as seeking to minimize the time towards the goal.

- Implementation of a planning and navigation technique based on the *DOVTS* model, using explicitly the time information for planning motions. An *A*-like* algorithm is defined which computes several motions for the next instants, increasing the maneuverability of the robot for collision avoidance with respect to the method stated in the previous point, at the expense of a greater computation time. Differences of the robot navigation behavior with respect to the previous method are also analyzed.

The several aspects studied in this thesis are meant to improve in achieving autonomous robotic systems that could share our daily life in scenarios typically dynamic in a natural and safe way.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Robotics has become, for a long time now, a recognized branch of development both in industry and research, given its many applications already present in our lifes and the prospective ones to come.

We encounter industrial robots as robotic arms, *manipulator* robots, in the assembly lines of several companies. Having one of their ends in a fixed position, these robots appeared as a tool to perform repetitive tasks, such as moving objects from one position to another, blending or painting, with more accuracy and in a shorter time than humans do, thus reducing production costs. Due to the fixed end, the controllability of these robots is high: computation of the final position of the end effector performing the specific task is straightforward [RSS11].

Nonetheless, there are numerous applications where the capacity of the robot to move around results of interest, for example, to transport heavy goods in an industrial plant or to explore a dangerous area after a gas leakage. But the context in which these *mobile* robots become more popular is for serving human needs or demands, such as vacuuming the floor, guiding a group of people in a museum, delivering pills or food to patients in hospitals, or for surveillance. The aim of these *service* robots is to perform tedious tasks for humans. Due to their ability for moving, the position estimation for mobile robots results challenging.

In order a robot moves, it is necessary to estimate its *localization* with respect to the environment around. This computation is performed due to the information obtained from the *proprioceptive* and *exteroceptive* sensors which measure the motion of the wheels as well as the distance to the obstacles, respectively. In addition, these measurements will allow the robot to build a map of the environment, which is mandatory for the localization in structured environments and indoors. However, the type of environment might be unknown or partially known (which limits the controllability of the robot) and diverse: indoors/outdoors, structured/non-structured or static/dynamic. Usually, outdoor environments are hardly structured, thus sensors used in many applications indoors are not practical. In terms of the data available from sensors, non-homogeneous environments are challenging concerning the integration of these data to maintain a seamless localization of the robot while moving. Robots that really must have an autonomous behavior in mixed indoor-outdoor missions, have to seamlessly move between both kind of scenarios without loss of localization within the transition. Within this transition, the sensors have to change in order to use the best ones in each situation. This issue is not extensively addressed in the literature, but it is essential to have a localization without discontinuities. Techniques presented in the first part of the thesis are devoted to solve this problem, named as *seamless localization*.

Optimal techniques for navigating robots in changing, dynamic, and unforeseen scenarios, where autonomous robots have to coexist with other robots and humans in a friendly way, for example in crowded scenarios such as airport or hospital halls, require using planning and navigation strategies different from the techniques used for static scenarios. During the last years many efforts have been dedicated to improve planning and reactive navigation algorithms for static scenarios, some of them also applied to dynamic environments. However, the use of purely reactive techniques together with classical global planners for navigation leads to unexpected, avoidable collisions, and even robot blocking. In addition, in dynamic environments, the free space available for the robot movement changes every sampling time, which leads to a more reduced effective free space for a motion planner. Thus, an explicit consideration of the obstacle's trajectory and its velocity will eventually improve the navigation performance.

The bulk of the work developed in this thesis deals with this topic, the *local motion planning* for *obstacle avoidance* in dynamic environments. An important issue concerns representing the dynamism of the environment in order to make thorough decisions for safe robot navigation. Many approaches use the Workspace, which is the Cartesian space where the state of the robot is described by (x, y, θ). However, this space may contain positions not reachable by a robot given its kinematic and dynamic constraints. Other approaches decide to use the Configuration space, which is defined by all the possible transformations the rigid body of the robot can experiment [LaV06], whereas others use the Control space of the robot, defined by velocities or accelerations. Usually, using the Control space leads to reduce the search space by mapping directly the variables that control the robot motion. After modeling the obstacles, the space will contain free and forbidden velocities for planning.

In addition, it is important to consider the kinematic and dynamic constraints of the robot. The geometry of the robot and each of the wheels impose restrictions that determine the kinematic constraints of the robot, while the dynamic constraints have to do with the actuating forces to achieve the kinematic motion. The maneuverability of the robot is determined by the mobility available based on the kinematic sliding constraints of the wheels, and the contribution by steering the steerable wheels [RSS11]. Many of the robots found in real world are *non-holonomic*, that is to say, with kinematic constraints. This kind of robots are dealt through this thesis.

Usually, trajectory planning comes with the use of a function which balances some criteria to evaluate the different motions for the robot. Several criteria can be chosen, such as generating smooth paths versus jerky paths or optimize either distance, time, entropy or energy consumed. Depending on the aim of the application we will choose one or several criteria to compute the best motion for the robot. Through this thesis, we will focus on minimizing time and distance, without losing sight to the key issue of safety.

1.1 Goals and contributions

Many works regarding planning and navigation have been developed for the last years for static environments. However, in most applications, the environment is dynamic, populated, where other vehicles or people are moving around the robots. If we want to have real autonomous robots in real-world scenarios, we have to develop techniques which take into account this dynamism.

Another relevant issue for the deployment of real autonomous robots, not extensively treated in the literature, is the continuous localization in scenarios involving indoor-outdoor transitions. This fact justifies that, before addressing techniques for planning and navigation in dynamic environments, the problem of seamless localization in indoor-outdoor scenarios and in-between transitions is addressed in Chapter 3. A seamless localization and navigation system is implemented, where the data obtained from different sensors are integrated to obtain an accurate robot localization without discontinuities, especially in the indoor-outdoor transition areas or vice-versa. This part was developed within a technological transfer project, TITAM_ie, in collaboration with another coauthor whose main effort was to develop the navigation issue, while mine was focused towards the outdoor localization problem. All the other issues dealt within the project and the extensive evaluation, both in simulation and experimentation involved both coauthors equally.

The rest of the thesis focuses in the development of new techniques for navigation in dynamic environments. An analysis of the trajectories and velocities of the moving entities around a robot, either robots or vehicles, is performed to determine the collision time which restricts the potential controls for the robot movement. The definition of a proper model which reflects the obstacles in terms of forbidden controls helps in the decision process when searching for commands concerning safety and other criteria to evaluate the system performance. Chapter 4 provides the theoretical definition of the proposed model and its properties, used to map the moving obstacles, and establishing a framework for planning and navigation algorithms. Chapter 5 develops a planning and navigation technique based on the $DOVTS$ model. The algorithm identifies a set of situations for obstacle collision avoidance, associating a specific navigation strategy to each of them, ensuring safety and near-optimal time-to-goal motions. In Chapter 6, an improvement of the previous technique is developed. The time is explicitly used from $DOVTS$ model, which increases the maneuverability of the robot against the moving obstacles. Both methods are compared within Chapter 6.

The contributions and associated publications obtained from the results of the work developed through this thesis are:

- A seamless indoor-outdoor localization and navigation system for car-like robots. The context is a real logistics park scenario with heterogeneous areas, where data from different sensors are integrated to achieve a continuous localization. Some results with regard to localization have been published at the Iberian conference ROBOT13 [ULVM13]. The main results were presented at the European Robotics Forum'15 [ULVM14] and a complete version of the whole system was published in the Journal of Field Robotics [ULVM17].

- The definition of a model for moving obstacles, named *Dynamic Obstacle Velocity-Time Space* (*DOVTS*). The model is defined on the Control space, the Velocity Space, of a differential-drive robot, in which the kinematic constraints of the robot are considered. The formal characterization of this model was published in the International Journal of Robotics Research [LOM18].
- A model-based planning technique and navigation for differential-drive robots, based on the *DOVTS* model. A set of situations is identified from the model of the environment to determine associated strategies of navigation. The results of this approach were published in the International Journal of Robotics Research [LOM18].
- A planning and navigation technique based also on *DOVTS* model, which explicitly takes into account the time for planning, using an *A*-like* algorithm adapted to the *DOVTS* model. The results were presented at the European Conference on Mobile Robots [LM17].

Next, Chapter 2 provides a review of several approaches encountered in the literature related to the topics discussed in the thesis before going into them in depth. Finally, Chapters 7 and 8 summarize the achievements obtained through the several aspects addressed in this thesis and expose some future lines for research, in both English and Spanish respectively.

Chapter 2

State of the art

This chapter presents some works related to the research developed about localization in heterogeneous scenarios, local motion planning and obstacle avoidance in dynamic scenarios, and safety for motion planning.

2.1 Localization in non-homogeneous environments

The deployment of autonomous robotic systems in scenarios with several zones for navigation, like indoors and outdoors, requires the system to manage some difficulties that may arise in such scenarios, such as the integration of the measurements from different sensors and the reliability of the data obtained. In outdoor scenarios, it is common to use the GPS as the main sensor combined with others such as odometry and IMU, as in [GRS99]. The problem of localization in scenarios with indoor and outdoor zones has been extensively managed in mobile devices using WPAN and WLAN and some authors such as [KKK$^+$08] have applied these techniques to robots. However, these approaches rely on an infrastructure that may not be found in all scenarios and the accuracy is not good enough for autonomous robots. Works such as [CRS09] use a learning scheme with cameras that allows indoor and outdoor light characteristics to be differentiated, applying different localization techniques depending on the situation. In [PSA$^+$06], a Kalman Filter (KF) based integration of a laser-based SLAM approach with GPS, IMU and odometry measurements is presented. Nevertheless, using KF with information sources having an unknown correlation may lead to overconfident estimations, as shown in [JU07].

Additionally, the integration of GPS measurements into localization filters are known to be problematic due to artifacts such as variable uncertainty caused by different satellite constellations and multipaths. Thus, as studied by [AP99], [CDPV06] and [MK10], extra GPS signal treatment is required to use it in a Kalman Filter.

To estimate the orientation in unmapped zones or when the GPS does not provide this information directly, a compass can be used. However, the compass is not reliable in environments with variable magnetic fields. Instead, the GPS measurements of position can be used, with some restrictions. In [HWW08], the authors propose a GPS-based method to compute the orientation assuming that between any two consecutive GPS measurements the robot is moving straight forward. However, this assumption may be easily broken as the frequency of the GPS decreases and the speed of the robot increases. In [GES08], the authors compare the measurements obtained from GPS with the measurements from the IMU to decide whether the robot is moving straight or not. This comparison validates the GPS measurements to be used to estimate the orientation of the robot.

Few works have been found that deal with the localization problem during transitions between indoor and outdoor zones. In a preliminary work [ULVM13], a seamless system to localize a robot in heterogeneous scenarios was described. A laser-based technique in mapped zones and GPS measurements in outdoor zones were used to estimate the localization, jointly with odometry/IMU information that is available all the time. The GPS-based orientation method uses, when possible, a weighted mixture of several estimations to increase the robustness of the localization. In the work exposed in the thesis, this technique is improved, increasing the number of estimations computed. A theoretical stability analysis is also performed, providing an observability analysis of the system and establishing the uncertainty limits which assure the convergence of the state estimation.

Robot companies such as Kuka [KUK15] or Clearpath Robotics [Cle15] are proposing autonomous robots for transportation in warehouses with no dedicated infrastructure. In [GN15] the authors propose a vision based localization and navigation method. The approach is infrastructure-free and is implemented on two phases: a teaching phase where the robot acquires knowledge of the environment and builds a topological map, and a navigation phase, where the robot uses the map for localization and planning purposes. But all these approaches are limited to indoor environments, and thus its application is limited compared to the approach presented herein that works seamlessly indoors and outdoors.

This thesis develops a seamless localization technique for indoor and outdoor environments, which is integrated with a reactive navigation method, including maneuvering capabilities. The localization, navigation and maneuvering techniques are integrated in a complete navigation system, which has been exhaustively evaluated in a real scenario of an industrial park.

2.2 Local motion planning in dynamic environments

Motion planning and reactive navigation in static and dynamic environments have been extensively addressed in the literature. Reactive approaches combined with global planning techniques lead to so called iterative motion planners, e.g. ([Fra98], [BK00], [HKLR02], [FDF01], [SB02], and [MM05]). These techniques calculate several steps ahead depending on the time available. The planner evaluates different branches in a tree within a horizon and works out a partial trajectory. This process may be interrupted at any time or is time-bounded to maintain the robot's reactivity. These systems are mainly designed for static environments, which do not provide optimal or good solutions when are directly applied in dynamic scenarios. We will focus in this thesis on the latter kind of scenarios.

Modeling the dynamic environment is a major problem to solve that requires representing the dynamics of the robotic system and the motion evolution of the moving obstacles. Two of the models most commonly used and referenced are the *Velocity Obstacle*, *VO*, [FS98], expanded in [SLS01] for obstacles moving along arbitrary trajectories (*Non-Linear Velocity Obstacle*), and the *Inevitable Collision States*, *ICS*, introduced by [FA03]. In *VO* the set of velocities from the current state that would provoke a collision between a holonomic robot and an obstacle moving in a straight line at a time in the future, is computed. *ICS* computes the robot states which lead to inevitable collision, i.e., there is no control the robot can apply to avoid a collision for a set of *evasive maneuvers. ICS* is directly related to safety.

The *Non-Linear Velocity Obstacle*, *NLVO*, which is the *VO* concept applied to obstacles moving along arbitrary trajectories, is used in [LSSL02] to compute a risk function from the time to collision in order to select the best velocity for the robot. An evolved technique that employs the 3D information obtained by the *NLVO* is presented in [LLS05]. The authors develop an iterative approach based on the $A*$ algorithm to incrementally build a solution towards the goal, where real-time issues are also considered. In [WvdBM09] the authors generalize the *VO* concept by defining the set of controls that result in collision at some time in future, in order to consider the kinematic constraints of the robot. The approach evaluates a set of controls and computes the minimum distance between the robot and every obstacle in the environment, for a certain time horizon. Recently, [KO16] presented a new *VO*-based approach ($OVVO$) which improves the velocity selection compared to the *VO* approach. The method defines a cost function which weighs the similarity to a desired velocity and the level of obstacle avoidance (to move around obstacles to a greater or lesser extent) to decide on the best velocity for the robot in crowded or less dense scenarios. However, the results are very dependent on weight heuristic parameters.

In [PSBF07], [VdBO08], [BTK09], [BVdB13], [vdBGLM11], the authors address multi-robot systems in which all the agents are capable of making avoidance decisions. They developed coordination schemes to distribute the avoidance actions among the robots. [BVdB15] extend this work, generalizing the reciprocal collision avoidance for robots with different kinematic constraints.

A different approach is carried out in [MLP16], where authors deal with motion planning through a B-spline parametrization. The method exploits B-splines properties to replace constraints over the entire time horizon by finite sets of constraints, suitable for real time optimization.

The computation of time-optimal trajectories formulated as a problem of computing time-optimal motions has been explored by several works. [RP94] address this problem for a non-holonomic robot with bounded acceleration, in an obstacle-free environment. The authors demonstrate that time-optimal motions are obtained from bang-bang controls (maximum acceleration or deceleration), and parametrize the set of time-optimal trajectories using the switching times. [RF97] compute sequences of extremal controls for acceleration-driven robots to reach time-optimal trajectories in free-space. In [BM02], the authors focus on differential drive and car-like robots with bounded linear and angular velocities but without acceleration limitations. [SSSS13] presents an on-line motion planner, where the robot dynamics and the actuator constraints are considered. A near time-optimal trajectory which avoids stationary obstacles one at a time is incrementally computed, obtaining comparable results with respect to the global optimal solution. In [PCKL09] the authors deal with one moving obstacle, and solve an optimization problem with inequality constraints for a holonomic robot with infinite acceleration. In [MLP16] the authors formulate the optimization problem in a scenario with moving and static obstacles for a holonomic robot with fixed orientation. Real-time optimization is achieved by a B-spline parametrization of the trajectory and exploiting B-splines properties to limit the set of constraints. A different approach to obtain near time-optimal trajectories is developed in [GSR09] and [SGF10], which adapt time-optimal solutions based on extremal controls for static or free space environments.

2.3 Safety for motion planning

Regarding safety, [LK01] introduced the concept of *region of inevitable collision* to identify the states in which there is no control input the robot can apply to avoid a collision. [FDF01] formulated the concept of τ-safety as a guarantee of no collision during τ seconds for each node of a tree created with a sampling-based algorithm. However, the *ICS* model focuses on safety by computing the robot states which lead to inevitable collision in a time horizon, for a set of *evasive maneuvers*, including the braking maneuver.

In [Fra07], the author introduces three safety criteria that a robotic system should consider in order to compute its future motion: i) its own dynamics, ii) the environment objects' future behavior, and iii) to reason over an infinite time-horizon. Then, *ICS* is presented as a suitable framework to deal with motion safety for motion planning and navigation. [PF05], [KvdP07], [CKZ08] and [BK07] exploit an approximation of *ICS* in different ways. They perform a tree-based search and check for inevitable collision states to provide safety for a time horizon. In [MGF09], the authors present *ICS*-Avoid, a reactive navigation approach which selects controls for the robot to move through *ICS*-free states. Probabilistic versions of *ICS* are defined in [AAWB10] and [BMGF10] to capture the uncertainty about future behavior of moving objects in real-world situations. In [AKWB12] interactions among objects in terms of collision avoidance are considered, reasoning beyond the planning horizon. [BFS12] define *Braking-ICS*, a version of *ICS* which guarantees that if a collision takes place, the robot will be at rest. It is later applied to the development of a partial motion planner ([BFAS14]).

[SGF10] addressed the motion safety issue in the *VO* framework using the *ICS* concept for holonomic robots. Calculating the proper time horizon as the minimum time for the robot velocity to exit the velocity obstacle yields a representation of the environment that ensures robot safety as long as the robot velocity does not lie in the velocity obstacle. In [SP07], the authors adapt the *DWA* approach ([FBT97]) to consider moving obstacles, which are represented as moving cells in a grid map. Safety is thus considered by computing intersection points between simulated obstacle and robot trajectories in the short term. The cells resulting in collision are marked as forbidden, and a command optimizing a weighted navigation criterion is selected for every sampling time.

The *Dynamic Object Velocity-Time Space* (*DOVTS*) defined in this thesis computes safe linear and angular robot velocities (control variables for the robot) explicitly, which can be applied without collision within an horizon limited by the field of view of the sensors on-board the robot. In such a way, building *DOVTS* entails an implicit computation of safe velocities (trajectories) without collision within the space horizon established. This work extends and formalizes the model of a preliminary work, [OM05]. The authors presented the *DOV* object (*Dynamic Object Velocity*) to model the moving obstacles in the environment. The *DOVTS* space herein developed explicitly takes into account the future evolution of the obstacles and the dynamics of the robot to compute a set of velocities leading to collision in the future. *DOVTS* is defined upon the concepts of *estimated arriving time* of the obstacles, used to compute the times to potential collision or escape, and on the *maximum* and *minimum velocities* for the robot not to collide. As a consequence, a set of safe and feasible robot velocities is obtained for further planning.

DOVTS model exhibits several differences with respect to the discussed previous works. *ICS* calculates the states of inevitable collision for a set of escaping maneuvers, but it does not explicitly compute motion controls for planning motions among the obstacles towards the goals. In addition, the computational cost for several maneuvers and several objects is high. The *VO* formulation computes states, including velocity, which lead to collision between a holonomic robot and the obstacles around; usually, planning techniques based on this model compute the next command by evaluating the available commands only for the next period.

This thesis contributes with a new environment model, *DOVTS*, for non-holonomic robots (in our case, differential-drive robots) on the Velocity space. The model represents relative variables (Workspace, velocities) between the robot and the obstacles explicitly or implicitly, which are required for the collision avoidance strategies. The model contains free velocities and collision-leading velocities within some time, which allows to consider safety explicitly. Moreover, as discussed in Section 4.3.6, the model is not affected by some complexity issues that do apply to those previous techniques.

New techniques for planning and navigation are proposed, which exploit the information on the model to implement motion strategies balancing safety (deviation from obstacles) and low times to the goal. One approach is based on the identification of relative situations between the robot and the obstacles from a set of situations, each of which has a different navigation strategy associated. Near-optimal trajectories are obtained, by adapting techniques for optimal motion in free space to scenarios with obstacles. The other approach computes an optimal motion sequence in the Velocity-Time space, by means of an *A*-like* algorithm tailored to the model. A comparison of the methodology proposed in the first technique with respect to another very common in the literature, named in this thesis *one-step* approach, is performed. In addition, the approaches underlying both techniques concerning *DOVTS* are also under comparison. This allows to draw conclusions on the benefits and drawbacks of the proposals in several scenarios with a high number of robots.

Chapter 3

A seamless robot localization and navigation system for indoor and outdoor environments

The work presented in this chapter is developed in the context of a technological transfer project, TITAM_ie, for the deployment of a robotic system in a realistic application. The objective consists in developing techniques that allow an automated robot to move around an industrial plant, with indoor and outdoor scenarios, for the transport of loads. The experimental evaluation to validate the whole system becomes an essential part in the work.

Specifically, this chapter describes a seamless localization and navigation system that permits car-like robots to move safely in heterogeneous scenarios within indoor and outdoor environments. The robot localization integrates different sensor (GPS, odometry, laser rangefinders) information depending on the kind of area (indoors, outdoors and areas between) or on the sensor uncertainty in such a way that there are no discontinuities in the localization and a bounded uncertainty is constantly maintained. Transitions through indoor and outdoor environments are thoroughly considered to assure a smooth change in-between. A navigation technique that combines two well-known obstacle avoidance techniques, the Nearness Diagram and the Dynamic Window approaches, is addressed, incorporating forward-backward maneuvers to manage difficult situations in dense scenarios.

The work presented here was developed in collaboration with another coauthor. In concrete, the work I contributed the most was devoted to the localization method outdoors, whereas the coauthor focused on the navigation part. However, for completeness, this chapter explains the full system developed. The rest of the work and the extensive evaluation both in simulation and experimentation led both coauthors to contribute equally.

The results obtained at different stages of the project were published at the Iberian conference ROBOT'13 [ULVM13], at the European Robotics Forum ERF'15 [ULVM14] and in the Journal of Field Robotics [ULVM17].

3.1 Introduction

The rise of the logistics industry during recent years and the accompanying increase in the volume of goods transported all over the world points to the key importance of warehouse management systems.

One solution adopted by some companies is to fully automate warehouses by adapting their structure to make them *robot-friendly* so that the level of autonomy required for the robot is as low as possible. For instance, warehouses artificially populated with expressive landmarks that facilitate the localization and navigation of robots have been developed. However, these solutions are not always practical when there is no pavement in the warehouse, there are trucks entering the corridors which damage the landmarks, there are changes in the scenario or the goods occupy part of the free space to navigate. In these situations, the cost of deploying and maintaining the landmarks in the environment can be very high. Moreover, these approaches take advantage of highly predictable (static or structured) environments as the limited autonomy of the robots also limits their reaction to unexpected situations.

The solution proposed here is to provide greater autonomy to the robots deployed in warehouses so that they do not require such a structured environment in addition to being able to face unexpected situations that may arise. Thus, they need first to be able to move around the warehouse while continuously maintaining a good estimation of their pose.

Nowadays, robust localization techniques for mobile robots are available with a very mature level of development. These methods, depending on the environment and the characteristics of the robot, rely on different sets of sensors such as sonars or laser range finders, monocular, stereo or omnidirectional cameras, GPS receivers, etc. Due to the variety of the sensors and the different techniques needed to process the data gathered from them, it is common in the research literature to assume that the capabilities of the sensors are available at every point in the operating area. However, in real robotic applications we usually find non homogeneous environments in terms of sensor capabilities. For instance, there are scenarios that combine indoor and outdoor zones, where the conditions for the sensors are very different. Outdoor scenarios are usually less cluttered than indoors and thus range finder sensors may not be able to perceive the structure of the environment. Also, cameras are greatly affected by the light conditions, which can be easily controlled indoors but not outdoors. GPS receivers, however, are useless indoors.

One approach is to use methods that are able to deal with measurements with greatly varying *quality* or even with a temporary lack of measurements from one of the sensors. However, these techniques are not able to achieve the performance levels provided by approaches designed for scenarios with homogeneous specific characteristics. The challenge, then, is to be able to seamlessly integrate the specific techniques so that continuous localization is achieved regardless of the limitations of the particular sensors in given zones.

Vehicles for warehouse applications with big load requirements are large car-like vehicles which have to deal with high accelerations, and even with other moving vehicles, which makes robust navigation difficult and challenging. As a consequence, navigation techniques have to be adapted to all of these characteristics of the environment. In our experience, much of the existing available software sometimes fails when all of these events and characteristics appear in real large-scale environments. The localization and navigation techniques must be robust such that the robot is always localized and that it gets never blocked due to the maneuvers for avoiding obstacles.

3.2 Scenarios, Overview, and Contributions

The framework we are considering is oriented to real applications, mainly to logistic applications. Two scenarios are used to evaluate the techniques developed, one of them being an industrial park, shown in Figure 3.1.

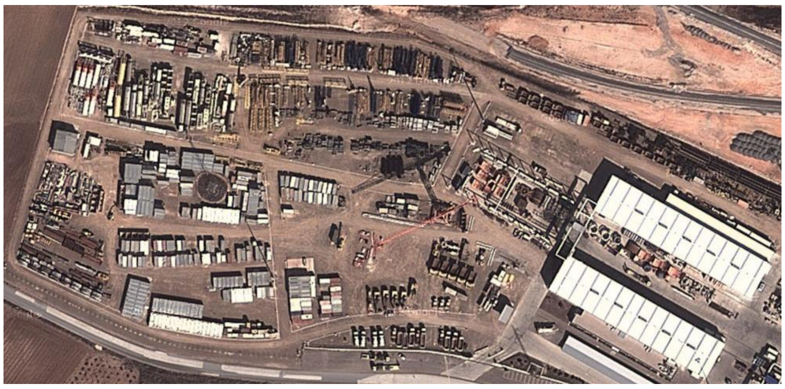


Figure 3.1: Industrial park scenario ($400m \times 250m$) with indoor and outdoor zones where an autonomous robot has been introduced.

The size of this environment, about 100000 m^2, the activity that takes place there and its changing nature do not allow for a fixed and static map to be built for all the scenario, and only some restricted indoor zones can be mapped in advance. Even these zones can change due to new materials being brought to or removed from the warehouse during the normal daily operation. The unmapped parts are outdoors and thus GPS measurements are eventually available in many areas, although with varying degrees of accuracy.

This industrial park is a warehouse for construction material. The pictures in Figure 3.2 show the general aspect of the scenario. Outdoor zones are used for the big pieces and vehicles, they are unpaved and not structured. Inside the warehouse, with dirty concrete floor, there is a fixed structure made out of big shelves. But the goods are not always kept on the shelves but piled on the floor close to the racks. Additionally, in the present scenario, a compass cannot measure the orientation because the Earth's magnetic field is affected by the ferromagnetic structures and stored goods resulting in a useless chaotic magnetic field structure.

The transitions between mapped (usually indoors) and unmapped zones (usually outdoors) take place through known and limited places, typically the doors of the warehouse.

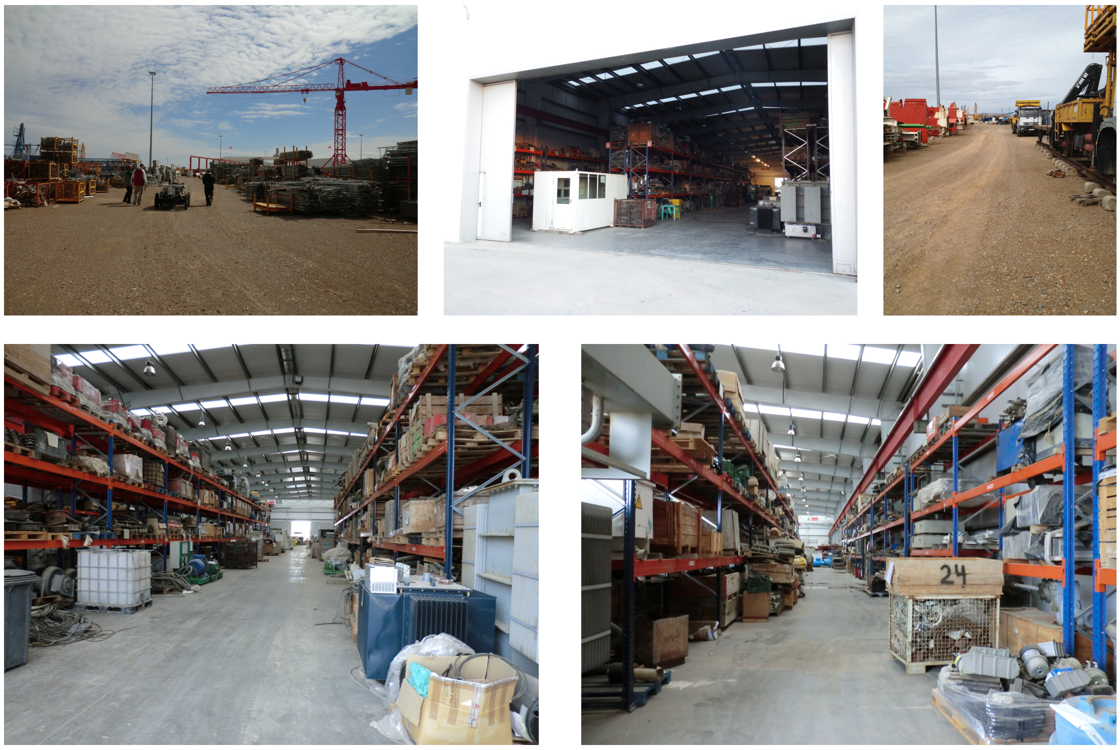


Figure 3.2: The industrial park for management of construction material

In transition zones the sensors are not as reliable as in the zones for which they have been designed.

The robot transports goods all around the scenario, sharing the environment with people walking and other human-driven vehicles. Thus, a high maneuverability in a limited space is also required. The aim of this work is to develop techniques to jointly consider an autonomous continuous localization all over the scenario and a robust, safe navigation between the load and unload points selected by the operator. Load and unload tasks, including the precise approaching maneuver to the shelf are considered independent tasks and thus they are not considered in this work.

The work developed contributes in the following:

- A robust seamless localization method, based on an Extended Kalman Filter (EKF) technique, exploiting the best information gathered from the onboard sensors in hybrid indoor-outdoor environments.

- A special focus on localization during transitions between indoor and outdoor zones.

- A stability analysis for localization, providing theoretical limits and conditions to the localization precision.

- A navigation technique for changing environments, in which not only planning but mainly reactive navigation and maneuvering is needed to comply with unexpected events and scenario changes. The new navigation technique improves the behavior provided by two well-known local planning/reactive techniques, enriching them with automatic maneuvers for cluttered spaces.

- An exhaustive evaluation of the integrated techniques in real-world large scenarios, one of them a currently operating industrial park. The precision of the localization in the experiments, the robustness of the navigation and maneuvering techniques, and the assessment of the theoretical observability results in those scenarios are analyzed and discussed.

Sections 3.3 and 3.4 present the seamless integrated localization technique and the reactive navigation approach, respectively. Section 3.5 is devoted to evaluating the whole system through several simulations and real-world experiments as well as the lessons learned. Section 3.6 sets out the main conclusions of the work.

3.3 Indoor-Outdoor Seamless Localization

In this section we develop a complete and seamless localization technique, which combines information from several sensors, weighting it according to their accuracy. The scenarios we consider are very heterogeneous, composed of mapped and unmapped zones and with variable GPS coverage. In these conditions, we use a framework to provide the robot with a robust and seamless localization, mandatory for achieving a safe and reliable navigation around the whole scenario. The localization framework was proposed in a preliminary work, [ULVM13]. This chapter presents the final approach with the improvements introduced, indicated through the text.

3.3.1 Localization Framework

The localization framework combines the use of independent localization methods for indoor and outdoor zones, while keeping a seamless common localization throughout the scenario with smooth transitions between the different zones.

These independent methods are able to localize the robot in different reference frames, as shown in Figure 3.3. The localization method for indoor mapped zones is able to localize the robot in the MAP reference frame and the GPS-based localization method localizes the robot in the GPS reference frame. To relate the estimations in both methods, it is mandatory to know the relative transformations between the reference frames.

However, it is hard to directly georeference the MAP reference frame into the GPS frame, especially the orientation. Moreover, anytime the map is rebuilt, because of the dynamism of the environment, it would need to be georeferenced again. So, we define a new LOCAL reference frame associated to a feature easily identifiable in the scene. In the particular case of the scenarios used in this work, this feature is located in the middle of the threshold of the door communicating the indoor zones with the outdoor zones. Any other common feature could be used in another scenario. The benefits of selecting this location for the LOCAL reference frame are several:

- The localization of the door in the GPS frame will always remain unchanged.
- The door is easily localized in the MAP frame because it can be found inside the map.
- The door is an important feature of the scenario, as it is on the boundary between the indoor and the outdoor zones.

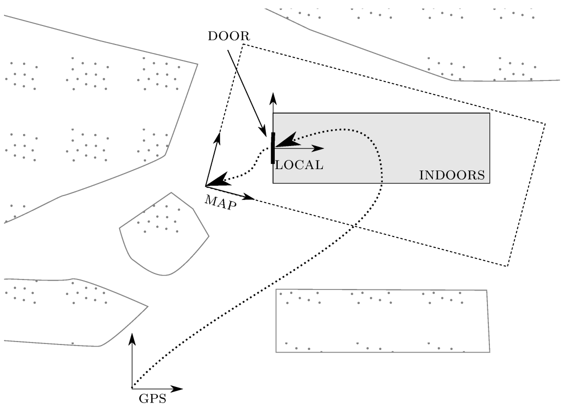


Figure 3.3: The different reference frames used by the outdoor localization (GPS), the indoor localization (MAP) and the LOCAL reference frame created to relate both. For simplicity, the LOCAL reference is defined in the middle of the threshold of the door that connects the indoor and the outdoor zones of the scenario. The pointed arrows are the transformations between the reference frames that need to be computed in order to relate the estimations obtained by the different localization methods.

We are considering a car-like robot that moves on the ground surface. Even though the ground is not completely flat, we can make this assumption as the height variations are negligible and the extra dimension do not provide any useful information to the system. Thus, the localization of the robot in the LOCAL reference frame is fully defined with its position and orientation in the plane $\mathbf{x} = (x, y, \theta) \in \mathcal{R}^2 \times \mathcal{S}^1$.

Localization of the Transition Feature in the Map

The transition feature associated to the LOCAL reference is a door that can be identified in the scenario. To compute the transformation between the MAP and the LOCAL reference frames, either the map origin must be localized with respect to the door or the door must be localized into the map.

The second approach is better as the door can be recognized in the map. By localizing the two jambs of the door in the map (**A** and **B** in Figure 3.4), the computation of the

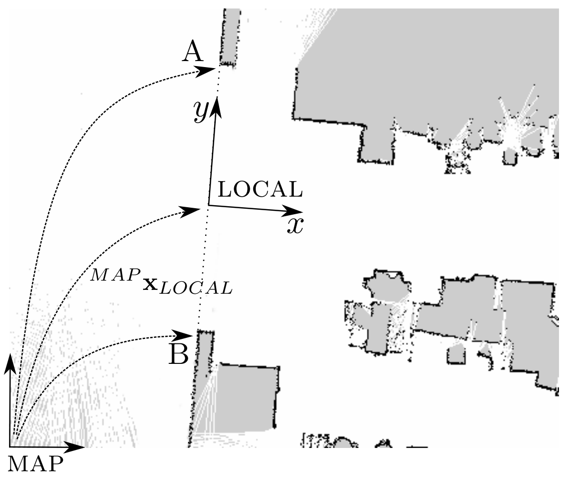


Figure 3.4: The transformation between MAP and LOCAL frames $^{MAP}\mathbf{x}_{LOCAL}$ is computed from the position of the jambs of the door **A** and **B** in the map.

LOCAL reference frame pose with respect to the map $^{MAP}\mathbf{x}_{LOCAL}$ is straightforward using equation (3.1), $\widehat{AB}$ being the angle of the vector **AB** in the MAP frame.

The LOCAL reference frame pose is then set to be in the middle of the door and with an orientation perpendicular to it.

$$^{MAP}\mathbf{x}_{LOCAL} = \left(\frac{A_x + B_x}{2}, \frac{A_y + B_y}{2}, \widehat{AB} + \frac{\pi}{2} \right)^T \tag{3.1}$$

This process of localizing the door in the map is needed every time the map is rebuilt, depending on the dynamism of the elements in the indoor zone.

Geolocalization of the Door

To obtain the transformation between the GPS and the LOCAL reference frames, the door must be geolocalized. Any method is acceptable, including topographic techniques, because the measurement is only performed once. Here, we propose a method that uses only GPS and a laser rangefinder to estimate the pose of the door in the GPS frame. As the GPS measurements do not directly provide information about the orientation, we observe the door, points **A** and **B** in Figure 3.5, with the rangefinder from two different positions. From this pair of observations of the door, the rotation of the sensor at both positions $\mathbf{R}_1$ and $\mathbf{R}_2$ can be computed from the GPS positions $\mathbf{x}_1$ and $\mathbf{x}_2$ and the observations $\mathbf{A}_i$ and $\mathbf{B}_i$ using equations (3.2) and (3.3) and thus, the position of the door can be computed in the GPS reference frame.

$$\mathbf{A} = \mathbf{R}_1\mathbf{A}_1 + \mathbf{x}_1 = \mathbf{R}_2\mathbf{A}_2 + \mathbf{x}_2 \tag{3.2}$$

$$\mathbf{B} = \mathbf{R}_1\mathbf{B}_1 + \mathbf{x}_1 = \mathbf{R}_2\mathbf{B}_2 + \mathbf{x}_2 \tag{3.3}$$

The LOCAL reference frame pose is then set to be in the middle of the door and with an orientation perpendicular to it,

$$^{GPS}\mathbf{x}_{LOCAL} = \left(\frac{A_x + B_x}{2}, \frac{A_y + B_y}{2}, \widehat{AB} + \frac{\pi}{2}\right)^T \tag{3.4}$$

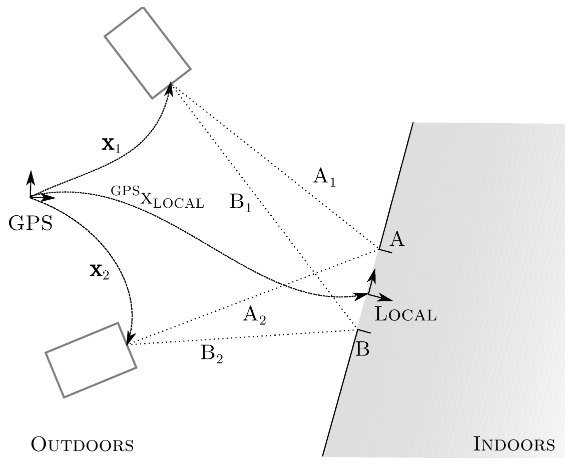


Figure 3.5: Method to find the GPS position and orientation of the door and thus the LOCAL reference frame $^{GPS}\mathbf{x}_{LOCAL}$. $\mathbf{x}_1$ and $\mathbf{x}_2$ are the GPS measurements of the positions of the sensor and $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{B}_1$ and $\mathbf{B}_2$ are the rangefinder measurements of the jambs of the door $\mathbf{A}$ and $\mathbf{B}$.

Localization Framework Structure

As we have mentioned above, the localization framework we propose combines two different localization techniques, outdoors and indoors, and explicitly manages the transitions between them to guarantee a smooth change.

Figure 3.6 shows the state machine that manages the transitions among the different localization techniques. There are different events that produce changes in the state of the localization system. When the robot crosses the threshold of the door, the transitions IN–OUT or OUT–IN, depending on the direction of the crossing, start. This door crossing detection is computed based on the current localization of the robot. We use the GPS quality to trigger the events to leave the transition states. During an IN–OUT TRANSITION, the robot is considered completely OUTDOORS when a sufficient GOOD GPS signal is received. On the other hand, an OUT–IN TRANSITION finishes when there is a BAD GPS event. The special event DOOR DETECTED is used to initialize the indoor localization method before the robot starts the OUT–IN transition. Using this initialization, the localization during the transitions is improved because both indoor and outdoor estimations are coherent at the beginning of the transition.

In the next sections we provide the details of the localization technique used within each state of the state machine.

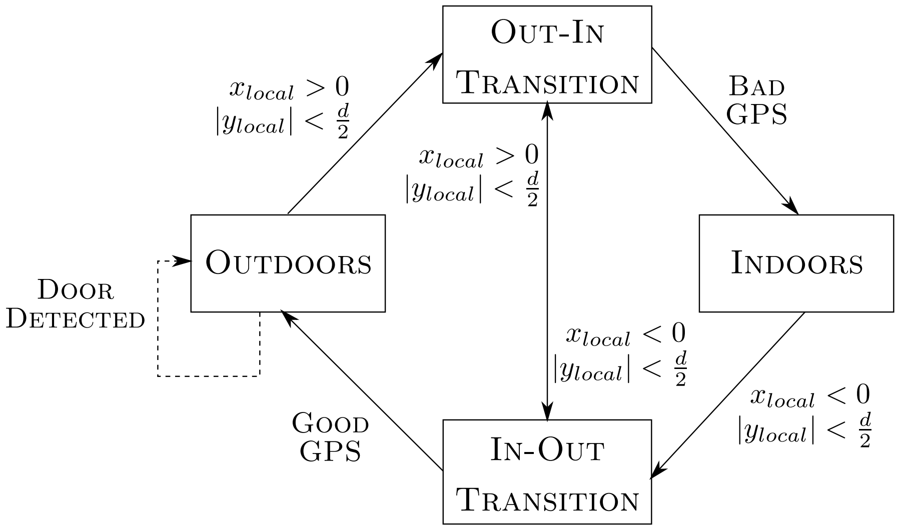

Out-In Transition
Outdoors
Indoors
In-Out Transition
$x_{local} > 0$ $|y_{local}| < \frac{d}{2}$
Bad GPS
$x_{local} > 0$ $|y_{local}| < \frac{d}{2}$
Door Detected
$x_{local} < 0$ $|y_{local}| < \frac{d}{2}$
$x_{local} < 0$ $|y_{local}| < \frac{d}{2}$
Good GPS

Figure 3.6: State machine of the possible situations where the robot may be found and the events that trigger the changes of state. (x_{local}, y_{local}) is the estimation of the position of the robot in the LOCAL reference frame and d is the width of the door (see Figure 3.4).

3.3.2 Localization during Transitions

To track the best estimation of the robot localization $\mathbf{x}$, an adaptation of the EKF is used. The prediction phase is computed from the odometry/IMU data $\mathbf{y}_{od}$, as shown in equations (3.5–3.7), where the relative movement $\mathbf{u}_{od}$ with respect to a previous odometry/IMU measurement is used. $\mathbf{Q}$ represents the covariance of the odometry/IMU error, $\mathbf{F}$ is the Jacobian of the prediction function (3.5) and $'$ denotes matrix transposition.

$$\mathbf{x}(k+1|k) = \mathbf{x}(k|k) \oplus \mathbf{u}_{od}(k+1) \tag{3.5}$$

$$\mathbf{u}_{od}(k+1) = \mathbf{y}_{od}(k+1) \ominus \mathbf{y}_{od}(k) \tag{3.6}$$

$$\mathbf{P}(k+1|k) = \mathbf{F}(k)\mathbf{P}(k|k)\mathbf{F}(k)' + \mathbf{Q}(k+1) \tag{3.7}$$

The operator $\oplus$ (and $\ominus$) are the composition of one 2D transformation with another 2D transformation (inverse in case of $\ominus$) as defined in (3.8) and (3.9).

$$\mathbf{x}_1 \oplus \mathbf{x}_2 = \begin{pmatrix} x_1 \\ y_1 \\ \theta_1 \end{pmatrix} \oplus \begin{pmatrix} x_2 \\ y_2 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} x_2 \cos\theta_1 - y_2 \sin\theta_1 + x_1 \\ x_2 \sin\theta_1 + y_2 \cos\theta_1 + y_1 \\ \theta_1 + \theta_2 \end{pmatrix} \tag{3.8}$$

$$\mathbf{x}_1 \ominus \mathbf{x}_2 = \begin{pmatrix} x_1 \\ y_1 \\ \theta_1 \end{pmatrix} \ominus \begin{pmatrix} x_2 \\ y_2 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} -x_2 \cos(\theta_1 - \theta_2) + y_2 \sin(\theta_1 - \theta_2) + x_1 \\ -x_2 \sin(\theta_1 - \theta_2) - y_2 \cos(\theta_1 - \theta_2) + y_1 \\ \theta_1 - \theta_2 \end{pmatrix} \tag{3.9}$$

The update phase is modified to introduce a covariance intersection technique [JU07] which permits the fusion of different sources of information with unknown correlation among them. Essentially, this technique introduces an extra weighting factor γ for the different measurements that are being mixed (see Table 3.1). This is one of the strategies used in [CDPV06] to deal with the GPS estimations with poor uncertainty estimations, which are known to cause problems when they are directly introduced to a Kalman filter.

In order to evaluate the performance of the filter in such situations, we have tested it with different values of the weighting factor γ for the GPS, γ_{gps}. Figure 3.7 shows the deterioration of the trajectories when the value of γ_{gps} increases and there are multipath problems. Concretely, when γ_{gps} is 1, i.e. the filter is using the uncertainty provided by the GPS directly, the trajectory obtained is erroneous (cyan in figure 3.7) due to the unreliable uncertainty of the measurements given by the GPS. Thus, the need for an extra parameter to weight the measurement and uncertainty becomes essential for a proper performance of the filter, as shown with simulations in [CDPV06].

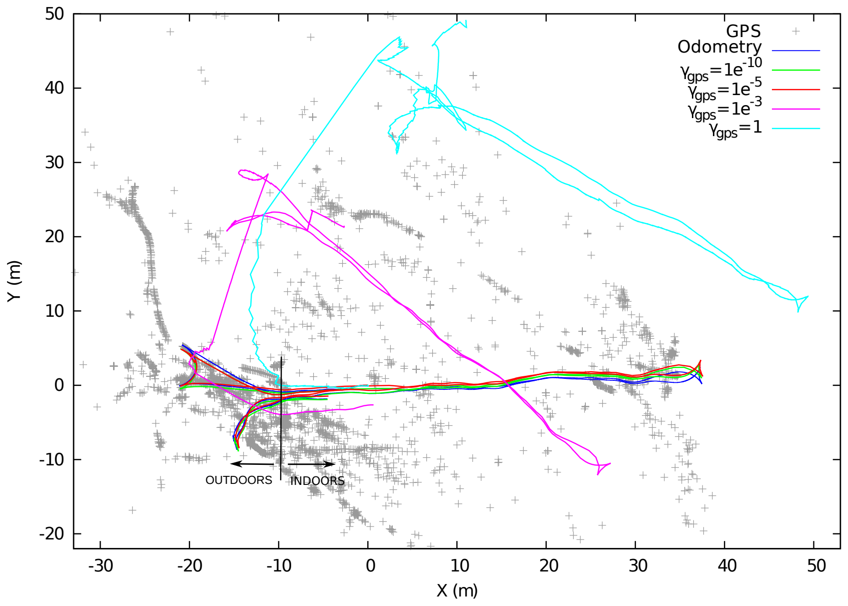


Figure 3.7: Trajectories computed by the EKF with different values of γ_{gps}. Grey marks correspond to raw GPS measurements. The black line differentiates the indoor and outdoor zones in the environment. Trajectory drawn in cyan is obtained from considering directly the uncertainty provided by the GPS ($\gamma_{gps} = 1$), which is not always reliable, and thus leads to an incorrect trajectory.

$$\mathbf{P}(k{+}1|k{+}1)^{-1} = \mathbf{P}^{-1}(k{+}1|k) + \gamma_{map}\mathbf{R}_{map}^{-1} + \gamma_{gps}\mathbf{R}_{gps}^{-1} \tag{3.10}$$

$$\mathbf{x}(k{+}1|k{+}1) = \mathbf{P}(k{+}1|k{+}1)\left(\mathbf{P}^{-1}(k{+}1|k)\mathbf{x}(k{+}1|k) + \gamma_{map}\mathbf{R}_{map}^{-1}\mathbf{y}_{map} + \gamma_{gps}\mathbf{R}_{gps}^{-1}\mathbf{y}_{gps}\right) \tag{3.11}$$

In this case, three different sources are used: the odometry-based prediction $\mathbf{x}(k{+}1|k)$, the GPS estimation $\mathbf{y}_{gps}$ and the estimation of a laserscan-based localization method

Table 3.1: Update weights in different situations. The variable p_{gps} computed in (3.12) lets a smooth change in estimation between MAP and GPS measurements during transitions. The weights indoors and outdoors are set so that the localization estimation is equivalent to a standard EKF.

Situation	γ_{map}	γ_{gps}
Indoors	1.0	0.0
In-Out transition	$1 - p_{gps}$	p_{gps}
Out-In transition	$1 - p_{gps}$	p_{gps}
Outdoors (good GPS)	0.0	1.0

$\mathbf{y}_{map}$ defined in [Fox03] for mapped zones. The final estimation is computed by means of a weighted sum of all the sources, as shown in equations (3.10, 3.11).

For weighting, besides using the covariance matrices of the estimations ($\mathbf{R}_{gps}$ and $\mathbf{R}_{map}$ respectively), two factors (γ_{gps} and γ_{map}) are applied depending on the situation of the robot at each moment, as defined in Table 3.1. A fixed weight was used in the preliminary approach in [ULVM13, Table 2]. Now we improve the technique by introducing a varying weight method. During transitions, the value of γ_{gps} is given by a function which depends on the quality of the GPS measurements, following the *log odds ratio* [TBF05, Chapter 4.2] formulation in equations (3.12, 3.13), which is simpler formulation for binary Bayes filters.

The parameter p_{gps} in Table 3.1 represents the probability of having a reliable GPS, as shown in Figure 3.8. It is computed as,

$$p_{gps} = \frac{e^{l_k}}{1 + e^{l_k}}, \quad l_k = l_{k-1} + L_{gps_k} + l_0 \tag{3.12}$$

where L_{gps_k} is,

$$L_{gps_k} = \begin{cases} 1 & \text{if } Q_{gps_k} >= \xi_{gps} \\ -0.5 & \text{if } Q_{gps_k} < \xi_{gps} \\ -1 & \text{if there is loss of GPS} \end{cases}, \quad l_0 = \begin{cases} L_{max} & \text{if } \textsc{Outdoors} \text{ (Figure 3.6)} \\ L_{min} & \text{if } \textsc{Indoors} \text{ (Figure 3.6)} \end{cases} \tag{3.13}$$

and the signal quality is computed as,

$$Q_{gps_k} = -log(\sigma^2_{lat_k} \sigma^2_{lon_k}) \tag{3.14}$$

The quality of a GPS measurement is taken from [ULVM13], where the threshold ξ_{gps} to distinguish between a good or a bad measurement is set to 4 to the light of the results obtained in Figure 3.9. The tuning parameters in this schema are L_{gps_k}, L_{min} and L_{max}. The relation between these values determine how many good quality GPS measurements are needed to determine that the GPS is reliable and how many bad quality GPS measurements or how long since the last GPS measurement it takes to assume that the GPS is neglected.

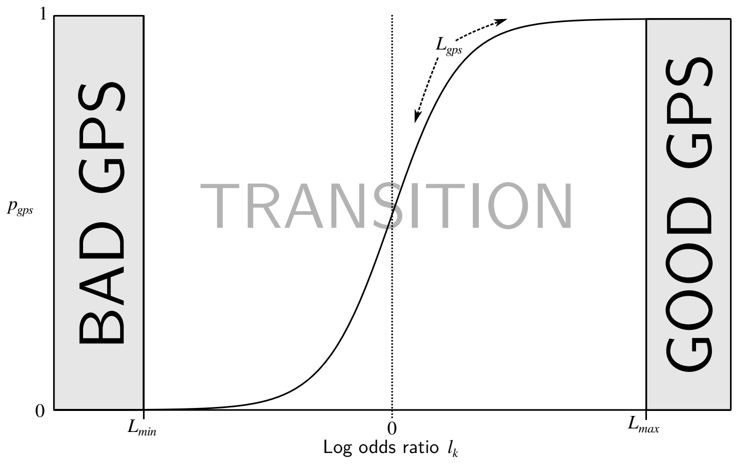
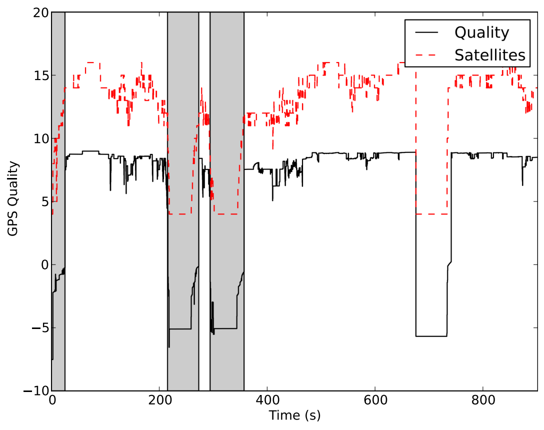
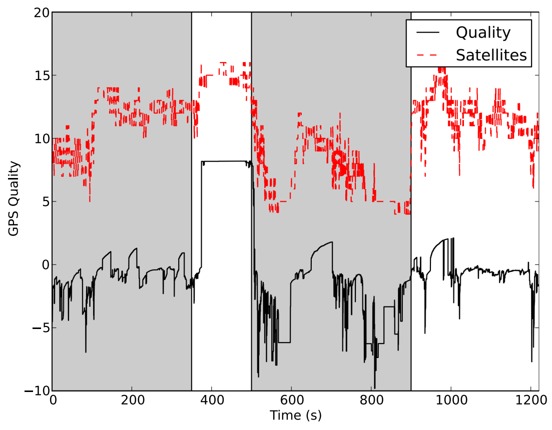


Figure 3.8: Probability of having a reliable GPS. The graph evolves with every update of the log odds ratio l_k which is displaced by the L_{gps_k} value that depends on the quality of the GPS measurement. When l_k reaches L_{min} or L_{max}, the reliability of the GPS is defined and the transition finishes.



(a) GPS measurements quality at *Unizar* scenario.



(b) GPS measurements quality at *Park* scenario.

Figure 3.9: These figures show the number of satellites and the quality of the GPS measurements (from eq. 3.14) in different indoor (gray) and outdoor environments. Note that because of the multipath phenomena in the *Park* scenario, there are many satellites observations indoors. Thus, the number of satellites is not a good estimator for the GPS quality. The low quality GPS measurements outdoors in the *Park* scenario were forced by keeping the GPS close to the building.

As we said in the previous section, the events to leave the transition states in Figure 3.6 are triggered by the reliability of the GPS. Specifically, when the robot is in a IN–OUT transition, if the log odd ratio l_k reaches the maximum value L_{max} the GOOD

GPS is triggered. Conversely, during a OUT–IN transition, if l_k reaches the value L_{min}, the event BAD GPS is triggered.

3.3.3 Indoor Localization

Any well known map-based method can be used for the localization of the robot inside the warehouse. Moreover, our approach of using different localization techniques for outdoors and indoors relies on the fact that robust solutions can be found in the literature for map-based localization methods. However, they generally lose their good performance when they are extended to wider and less structured zones such as outdoor environments.

We have chosen the Monte Carlo based localization method proposed by [Fox03]. This is a particle filter that maintains a continuous localization of the robot by matching the laser scan measurements with a previous grid map, shown in Figure 3.10.

The map is built using the *gmapping* method proposed by [GSB07] with the odometry and laser scan data recorded during a teleoperated trajectory. Maps are only built for indoor zones as these zones are partially structured and thus the usability of these maps can be extended over time, with little changes in the environment that do not compromise the quality of the localization. When the changes in the environment are big enough to worsen up the localization estimation to unacceptable levels, the map is rebuilt again.

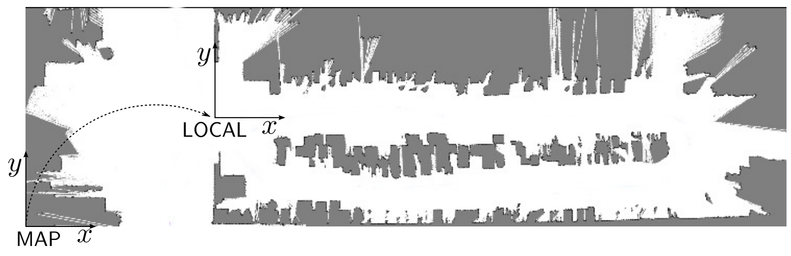


Figure 3.10: The grid map used for indoor localization. The cell size is 0.05 m. Note that there is a mapped zone outdoors ($x < 0$ in LOCAL reference frame) to facilitate transitions.

We have adapted this solution to the scenario by constraining the resample method used for replacing low-weighted particles. Instead of using random particles sampled all over the map, we limit the sampling space to the corridor the robot is in, as estimated by the last output of the particle filter. The rationale behind this is that most corridors have very heavy symmetries and it is easy to fit particles in different positions along different corridors so that multiple hypotheses are considered in the particle filter, which may lead to jumps in the estimation. Moreover, in such a scenario the robot cannot change suddenly to another corridor, or outdoors if it is far from the transition location. These a priori pieces of information are used as heuristics to tune the localization technique, avoiding robot losses.

From the particle filter, by computing the mean and covariance of the particles we obtain the values $\mathbf{y}_{map}$ and $\mathbf{R}_{map}$ for the update phase during transitions in equations (3.10) and (3.11).

Initialization

Indoor localization methods need an initial value for the estimation as they do not use any absolute measurements of the position of the robot. When the localization starts indoors, the method defines an initial MAP reference, which is used as an absolute localization. This map and its absolute reference must first be built. When the robot detects an Out-In Transition, the indoor localization method must be initialized. An indoor localization estimation is needed for the transition localization. The best estimation of the outdoor localization method is used as the initial value for the indoor localization filter, so that particles are randomly distributed around this pose.

This initialization method has the drawback that, if the transition state is very short, i.e. the GPS becomes bad very fast, then the particles may not have converged to the correct robot position before the Indoor localization state has started. This situation may cause a bad initialization of the particles and thus the robot might get lost, especially if the robot performs complicated maneuvers as soon as it enters the warehouse. To minimize these situations, we develop an earlier initialization of the indoor localization method so that, whenever the estimation of the robot with respect to the map is needed, the particles will already have converged to the robot pose.

Door Detection

We use the detection of the door as the trigger to initialize the indoor localization method. When the robot is outside the warehouse and detects the door, the particles of the indoor localization technique are initialized around the current estimation of the position of the robot (see Figure 3.6).

Based on the aspect of the door in the map (see Figure 3.10), we use a segment based model. Thus, the door is modeled as two colinear segments separated by a distance that equals the door width. A split and merge method [CT96] is used for laser scan segmentation.

To avoid false positive door detections due to similar structures around the scenario, we rely on the current estimation of the robot so that only the doors detected in the expected GPS position of the correct door are considered as true positives.

Door detection is not needed for in–out transitions because the GPS provides absolute measurements so there is no need to initialize the outdoor localization.

3.3.4 Outdoor Localization

The localization method uses the GPS information to localize the robot when it is Outdoors (Figure 3.6), providing an absolute measurement of the position of the robot.

However, in this framework, the only direct source of absolute orientation estimations is the map-based method, as the GPS only provides position estimations directly and so the orientation of the robot must be obtained in another way. A compass can measure the orientation but it is affected by the magnetic fields in this environment, making its estimations useless in scenarios with sources of magnetic fields. In the absence of other information, odometry/IMU estimations can be used for this purpose, but they accumulate error during the movement. By estimating the orientation from several GPS measurements this error can be periodically reduced, but it is only accurate when the robot is moving close to a straight line path. In addition, the convergence to the real orientation is faster if we add direct measurements than letting the filter compute it using the correlation with the position variables.

We propose the computation of the orientation of the robot from a set of consecutive GPS measurements $\{\mathbf{z}_{gps_k}, \cdots, \mathbf{z}_{gps_{k+n}}\}$, under some constraints. For each of these measurements, a partial estimation of the orientation $\hat{\theta}_{k+j}$ is computed from $\mathbf{z}_{gps_k}$ and $\mathbf{z}_{gps_{k+j}} (j = 1 \dots n)$ following equation (3.15). The covariances of the measurements $\mathbf{R}_{gps_k}$ and $\mathbf{R}_{gps_{k+j}}$ are propagated by the Jacobians to obtain the covariance of the partial estimation σ^2_{k+j}, as shown in equation (3.16), where $\mathbf{J}_{gps_k}$ is the Jacobian of $\hat{\theta}_{k+j}$ with respect to $\mathbf{z}_{gps_k}$ and $\mathbf{J}_{gps_{k+j}}$ is the Jacobian with respect to $\mathbf{z}_{gps_{k+j}}$. A final estimation of the orientation $\hat{\theta}_{gps_k}$ is calculated as a weighted sum of the partial estimations using equations (3.18) to, (3.20). The weights are valued depending on the covariance of each partial estimation.

$$\hat{\theta}_{k+j} = \arctan\left(\frac{y_{gps_{k+j}} - y_{gps_k}}{x_{gps_{k+j}} - x_{gps_k}}\right) \tag{3.15}$$

$$\sigma^2_{k+j} = \mathbf{J}_{gps_k}\mathbf{R}_{gps_k}\mathbf{J'}_{gps_k} + \mathbf{J}_{gps_{k+j}}\mathbf{R}_{gps_{k+j}}\mathbf{J'}_{gps_{k+j}} \tag{3.16}$$

$$cov(k+i, k+j) = \mathbf{J}_{gps_{k+i}}\mathbf{R}_{gps_k}\mathbf{J'}_{gps_{k+j}} \tag{3.17}$$

$$\hat{\theta}_{gps_k} = \sum_{j=1}^{n} w_{k+j}\hat{\theta}_{k+j} \tag{3.18}$$

$$\sigma^2_{gps_k} = \sum_{j=1}^{n} w^2_{k+j}\sigma^2_{k+j} + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{k+i}w_{k+j}cov(k+i, k+j), i \neq j \tag{3.19}$$

$$w_{k+j} = \frac{1}{n-1}\left(1 - \frac{\sigma^2_{k+j}}{\sum_{j=1}^{n}\sigma^2_{k+j}}\right) \tag{3.20}$$

To consider this estimation useful, the constraints are: a minimum distance d between the initial measurement $\mathbf{z}_{gps_k}$ and the following one considered for calculation $\mathbf{z}_{gps_{k+j}}$ to assure that the robot is moving and that the uncertainty of the first measurement does not include the position given by the next GPS measurement considered; a minimum number of GPS measurements n such that the distance between $\mathbf{z}_{gps_k}$ and $\mathbf{z}_{gps_{k+n}}$ is L, for a given GPS data frequency; finally, the steering angle ϕ of the robot must be close to 0, making sure the robot is moving in a straight line and thus the estimated orientation is constant.

Then, the GPS estimation $\mathbf{y}_{gps}$ provided to correct the localization of the robot will be different. We consider two cases in order to compute the covariance matrices in equations

(3.10) and (3.11). If the orientation has been calculated from GPS measurements,

$$\mathbf{y}_{gps} = \mathbf{H}' \mathbf{z}_{gps}, \quad \text{where} \quad \mathbf{z}_{gps} = \begin{pmatrix} x_{gps} \\ y_{gps} \\ \hat{\theta}_{gps} \end{pmatrix}, \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_{gps} = \begin{pmatrix} \mathbf{R}_{x_{gps}} & 0 & -\frac{w_n}{L}\mathbf{R}_{x_{gps}} \sin\hat{\theta}_{gps} \\ 0 & \mathbf{R}_{y_{gps}} & \frac{w_n}{L}\mathbf{R}_{y_{gps}} \cos\hat{\theta}_{gps} \\ -\frac{w_n}{L}\mathbf{R}_{x_{gps}} \sin\hat{\theta}_{gps} & \frac{w_n}{L}\mathbf{R}_{y_{gps}} \cos\hat{\theta}_{gps} & \sigma^2_{gps} \end{pmatrix}$$

where $\mathbf{R}_{x_{gps}}$ and $\mathbf{R}_{y_{gps}}$ are the variances of the position measured by the GPS. Otherwise, the orientation is taken from the odometry/IMU, and therefore,

$$\mathbf{y}_{gps} = \mathbf{H}' \mathbf{z}_{gps}, \quad \text{where} \quad \mathbf{z}_{gps} = \begin{pmatrix} x_{gps} \\ y_{gps} \end{pmatrix}, \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{R}^{-1}_{gps} = \mathbf{H}' \begin{pmatrix} \frac{1}{\mathbf{R}_{x_{gps}}} & 0 \\ 0 & \frac{1}{\mathbf{R}_{y_{gps}}} \end{pmatrix} \mathbf{H}$$

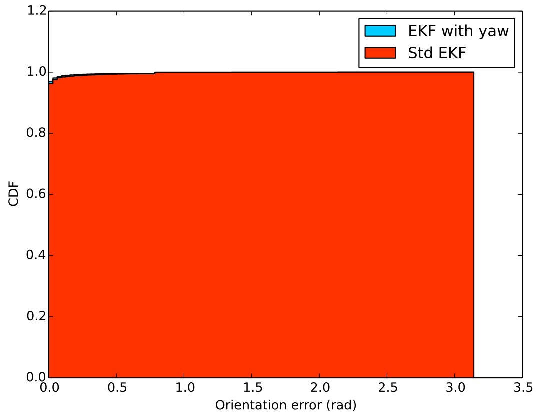
We have tested our method with an experiment where we simulate a straight line movement and we add variable error to the GPS measurements. Along the path, we estimate the position and orientation using these measurements and a simulated odometry. We have recorded the error in orientation estimation along the path for 20 different runs. Figure 3.11 shows the cumulative distribution of the error in the estimation of the orientation using standard EKF and the EKF with our orientation estimation method. It is noticeable that the performance of our method is similar to the standard EKF when the GPS error is small. However the standard EKF cannot reduce the estimation error as fast as our method when the GPS error is big.

Observability Analysis

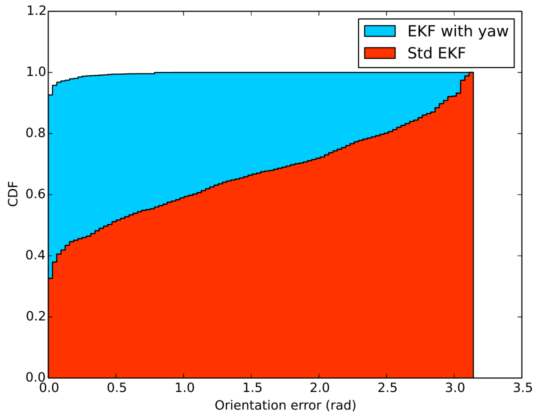
The method explained above provides an absolute estimation of the robot orientation if the constraints defined are satisfied. But, when they are not satisfied, the error grows due to the lack of an estimation to correct it. We provide in this section a theoretical analysis about the conditions for the filter stability, demonstrating the observability of our system and providing bounds which are used in the orientation estimation to verify that the error remains limited.

In [RGYU99], it is demonstrated that the estimation error of an EKF remains bounded as long as the system is observable and the initial estimation error and the noise terms are small enough. A methodology to find the bounds is also provided.

According to [HK77], a non linear system is *locally weakly observable* if the system satisfies the observability rank condition for every state $\mathbf{x}$ in the state space. A system satisfies the observability rank condition if the gradient of the Lie derivative matrix $\mathbf{G}$ in (3.21) is a full rank matrix. Equation (3.21) defines the Lie derivative matrix $\mathbf{G}$ and its gradient $d\mathbf{G}$ (the observability matrix $\mathbf{O}$), where ${L_f}^i(h)$ is the i-order Lie derivative



(a) Orientation estimation error with small GPS error ($\sigma^2_{GPS} < 10m^2$).



(b) Orientation estimation error with big GPS error ($\sigma^2_{GPS} < 50m^2$).

Figure 3.11: Cumulative density functions for the error in the estimation of the orientation. For every error (in the horizontal axis), the CDF is the probability of finding an smaller error during the experiment. For example, in the (b) experiment,the probability of having an error smaller than 1.0 radians is almost 1.0 with our approach but just 0.5 with the standard EKF approach.

of h with respect to f, h is the measurement function, f is the defining function of the system and n_x is the number of state variables. The Lie derivatives of h with respect to f of different orders are expressed in (3.22).

$$\mathbf{G} = \begin{pmatrix} L_f^0 h \\ L_f^1 h \\ \cdots \\ L_f^{n_x-1} h \end{pmatrix}, \qquad d\mathbf{G} = \mathbf{O} = \begin{pmatrix} dL_f^0 h \\ dL_f^1 h \\ \cdots \\ L_f^{n_x-1} h \end{pmatrix} \tag{3.21}$$

$$L_f^0 h = h, \quad L_f^1 h = \frac{\partial h}{\partial \mathbf{x}} \cdot f \quad \ldots \quad L_f^i h = \frac{\partial}{\partial \mathbf{x}}[L_f^{i-1} h] \cdot f \tag{3.22}$$

The state equation of our system (3.5) can be rewritten as,

$$f(\mathbf{x}_k, \mathbf{u}_{od_{k+1}}) = \mathbf{x}_{k+1} = \mathbf{x}_k \oplus \mathbf{u}_{od_{k+1}}, \quad \mathbf{x}_k = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_k, \quad \mathbf{u}_{od_{k+1}} = \begin{pmatrix} u_x \\ u_y \\ u_\theta \end{pmatrix}_{k+1} \tag{3.23}$$

where,

$$f(\mathbf{x}_k, \mathbf{u}_{od_{k+1}}) = \begin{pmatrix} x_k + u_{x_{k+1}} \cos\theta_k - u_{y_{k+1}} \sin\theta_k \\ y_k + u_{x_{k+1}} \sin\theta_k + u_{y_{k+1}} \cos\theta_k \\ \theta_k + u_{\theta_{k+1}} \end{pmatrix} \tag{3.24}$$

The measurement function h for our system defines the position given by the GPS, which is,

$$h(\mathbf{x}_{k+1}) = \mathbf{z}_{gps_{k+1}} = \begin{pmatrix} x_{gps} \\ y_{gps} \end{pmatrix}_{k+1} \tag{3.25}$$

Next, we define the Lie derivatives of h with respect to f and the observability matrix $\mathbf{O}$, following expressions in (3.21, 3.22). To demonstrate that $\mathbf{O}$ is full rank, it suffices to show that a subset of its rows are linearly independent. Thus, we state the equations up to order 1:

$$\begin{aligned}
L_f^0 h &= h = \begin{pmatrix} x_{gps} \\ y_{gps} \end{pmatrix} \\
L_f^1 h &= \frac{\partial h}{\partial \mathbf{x}} \cdot f = \begin{pmatrix} x_k + u_{x_{k+1}} \cos\theta_k - u_{y_{k+1}} \sin\theta_k \\ y_k + u_{x_{k+1}} \sin\theta_k + u_{y_{k+1}} \cos\theta_k \end{pmatrix} \\
\mathbf{G} &= \begin{pmatrix} h \\ L_f^1 h \end{pmatrix} = \begin{pmatrix} x_{gps} \\ y_{gps} \\ x_k + u_{x_{k+1}} \cos\theta_k - u_{y_{k+1}} \sin\theta_k \\ y_k + u_{x_{k+1}} \sin\theta_k + u_{y_{k+1}} \cos\theta_k \end{pmatrix} \\
d\mathbf{G} = \mathbf{O} &= \begin{pmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \\ 1 & 0 & -u_{x_{k+1}} \sin\theta_k - u_{y_{k+1}} \cos\theta_k & \text{(I)} \\ 0 & 1 & u_{x_{k+1}} \cos\theta_k - u_{y_{k+1}} \sin\theta_k & \text{(II)} \end{pmatrix}
\end{aligned} \tag{3.26}$$

To check if our observability matrix $\mathbf{O}$ has rank $n_x = 3$ we focus on expressions (I) and (II) of equation (3.26), and write them as in (3.27). The determinant of the first term is 1 $\forall\theta_k$, which implies that the product is 0 only if $u_{x_{k+1}} = 0$ and $u_{y_{k+1}} = 0$, i. e., when the robot is stopped. It is clear that, if the robot does not move, it will not be possible to compute the interpolation of the positions to estimate the orientation. Otherwise, we can assure that our system satisfies the observability rank condition whenever the robot moves, that is when $u_{x_{k+1}} \neq 0$ or $u_{y_{k+1}} \neq 0$ $\forall\theta_k$.

$$\begin{pmatrix} -u_{x_{k+1}} \sin\theta_k - u_{y_{k+1}} \cos\theta_k \\ u_{x_{k+1}} \cos\theta_k - u_{y_{k+1}} \sin\theta_k \end{pmatrix} = \begin{pmatrix} -\sin\theta_k & -\cos\theta_k \\ \cos\theta_k & -\sin\theta_k \end{pmatrix} \begin{pmatrix} u_{x_{k+1}} \\ u_{y_{k+1}} \end{pmatrix} \tag{3.27}$$

The stability analysis assures the convergence of the filter and the boundedness of the estimation error given some conditions. In particular, theorem 3.1 in [RGYU99] states that the convergence is assured if the following conditions hold:

1. There are some positive real numbers f_h, h_h, p_l, p_h, q_l, r_l such that the following

bounds are satisfied $\forall k \geq 0$:

$$\|\mathbf{F}_k\| \leq f_h \tag{3.28}$$

$$\|\mathbf{H}_k\| \leq h_h \tag{3.29}$$

$$p_l I \leq \mathbf{P}_k \leq p_h I \tag{3.30}$$

$$q_l I \leq \mathbf{Q}_k \tag{3.31}$$

$$r_l I \leq \mathbf{R}_k \tag{3.32}$$

where $\|\cdot\|$ denotes the spectral norm of a matrix, $\mathbf{F}_k$ is the matrix from the linearized equation (3.24), $\mathbf{H}_k$ is from measurement equation (3.25), and the others are the covariance matrices.

2. $\mathbf{F}_k$ is not singular $\forall k \geq 0$.

3. There are some positive real numbers κ_f, $\kappa_h > 0$ such that the linearization errors are bounded. κ_f is the bound for the error of linearizing the system equation (3.24), whereas κ_h bounds the linearization error for the measurement function (3.25).

Then, the estimation error of the filter is bounded, provided that there are some positive real numbers $\epsilon, \delta > 0$ which bound the initial estimation error and the covariance matrices of the noise terms. ϵ is the bound for the initial error of the filter estimation, and δ represents the bound for the matrices $\mathbf{Q}_k$, $\mathbf{R}_k$. The specific bounds proving the observability of our system and a practical case of the filter convergence are analyzed in section 3.5.3.

Restrictions to the GPS-based Orientation Computation

The constraints to compute the orientation from GPS measurements proposed in [ULVM13] were too restrictive. In this section, less conservative restrictions are established, allowing the frequency of orientation estimation to be increased, thus reducing the orientation error accumulated from the odometry/IMU estimations. These restrictions are computed and analyzed from the results of several experiments.

The vehicle under consideration has Ackermann kinematics, following equations (3.33), where D is the distance between the two wheel axes, (x, y, θ) is the pose of the robot, v is the linear speed, and ϕ the robot steering angle. Restrictions on the maximum and minimum steering angle $\phi \in [\phi_{min}, \phi_{max}]$ are equivalent to constraints in the minimum turning radius.

$$\dot{x} = v\,\cos\theta, \qquad \dot{y} = v\,\sin\theta, \qquad \dot{\theta} = v\,\frac{\tan\phi}{D} \tag{3.33}$$

In the previous preliminary work the steering angle ϕ of the wheels was used in order to determine if the robot moves in a straight line, when $\phi \simeq 0$. However, this is not always a good measure of this behavior, as is shown by the first experiment. In this case, the robot moves at constant linear velocity and the steering angle ϕ takes values from equation (3.34). Parameter A takes values from $\{0.1, 0.2, 0.3, 0.4\}$ and ω from $\{0.0, 0.5, 1.0, 1.5, 2.0\}$.

$$\phi(t) = A\sin(\omega t) \tag{3.34}$$

Figure 3.12 illustrates the results obtained with $A = 0.4$ and $\omega = 2$. In Figure 3.12a, the GPS positions of the robot are depicted, showing the robot moving almost in a straight line. In Figure 3.12b, the euclidean distance traveled during the experiment with respect to the initial measurement is illustrated. The steering angle, the orientation from the odometry/IMU and the orientation computed from GPS measurements are shown in Figure 3.12c, which shows that the amplitude of the steering angle is bigger than that provided by the odometry/IMU. So, in general we should consider another criterion different from the steering angle to determine whether the robot is moving in a straight line. Thus, assuming the kinematics of our robot, we consider for the calculation those only measurements which fit within the maximum feasible turn. This is shown in equations (3.35) and (3.36), which are related to equations (3.15) to (3.20).

$$|\hat{\theta}_{gps_{k-1}} - \hat{\theta}_{k+j}| < th, \quad j = 1 \dots n \tag{3.35}$$

$$th = v\,\frac{\tan(\phi_{max})}{D} \tag{3.36}$$

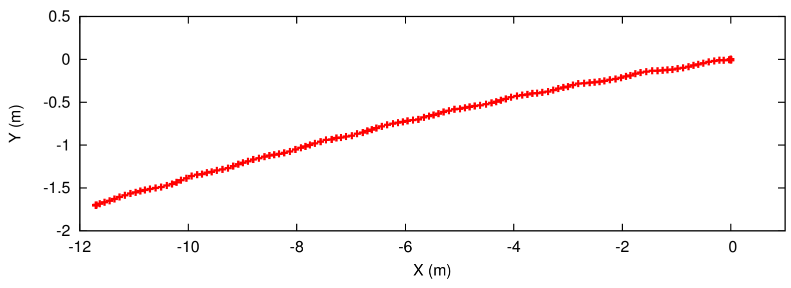
Equations (3.37) and (3.38) define the other constraints which limit the precision of the computed orientation estimation: the minimum distance d to obtain the first partial estimation $\hat{\theta}_{k+j}$, and the total distance L between the initial measurement $\mathbf{z}_{gps_k}$ and the last measurement $\mathbf{z}_{gps_{k+n}}$ to compute the final estimation from the weighted sum of the partial estimations (equation 3.18).

$$|\mathbf{z}_{gps_{k+j}} - \mathbf{z}_{gps_k}| \geq d, \quad j = 1 \dots n \tag{3.37}$$

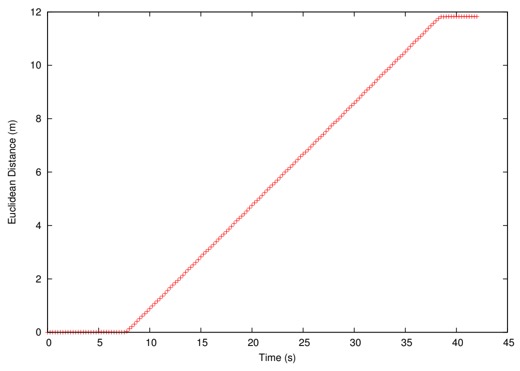
$$|\mathbf{z}_{gps_{k+n}} - \mathbf{z}_{gps_k}| \geq L \tag{3.38}$$

Figure 3.12c shows that when the robot is not moving the value of the orientation jumps, and thus there is a minimum distance d needed to obtain a good estimation for the orientation. In the experiment we have measured the error in the estimations while navigating the robot as straight as possible along 12 m at constant speed. Figure 3.13 shows the mean square error of the estimations. For distances d greater than 0.4 m there is no improvement in the estimations. Therefore, whenever the procedure for estimating the orientation is initialized, the minimum distance d between the initial measurement $\mathbf{z}_{gps_k}$ and the following measurement should be 0.4 m.

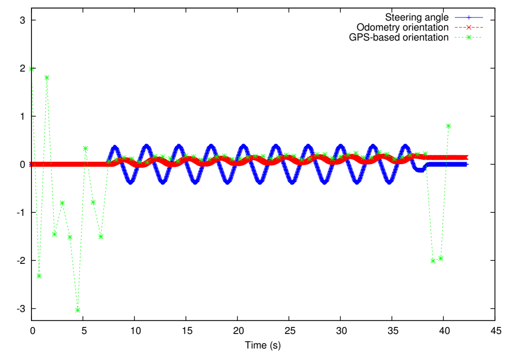
The constraint on the total distance L between the initial $\mathbf{z}_{gps_k}$ and the last $\mathbf{z}_{gps_{k+n}}$ measurements used to compute the estimation of the orientation (3.38) is examined by performing several experiments in which the robot moves following non-straight paths. Figure 3.14a shows the total number of estimations obtained during the experiment for different values of L and Figure 3.14b shows the mean of the covariances of the orientation estimations, for a GPS measurement frequency of 4 Hz. Values higher than 1 m decrease the frequency of the orientation updates as the number of total estimations is considerably reduced, so these are discarded. A value of $L = 1$ m reduces by almost



(a) GPS measurements of the position of the robot during the experiment.



(b) Euclidean distance with respect to the first measurement. The robot starts moving after 7 seconds.



(c) Steering angle, orientation from odometry/IMU and orientation estimated from GPS measurements.

Figure 3.12: Results obtained from the experiment following (3.34) with $A = 0.4$ and $\omega = 2$. The experiment is intended to find out how to detect that the robot is moving straight for different values of steering angle. Figure (c) shows that, even if the steering wheels are moving, if the variation is fast, it is not directly translated into significant variations in the orientation of the robot.

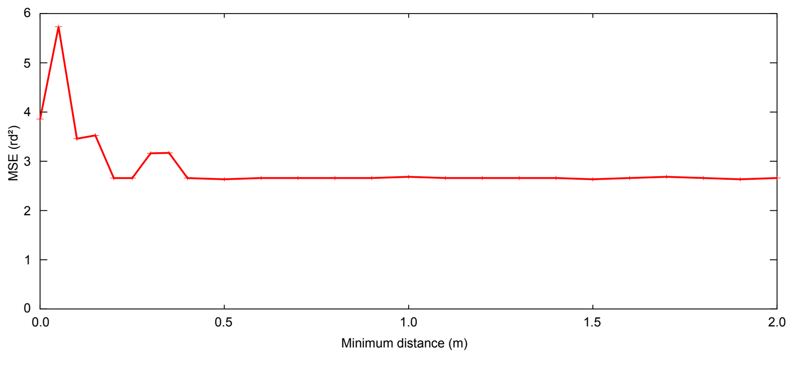
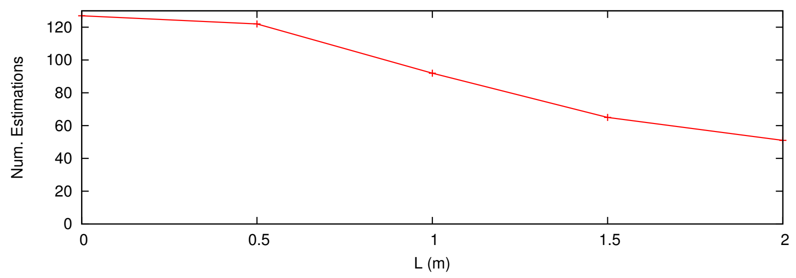
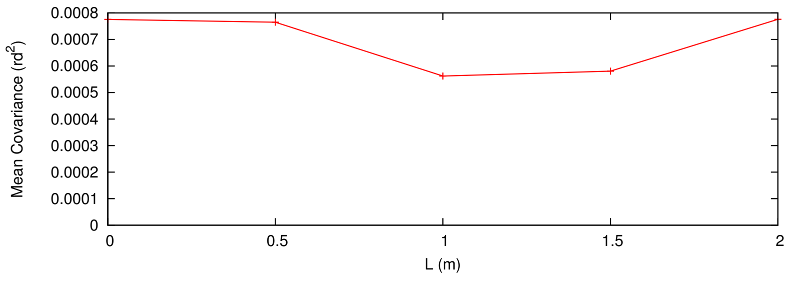


Figure 3.13: Mean square error of the estimation of the orientation using GPS measurements in straight line for different values of minimum distance constraint d needed to compute the first partial estimation $\hat{\theta}_{k+j}$.



(a) Number of estimations of the orientation.



(b) Mean of the covariances of the total estimations for different distances.

Figure 3.14: Results of the experiment to determine the total distance L between measurements at k and $k + n$.

25% the number of estimations computed with respect to $L = 0.5\ m$, whereas there is not much difference in the covariance. Thus, we consider that a total distance L of 0.5 m increases the number of orientation updates and provides good estimations of the orientation.

Algorithm 3.1 formally outlines the method to compute the orientation of the robot from the GPS measurements following equations (3.15)–(3.20) and taking into account the new constraints. In line 10, if the partial estimation $\hat{\theta}_{k+j}$ is not under the threshold th from equation (3.35) we discard the partial estimation and initialize the reference GPS measurement with the current one. Line 15 reflects that once the estimation of the orientation has been finally calculated, the process is initiated and the GPS measurement of reference is initialized with the current one.

Precise and seamless localization is a very important issue for real robotic applications. Localization, planning and reactive navigation have to be integrated. The following section describes the development of robust planning and navigation techniques for challenging scenarios, in which seamless localization is used.

3.4 Navigation

Long-term motion planning is usually used jointly with reactive navigation techniques to provide robust and adaptive navigation for robots. In real and changing environments, global planning alone cannot solve many situations that may appear during the motion.

Algorithm 3.1 Estimation of the orientation from GPS measurements

```
Require:
1: $\mathbf{z}_{gps_k}$ is the initial GPS measurement taken as reference
2: $\mathbf{z}_{gps_{k+j}}$ is the current GPS measurement
3: $th$ is the maximum variation of orientation the robot can experience since the previous
   estimation
4: $d$ is the minimum distance between $\mathbf{z}_{gps_k}$ and $\mathbf{z}_{gps_{k+j}}$
5: $L$ is the total distance to compute the final estimation
6: $\hat{\theta}_{gps_{k-1}}$ is the previous estimation of the orientation
7: function GPS-BasedOrientation($\mathbf{z}_{gps_k}$, $\mathbf{z}_{gps_{k+j}}$, $th$, $d$, $L$, $\hat{\theta}_{gps_{k-1}}$)
8:     if $|\mathbf{z}_{gps_{k+j}} - \mathbf{z}_{gps_k}| \geq d$ then
9:         Compute $(\hat{\theta}_{k+j}(\mathbf{z}_{gps_k}, \mathbf{z}_{gps_{k+j}}), \sigma^2_{k+j}(\mathbf{z}_{gps_k}, \mathbf{z}_{gps_{k+j}}))$ ▷ Equations (3.15), (3.16)
10:        if $|\hat{\theta}_{gps_{k-1}} - \hat{\theta}_{k+j}(\mathbf{z}_{gps_k}, \mathbf{z}_{gps_{k+j}})| > th$ then
11:            $\mathbf{z}_{gps_k} = \mathbf{z}_{gps_{k+j}}$
12:        else
13:            if $|\mathbf{z}_{gps_{k+j}} - \mathbf{z}_{gps_k}| \geq L$ then
14:                return Compute$(\hat{\theta}_{gps_k}, \sigma^2_{gps_k})$ ▷ Equations (3.18), (3.20)
15:                $\mathbf{z}_{gps_k} = \mathbf{z}_{gps_{k+j}}$
16:            end if
17:        end if
18:    end if
19: end function
```

Frequent re-planning is needed to cope with the dynamism of the environment detected in the sensors' field of view. Two main approaches can be adopted: using planners developed for dynamic environments (e.g. D^*-like) jointly with well-known and robust reactive navigation; or using global planners jointly with reactive techniques enriched with robust maneuvering for cluttered rooms. We adopt the second approach in this work, given that planning complex maneuvers in changing or dynamic environments is costly and less flexible than applying adaptive local maneuvering. Moreover, the techniques have to be adapted to the kinodynamic constraints of the robot. We develop and evaluate a novel reactive navigation and maneuvering technique for car-like robots, which allows the original planned trajectory to be modified in real time.

In the proposed method, the global planner provides, as a list of subgoals, paths from the initial position of the robot to the goal position. Instead of planning the trajectory considering the kinematic constraints on the robot (i.e. curvature, velocities), these are left to be managed by the reactive navigation method. In a changing scenario this solution is more flexible because new obstacles appearing in the field of view of the sensors could invalidate the long-term planned path.

3.4.1 Obstacle Avoidance

We develop an obstacle avoidance technique (NDW) that takes advantage of two different approaches: the Nearness Diagram [MM04] (ND) and the Dynamic Window (DW) [FBT97] adapted for car-like robots.

The ND method builds a model of the environment based on the *valley* concept, an area where the robot can safely navigate. The best *valley* is selected as the one that contains or is closest to the goal, and then the best command for the robot to navigate through that *valley* is geometrically computed. However, the original approach was designed for differential drive robots which are able to turn in place. On the one hand, dealing with car-like robots imposes constraints on the steering angle ϕ, and thus in the changes of the robot orientation. On the other hand, the kind of scenarios we consider include cluttered and non cluttered areas. The ND technique performs better in very cluttered environments, exhibiting a worse performance in open areas.

The DW approach is based on optimizing a navigation function that evaluates aspects such as safety and progress towards the goal for each possible command that the robot can select. This approach computes a solution that maximizes the balanced criteria. However, the selection of a suitable evaluation function and the balance between the different terms is not trivial, and the resulting solution depends on an empirical tuning of the balance weights. Equation (3.39) shows a common DW evaluation function, where $Clearance$ evaluates the safety of the trajectory performed with the steering angle ϕ, measured as the maximum distance the robot can navigate using this steering angle without collision, and $Heading$ evaluates the progress towards the goal position. The speed is not evaluated because the path of the robot does not depend on it.

$$Score(\phi) = \alpha\, Clearance(\phi) + (1-\alpha)\, Heading(\phi) \tag{3.39}$$

Figure 3.15 illustrates an example of the different solutions obtained by the $Score$ function for different weighting α values in a scenario with two corridors. The maximum provided by the DW solution is not always the best one for this scenario because, depending on α, the robot will be directed to the wrong corridor. Instead, the ND solution is always in the correct valley towards the next goal. Our method NDW computes the solution ϕ_{NDW} as that corresponding to the maximum of the $Clearance$ function, but imposing the solution within the valley. So the $Heading$ component of equation (3.39) is not needed in our approach, and no empirical weight α has to be selected.

Algorithm 3.2, computing the steering angle ϕ_{NDW}, performs a DW-like search for a solution but calculated within the best *valley* selected by ND, which in turn includes the kinodynamic robot constraints.

Limitations. The kinematic model of any car-like robot imposes limits on the turning radius. This fact leads to the possibility of finding an unreachable goal if the steering angle required to get to the goal is out of the mechanical bounds. In addition, dynamic or unknown obstacles may eventually appear too close to the front of the robot so that they cannot be avoided by any means if the robot keeps moving forward. These limitations are analyzed in depth in appendix A.

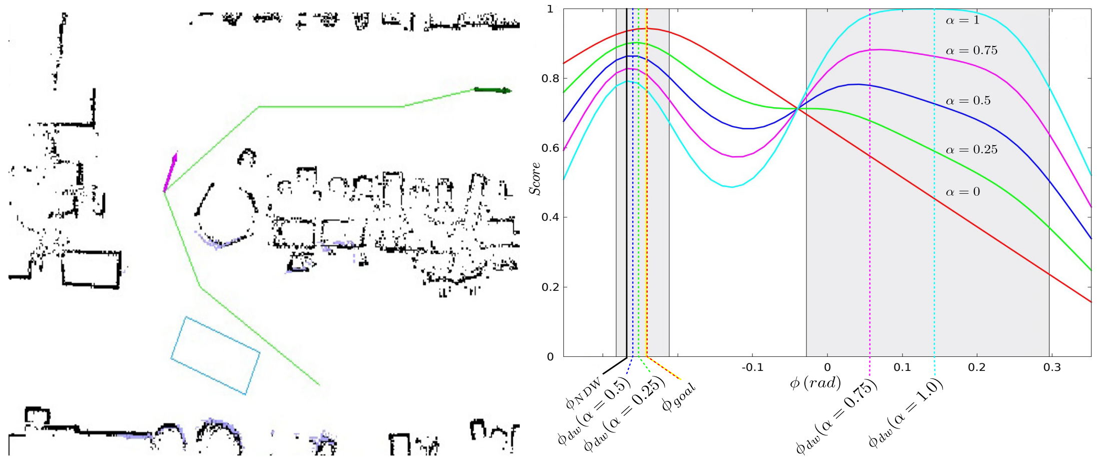


Figure 3.15: A situation where the robot has two navigation zones. Depending on the values of α in the equation (3.39), the global optimum of the score function (at ϕ_{dw}) may not lead the robot to the goal. Instead, a local optimum ϕ_{NDW} inside the valley of the goal (gray zones) is the correct solution.

Algorithm 3.2 Navigation Algorithm

```
Require:
 1: $goal$ is the position of the goal
 2: $Obstacles$ is the set of points where obstacles are
Ensure:
 3: $v$ is the speed for the robot
 4: $\phi_{NDW}$ is the best angle to avoid obstacles while going towards the goal
 5: function NAVIGATION($goal$, $Obstacles$)
 6:     $\phi_{goal} = \tan^{-1}\left(\frac{2Dgoal_y}{goal_x{}^2+goal_y{}^2}\right)$        ▷ goal direction with no obstacles
 7:     $\mathcal{V} = BestValley(\phi_{goal}, Obstacles)$        ▷ as in Nearness Diagram
 8:     $\phi_{NDW} = SearchOptimal(\phi_{goal}, Clearance(Obstacles), \mathcal{V})$        ▷ as in Dynamic Window + valley constraint
 9:     $v = SpeedControl(Clearance(Obstacles), goal)$        ▷ as in Dynamic Window
10:     return $(v, \phi_{NDW})$
11: end function
```

3.4.2 Limitations

Although the ability to maneuver using the method explained in appendix A is able to deal with many situations, some limitations arise from the fact that we are providing a reactive method. It is well known that greedy or unplanned decisions which only consider local information may lead to block situations. Two different problems may appear when maneuvering: oscillations and deadlocks.

- Oscillations appear when the scenario limits the movements of the robot in such a way that, no matter the direction of the movement (forwards or backwards), the robot can only select a very small interval of steering angles and thus the orientation does not change sufficiently to avoid the obstacle.

- Deadlocks occur when there is not enough room for the robot to make the maneuver. This situation is detected when there are obstacles in front of and behind the robot at the same time.

These situations cannot be predicted beforehand if the scenario is not completely known, which is our case. However, deadlocks and oscillations are avoided if there is enough space surrounding the robot to perform the maneuvers successfully.

3.5 Experimental Results and Assessment

In this section we present the experimental results in simulations and in real-world experiments. We evaluate the robustness of the navigation and seamless localization method.

In section 3.5.3 the experiments are oriented to evaluate the localization technique using the EKF filter. The focus is the outdoor localization, in particular the robot orientation computation from the GPS-IMU fusion to obtain the best estimation in each situation. Additionally, the bounds for filter convergence analyzed in the observability section are computed and verified for the system. Section 3.5.4 is devoted to assessing the navigation technique. Finally, in section 3.5.5 a complete experiment covering a wide area of the logistic park is described. Localization and navigation are evaluated in this scenario, paying particular attention to the localization performance during indoor-outdoor transitions, and to the NDW navigation technique performance.

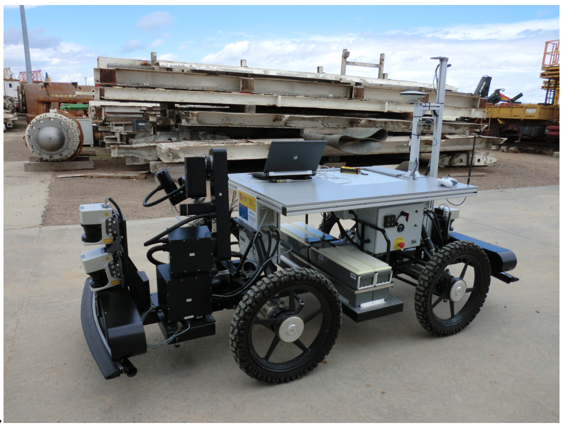
3.5.1 Scenarios and Robot Settings

For the experimentation, we used a customized robucar platform (see Figure 3.16), with two laser rangefinders on the front and one on the back with a 180° field of view, one measurement per degree and a range up to 80 m. IMU corrected odometry is provided at 100 Hz as well as a Novatel differential GPS measurement up to 4 Hz. The odometry is obtained from the wheel encoders. The front wheels have a maximum turning angle of 23° and the vehicle can reach a speed up to $2ms^{-1}$, although during these experiments a maximum speed of $1.5ms^{-1}$ was used.

Two real-world environments were used for the experiments. The first, $Unizar$, is mainly used to evaluate the GPS-based localization technique outdoors. The second, $Park$, is a part of a large operating industrial park of about $100000m^2$, having indoor and outdoor warehouses, in which the robot has to move goods from one to another (see Figure 3.1). In this scenario the whole system integrating all the techniques is evaluated.

3.5.2 Implementation

The system has been implemented in ROS as is now a widely spread robotics framework so that there is plenty of useful methods available. For instance, there is a free implemen-



.

Figure 3.16: The customized robucar used for the experimentation.

tation of the indoor localization method (AMCL, [Fox03]) which was tuned to adapt the technique to the requirements of the scenario. More precisely, the particle distribution is restricted to take into account the specific characteristics of the layout of the indoor scenario (e.g. corridors, dimensions, location of relevant features used for transitions). These restrictions notably improve the localization, avoiding robot losses in most cases, which occurs when the non-tuned technique is used.

In spite of the benefits of using ROS, the implementation of our methods must be adapted to the common specifications. A particular problem appears in the current ROS implementation of the transformations related to the robot odometry for localization. The transformations between different reference frames are required to be organized in a tree, where the nodes are the frames and the branches represent the transformation between a parent frame and a child frame. In other words, ROS does not allow the transformation from one reference frame to more than one parent reference frame. Unfortunately, this is the situation we have in our scenarios, where the robot frame is related to two different reference frames: the GPS frame outdoors and the map frame indoors (see Figure 3.17), which must be related for a continuous localization. To overcome this limitation, instead of providing the GPS $\rightarrow$ robot or the map $\rightarrow$ robot transformation obtained directly from the localization methods, we provide a unique transformation map $\rightarrow$ odometry using the indoor or the outdoor estimation depending on the situation of the robot. This transformation corrects the accumulated error from the raw odometry so that the composition of both transformations provides a good estimation of the robot in the map or GPS reference frame.

Concerning the navigation, we have adapted the techniques to the *navigation stack* of ROS. This stack provides an interface between localization, perception, path planning and obstacle avoidance. This interface has no impact on the development of the navigation method, except for deciding when the path planning is called, in case replanning is needed. Given the state machine design of the navigation method, shown in Figure A.2, we decided to allow replanning on every state change. The idea behind is that a change

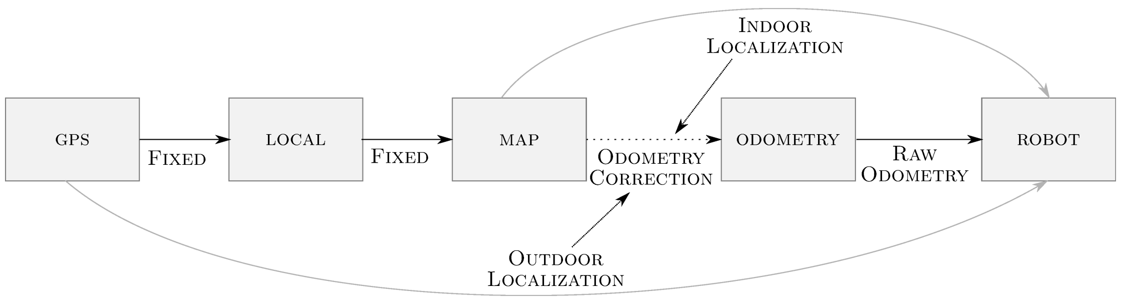


Figure 3.17: ROS reference frames tree used for localization. The system, depending on the situation, uses the indoor localization, the outdoor localization or a mix of them to correct the raw odometry.

in the state of the navigation method is needed when a big change in the scenario occurs and so, a replanning might be necessary to recalculate the best path to the goal.

3.5.3 GPS-based Orientation Results

In this section we describe experiments to evaluate the GPS-based orientation technique and the bounds for the filter convergence, analyzed in the observability section.

Experiment for Evaluation of the GPS-based Orientation Technique

Section 3.3.4 presented a technique to improve the robot orientation estimation in the light of the preliminary work [ULVM13]. Figure 3.18 reproduces the whole layout for the experiment in the *Unizar* scenario, showing the continuity in localization and the bounded uncertainty in the whole trajectory using the unified EKF localization technique developed in this work.

Figure 3.19 shows the improvement obtained by the seamless EKF estimation technique for orientation, depicted in terms of the number of estimations computed by each method during the experiment. Using the steering angle method just a few orientation estimations are computed (green in the figure). The frequency of the orientation estimation is increased when the maximum curvature method proposed in section 3.3.4 is used (red in the figure). As a consequence, the use of the seamless EKF estimation filter jointly with the maximum curvature method increases the orientation updating frequency and reduces the error accumulated from the odometry/IMU, which estimates the orientation when GPS data are not available. The intervals with no orientation estimations reflect the robot is either in Out-In transition or indoors or In-Out transition, where there are poor GPS data (the robot moves near the building) or lack of them. Between time 645 s and 734 s the GPS signal is lost and an estimation of the orientation can not be calculated from GPS measurements. Thus, the robot moves only with the odometry/IMU information.

If we concentrate on the outdoor area from time 645 s to 744 s, there is a no GPS signal event at instant 645 s approximately which lasts until time 734 s. Figure 3.20 focuses

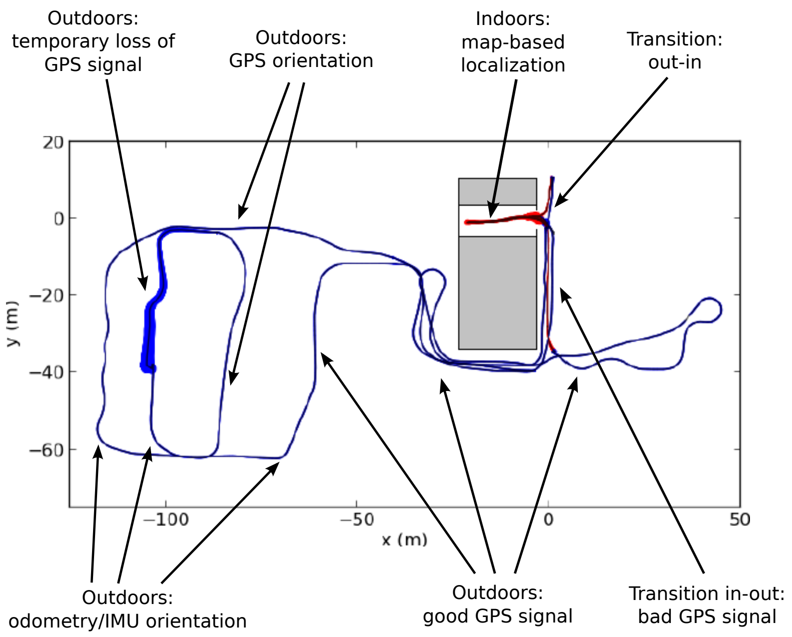


Figure 3.18: Test scenario for evaluating our localization method. This covers an area of 12000 m^2 and the trajectory length is about 1000 meters. The transitions in which the localization estimator has to combine the information from the available sensors are shown.

on this part, showing the robot orientation provided by the EKF and its uncertainty in terms of the 95% confidence interval, and the orientation estimations computed with the maximum curvature method. Since there are no GPS measurements, the uncertainty of the orientation grows with the odometry/IMU error. However, when the GPS signal is recovered (instant 734 s), there is a new GPS measurement providing the position of the robot. Then, the uncertainty of the orientation reduces and remains bounded, which strengthens the observability property of our system. Finally, when there is an orientation estimation available at instant 744 s, the orientation of the EKF converges to it, reducing the uncertainty to the value given by the uncertainty of the absolute orientation estimation provided.

Another experiment in the *Unizar* scenario was carried out to check if our method to estimate the orientation is accurate. Figure 3.21a represents the estimations of the orientation computed with our maximum curvature method. During the experiment, the robot is moving outdoors except inside the rectangle in the Figure 3.21a. Figure 3.21b shows the histogram of the differences of the orientation estimations calculated with our method and the estimations given by the GPS. The mean and median values are also depicted, showing that the 50% of the differences are less than 0.075 *rad*. This strengthens the case for the use of our method when orientation data are not available from the GPS. In our case, there was too much delay in the orientation measurements provided by the GPS and they were thus unusable for real-time estimation.

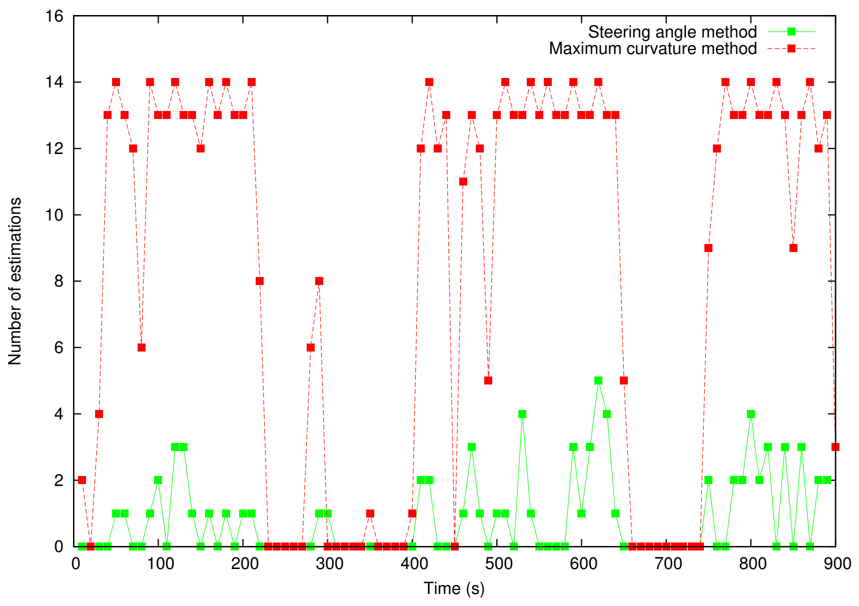


Figure 3.19: Number of estimations obtained with the steering angle method (green) and with the maximum curvature method (red). The new orientation estimation method increases the number of times that an estimation is obtained from GPS measurements.

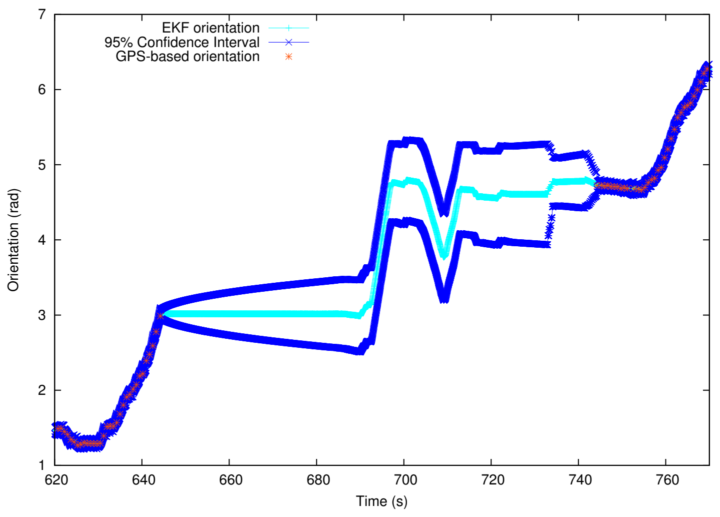
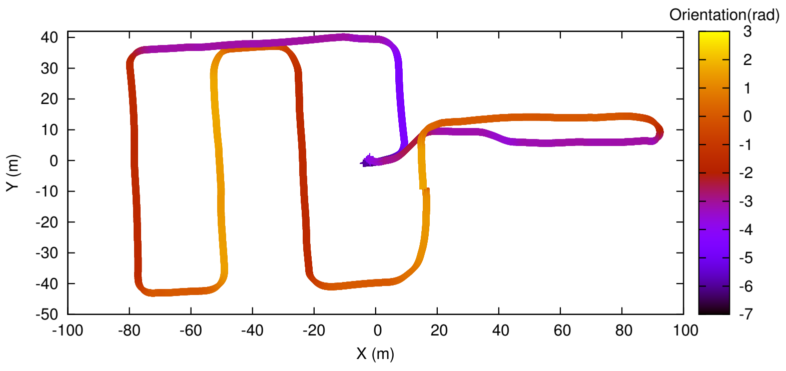
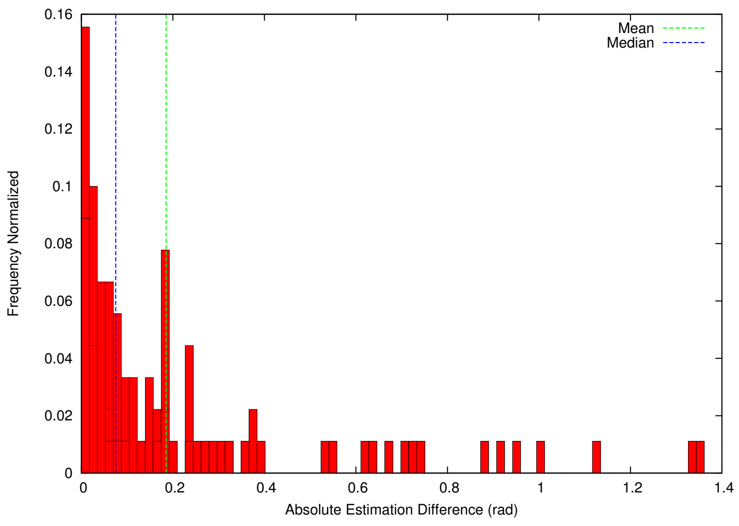


Figure 3.20: Behavior of the EKF when there is no GPS signal (time 645 s). The uncertainty of the orientation grows until a new GPS measurement of position is provided (time 734 s), maintaining the uncertainty bounded. When a new orientation estimation is computed by the GPS-based method (time 744 s), the EKF orientation is updated and the uncertainty of the orientation reduces to the value provided.



(a)



(b)

Figure 3.21: (a) Estimations of the orientation of the robot computed with our maximum curvature method from GPS position measurements. The rectangle includes indoor values, which are not considered for comparison. (b) Histogram of the differences on the orientation estimations between our method and the GPS.

Bounds for Filter Convergence

In the observability analysis, we introduced the conditions for assuring the boundedness of the estimation error provided by the EKF. In order to apply the calculations to estimate the bounds for our system we have considered the simulation results during the experimentation described in Figure 3.18.

The spectral norm of $\mathbf{F}_k$ (3.28) and $\mathbf{H}_k$ (3.29) can be computed before simulation from the formulation of the system, and verified afterwards during simulation. However, the minimum values q_l (3.31) and r_l (3.32) for state and measurement covariances are determined by analyzing the data obtained directly from our robot and the GPS Novatel, which depict the error of the odometry/IMU and the GPS. κ_f, the bound of the error of linearizing the system equation, is obtained from the dynamics of our system (3.33). The maximum linearization error on orientation for the prediction will occur when the robot turns at the maximum steering angle $\phi_{max} = 0.4\ rad$ and the filter estimates that it moves in a straight line. Thus, considering a linear speed of 1.5 m/s and a frequency rate of 100 Hz, the linearization error is valued $\kappa_f = 0.53 \times 10^{-2}$. κ_h, the bound of the linearization error of the measurement function, is set to 0 given that it is a linear function.

The bounds p_l and p_h (3.30) given for the estimation covariance matrix $\mathbf{P}_k$ are obtained from simulation, which correspond to the minimum and maximum eigenvalue of $\mathbf{P}_k$, $\forall k \geq 0$, respectively. Figure 3.22 shows the minimum and maximum eigenvalues during the simulation, expressed in a logarithmic base. The convergence of the eigenvalues is depicted. Whenever there are GPS measurements available, the values remain between the order of 10^{-3} and 10^{-5}. However, when there are poor or no GPS data the values grow because the robot navigates with the odometry/IMU only (intervals between instants 225 s and 280 s, 330 s and 385 s, 680 s and 776 s). The zones where the values are stable correspond to situations where the robot is not moving, and thus where the system is not observable (rank($\mathbf{O}$)=0 (3.21)). In this zone, the maximum values refer to the orientation, whereas the minimum values indicate that the position is estimated with low uncertainty given the data received from GPS.

Finally, the resulting bounds obtained which satisfy the conditions 3.28–3.32 are:

$$f_h = 1.015,\ h_h = 1,\ q_l = 4 \times 10^{-10},\ r_l = 0.0105^2,\ \kappa_f = 0.53 \times 10^{-2},\ \kappa_h = 0$$

It is easy to see that $\mathbf{F}_k$ is non singular. Therefore, as stated by [RGYU99] (theorem 3.1), we can affirm that the estimation error of our filter remains always bounded, when observable.

3.5.4 Navigation Results

In this section we evaluate the navigation technique in an indoor scenario. Figure 3.23 represents the scenario mapped using a rangefinder laser scanner. It corresponds to an indoor zone in the *Park* scenario. To test the robustness of our method, we have fixed six goal locations in the scenario that are selected randomly. The locations of the goals have been selected in relevant places for navigation (end of corridors) where the robot has enough space to maneuver. We have also added unmapped objects to verify the ability of the robot to avoid unexpected obstacles, not included in the map used for planning. Results concerning the maneuvering technique are exposed in appendix A.

We now compare the NDW navigation with the Dynamic Window (DW) and the Nearness Diagram (ND) methods in the scenario shown in Figure 3.23. For the DW we use different values of the empirical parameter α.

Figure 3.24 shows the results of reaching 50 random goals for all the methods. For some values of α, the DW method was unable to reach the goals because the robot was commanded too close to the obstacles and also because it selected a wrong direction to

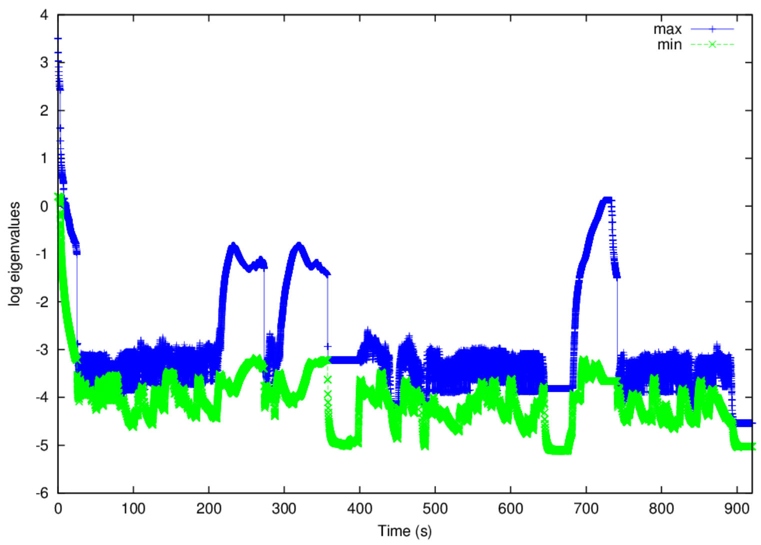


Figure 3.22: Minimum and maximum eigenvalues of the covariance matrix $\mathbf{P}_k \, \forall k \geq 0$, during the simulation. Upper values of the maximum eigenvalues correspond to non observable situations.

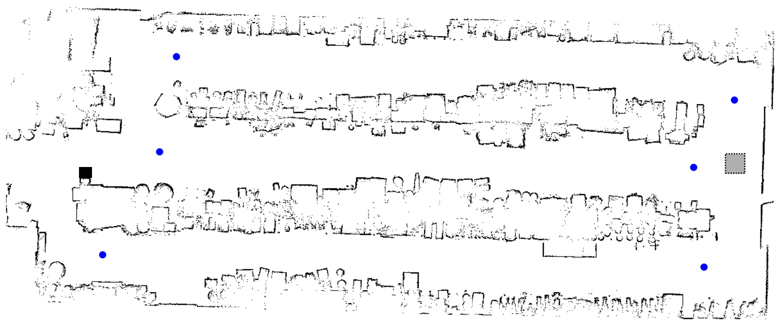
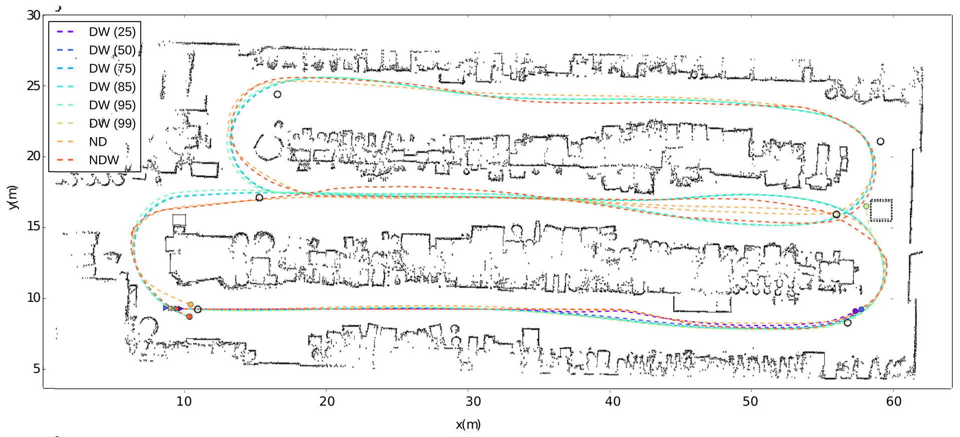


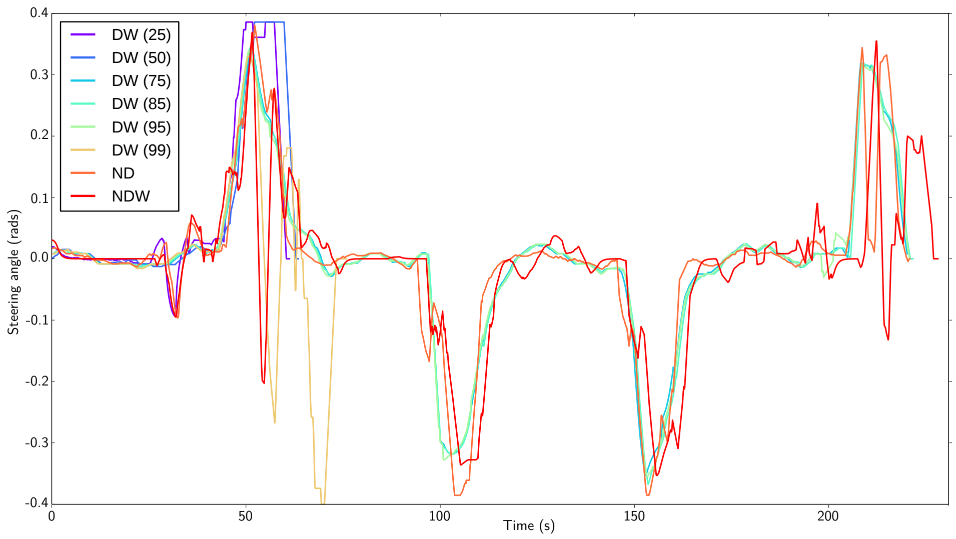
Figure 3.23: Indoor navigation scenario. Goals to be reached are represented by blue dots and the unmapped obstacles are the gray and black boxes.

follow the path. Although the other methods are able to reach the goal of the trajectory, the NDW method commands uses smaller steering angles (closer to 0) than the other methods, thus providing smoother trajectories. Table 3.2 shows the average steering angle and the variance for each navigation method.

Although some results using DW for a given α are equivalent or even better than those obtained with the NDW, the problem of finding the optimal value for the parameter is still present. Figure 3.24b, for instance, shows that the evolution of the steering angle with $\alpha = 0.95$ is smoother than the obtained with the ND and the NDW methods. But using a slightly different value $\alpha = 0.99$ makes the method unable to reach the goal safely. This high dependence on the parametrization of the DW method is solved by



(a) The trajectories performed given the same scenario as in Fig. 3.23. The trajectories start in the triangles and terminate at the circles. The *DW* method could not complete the trajectory for α values under 0.5 and over 0.99.



(b) The commanded steering angle for each navigation method. See Table 3.2 for statistics.

Figure 3.24: Comparison of obstacle avoidance methods to compute the steering angle commands using DW for different values of clearance parameter α, ND, and NDW.

using the NDW method.

3.5.5 Experiment in a Large Logistics Environment

The objective of this experiment was to evaluate the whole system in a real operating environment. The experiments in the scenario included integrated seamless indoor and outdoor localization, planning and navigation, and maneuvering. The robot was

Table 3.2: Comparing the inputs of different navigation methods

Method	Mean Angle (rad)	Variance (rad)
DW (0.25)	0.0857	0.0178
DW (0.50)	0.0849	0.0188
DW (0.75)	0.0725	0.0108
DW (0.85)	0.0728	0.0109
DW (0.95)	0.0731	0.0107
DW (0.99)	0.0792	0.0124
ND	0.0679	0.0118
NDW	0.0657	0.0089

autonomously localized and navigated around the scenario for 40 minutes. Its only commands were to reach a series of goals. The total distance of the experiment was about 1500 meters, reaching four times every goal commanded.

Figure 3.25a shows the mapped scenario, the full trajectory of the experiment, and the different transitions between indoor and outdoor zones. First a map was built in the indoor zone of the warehouse with the information from the rangefinder. This was subsequently used for navigating and localizing during the full experiment. The outdoor obstacle information, also built from the rangefinder sensors, is shown in the figure only to visualize the navigable areas, but it was not used for localizing.

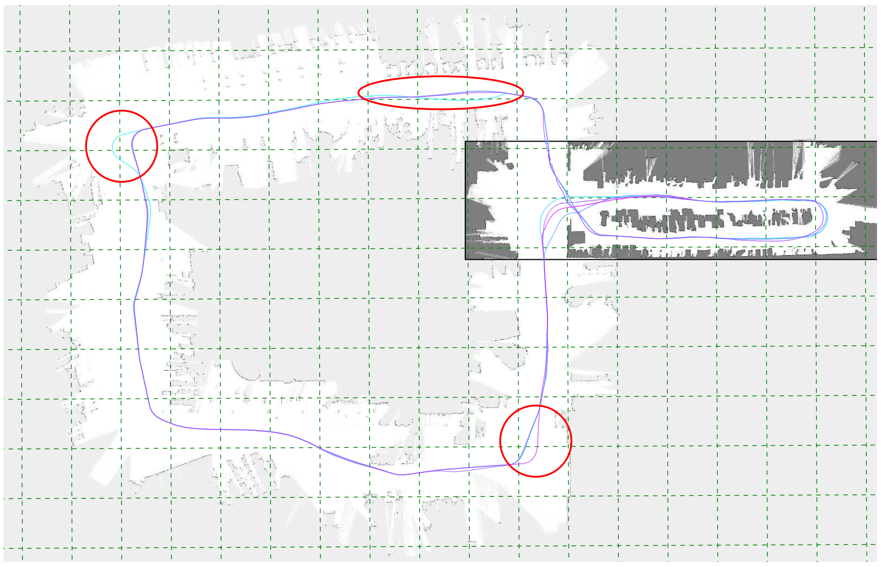
Throughout the experiment, the activity of the company continued as usual, including people and vehicles moving about. The interaction of the robot with the people and vehicles resulted is some deviations from the trajectory computed by the path planner, circled in Figure 3.25a. Nevertheless, even with some changes in the scenario, the system is robust enough to localize and navigate properly. The navigation performance is analyzed in depth below.

In the detailed view of the transitions (Figure 3.25b), small steps can be seen indicating where the robot moves from indoor to outdoor localization. It should be remembered that during a transition the system smoothly changes from rangefinder sensors to GPS. The use of a common local reference indoors and outdoors, jointly with the seamless EKF estimation, keeps the localization error limited at all times, although little jumps can appear. However, these are similar to the adjustments performed by the indoor localization method. These jumps are analyzed in depth in the next section.

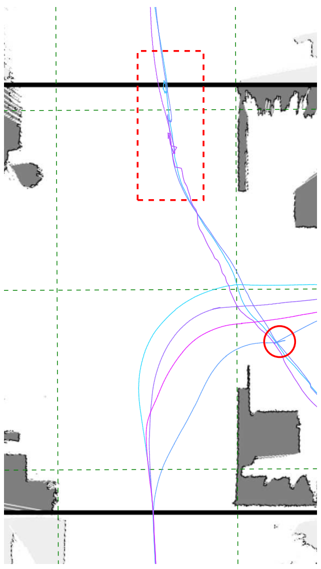
Seamless Localization Results

In this section we analyze the results of the seamless localization system. The key points occur during the transitions between the indoor and the outdoor localization methods.

Figure 3.26 shows the estimation of the movement of the robot along the trajectory. The blue lines represent the lateral movement of the robot as estimated by the localization system. This movement is kinematically impossible because of the non-holonomic constraints of the robot. Slippage of the wheels produces this kind of movements but



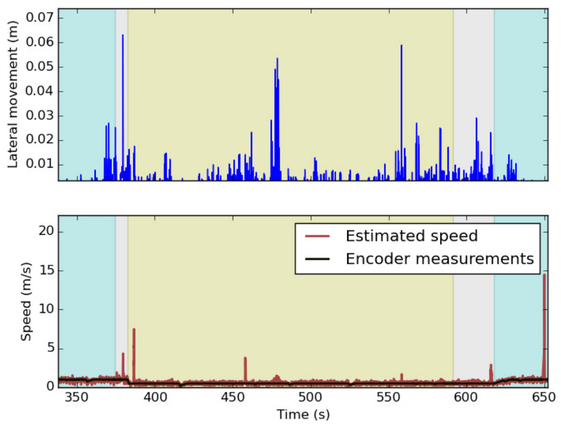
(a)



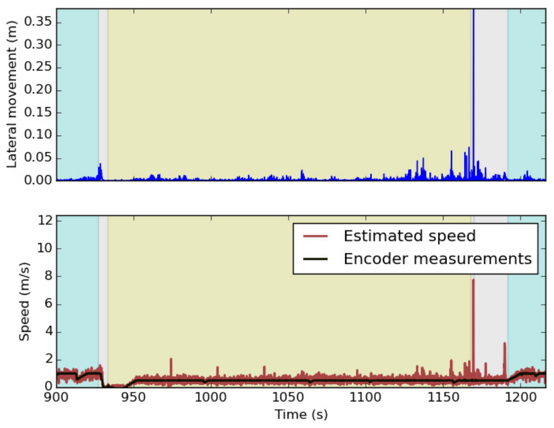
(b)

Figure 3.25: (a) Full trajectory performed by the robot during the experiment in the *Park* scenario and (b) a detailed view of the transition zone. Red circles indicate maneuvers to avoid dynamic obstacles not present in the map. The transition zone between indoor and outdoor localization is marked with a dashed rectangle. Green dashed lines are a 10 m grid.

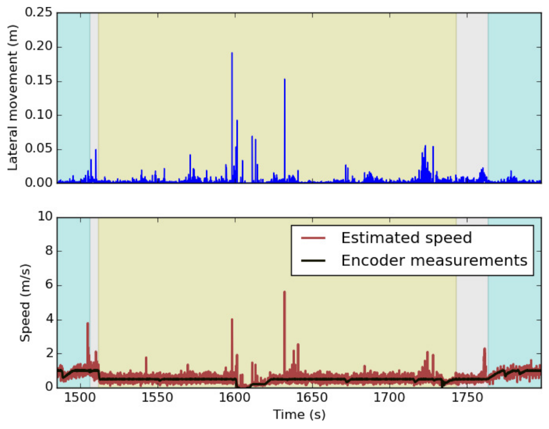
it is usually negligible compared with the effect of the rectifications in the estimation of the localization. In other words, in reality the robot hardly moves laterally. These mea-



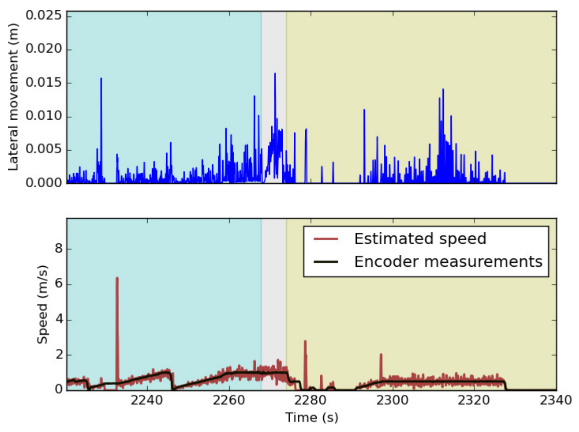
(a) First transition



(b) Second transition



(c) Third transition



(d) Fourth transition

Figure 3.26: The fictitious lateral movements introduced by the localization techniques and the estimated speed of the robot compared with the measurements of the encoders during the transitions. Light blue background represents outdoors, light yellow indoors and light gray transitions.

surements are mainly due to the corrections of the estimation. Thus, a good localization method will keep this magnitude small as a result of an accurate and continuous estimation. The EKF localization method uses a weighted mixture of the measurements and estimations from all the sensors involved. Thus, the quality of the estimation during the transition will be bounded by the quality of the measurements used. In the fourth transition shown in Figure 3.26d, the lateral movements (gray background) are comparable to the lateral movement of the indoor localization method (yellow background). Except for the outlier lateral movement of 0.35 m at the beginning of the second indoor-outdoor transition in Figure 3.26b and some other spurious data, the lateral movements during the transitions are smaller than 0.05 m.

The red and black lines in Figure 3.26 are, respectively, the norm of the speed vector of the robot measured as the variation in position over time and the advance speed of the robot gathered from the wheel encoders. Note that the red line can be described as a sum of the black line with some noise and, eventually, some outlier measurements. The noise is caused by the estimation uncertainty and also by the fact that the encoder measurements do not take into account the lateral movements of the robot, which is included in the estimated speed. Indeed, the variations in red are caused by the measurements from the GPS and the laserscan ranger to correct the accumulated error of the wheel encoders, which is not shown in the figure. The estimation during the transitions presents noise and some outlier measurements that are comparable to those obtained during the rest of the experiment.

Figure 3.27 shows the performance improvement of using a progressive weighting technique such as the one proposed here for the covariance intersection used during in-out transitions. During the first and the second transitions, the jumps in the localization estimation are smaller using a progressive weighting than using a fixed weight, so the accumulated fictitious lateral movement is smaller. This is due to the fact that the progressive weighting method assigns very low coefficients to the GPS estimation during the transition. The third transition does not present such a big difference between the two weighting policies. This is because there is another weighting factor, the covariance matrices, implicit in equations (3.10) and (3.11).

The varying weight method produces the biggest jump at the end of the transition, when the indoor estimation is discarded and only the GPS measurement is used. This is caused by the error during the geolocalization of the door which makes the indoor and outdoor estimations mutually incoherent. In contrast, the fixed weight method does not suffer these jumps only at the end of the transitions but during the whole process as the result is a mixture from the estimations. However, as the first estimations from the GPS after passing through the door are generally wrong, these poor measurements increase the jumps in the localization during the transition.

The last case is a special case where the robot was receiving very poor GPS measurements, that where barely compatible with the estimation of the map. That is the reason why, in that situation, the varying weight during the transitions jumped in a wrong direction that, afterwards, once the transition was finished, was corrected.

Figure 3.28 shows the localization uncertainty throughout the whole experiment. It can be seen that the values are always bounded. In general, the peak values of *theta* outdoors correspond to situations where the robot is turning, thus there are no orientation estimations computed and the uncertainty grows. Two special cases occur. The upper values of *theta* in the first outdoor interval refer to a situation where the robot is turning at a very low speed (top left circled navigation zone of Figure 3.25a). The uncertainty grows because we cannot compute an estimation but it remains bounded, as a consequence of the observability property of our system. The second case is in the last outdoor interval, where the robot is stopped for a while. Then, the uncertainty keeps growing and it is not limited because our system is not observable during this period. When the robot starts moving again a new estimation of the orientation can be calcu-

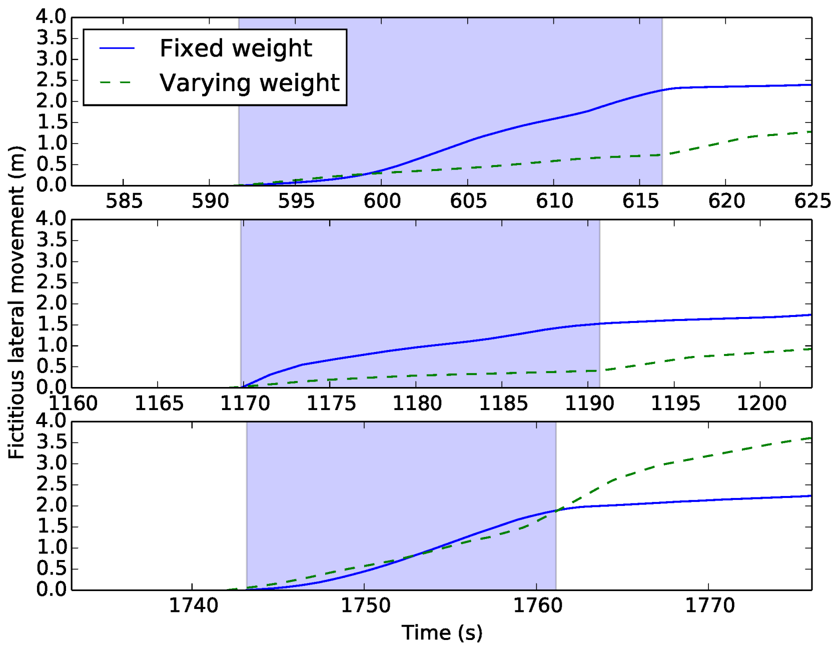


Figure 3.27: Comparison of the accumulated localization jumps during in–out transitions with and without progressive weighting. Blue background represents the transition gap.

lated and the uncertainty is reduced. There are some peaks in the uncertainty while the robot is indoors. These are caused by the higher uncertainty of the localization when the robot is turning. But these peaks are very short and the uncertainty goes back to normal levels as soon as the robot stops rotating.

As a result of the integration of different techniques in our seamless filter, there is a significant difference in the resulting values in each zone (outdoors/indoors), which is a consequence of the localization method used in each particular zone.

Thus, in light of these results, we conclude that our seamless localization method provides a continuous localization and high quality of estimation in a large and changing scenario during a long experiment.

Navigation and Maneuvering

In this section we analyze the performance of the proposed NDW navigation method and the reactive maneuvering system. Figure 3.29 shows the linear speed and the steering angle of the robot in the $Park$ scenario corresponding to the trajectory plotted in Figure 3.25. At first sight, the robot generally reaches the maximum speed predefined for each zone, 1 m/s outdoors and 0.5 m/s indoors (shadowed zones).

In a detailed view, we can distinguish different maneuvers and events during the experiment. For instance, between time 100 s and time 200 s, the robot performs a turning maneuver which corresponds to the maximum steering angle 0.4 rad and the

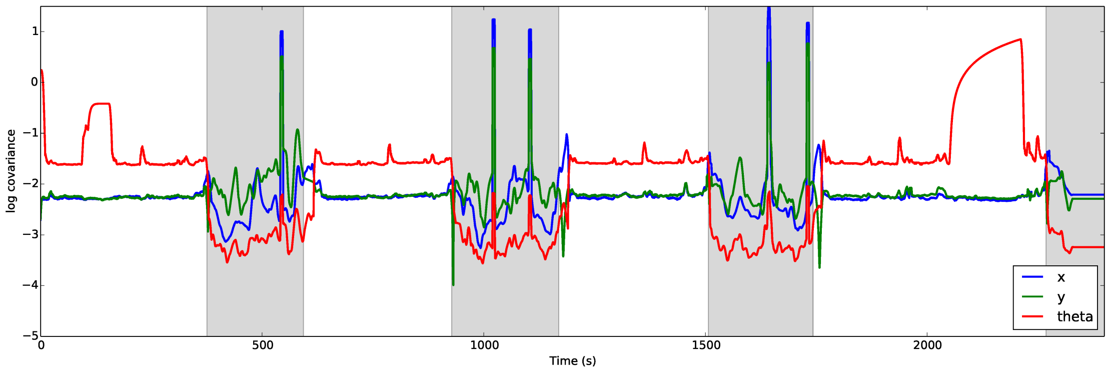


Figure 3.28: Logarithm-scaled covariance of the state variables during the full experiment in meters for variables x and y and radians for *theta*. Light gray zones represent indoor zones.

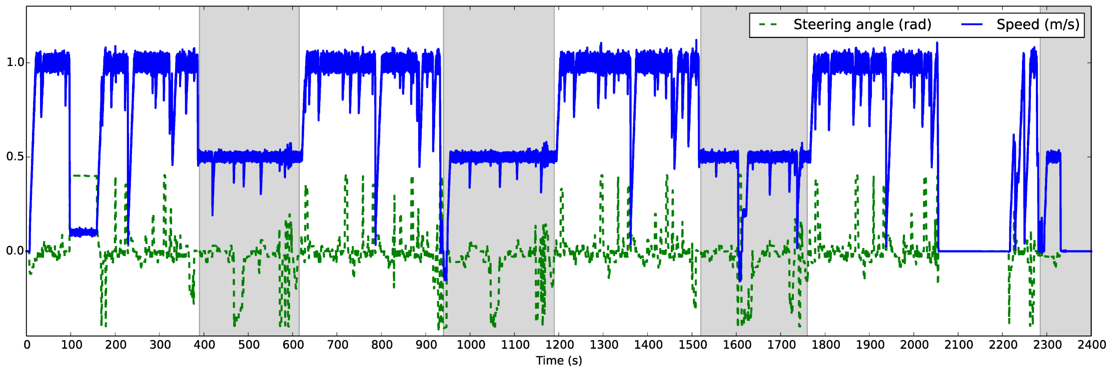


Figure 3.29: Speed and steering angle profiles during the full experiment in the *Park* scenario. Shadowed zones correspond to indoor zones.

maneuvering speed 0.1 m/s. This maneuver is also marked in Figure 3.25a with the top left circle. It was forced by the presence of a moving obstacle that blocked the planned trajectory and thus the robot was forced to move to a position where the next goal was unreachable without maneuvering. There are also two examples of avoidance maneuvers caused by unexpected obstacles at instants 950 s and 1600 s where the robot was forced to move backwards, represented by negative speed values, to pass by them. The first of them is marked in a circle in Figure 3.25b.

There was only one human intervention in the robot control in the whole experiment from time 2050 s to 2200 s caused by the presence of a human-driven car maneuvering in the path of the robot. The robot stopped automatically when the obstacle was perceived but it was manually kept still until the maneuvering of the car had finished. The robot then resumed the mission autonomously.

These results show that the navigation method proposed is robust to unexpected obstacles and that the maneuvering system is able to deal with different situations in a reactive fashion with successful results. A video of the robot navigating in the *Park* scenario[1] and a video showing the details of the transitions[2] are available.

3.5.6 Lessons Learned

In this section, we present some valuable lessons we gathered from the experience of performing experiments with the robot in warehouses.

The high degree of similarity in warehouse scenarios, with long indistinguishable corridors, let different localization hypotheses to be simultaneously acceptable. The problem is increased if the function for evaluating the hypotheses, the weighting function of the particles in the case of the AMCL algorithm used here, is based on simple scan matching techniques. The consequence is that the estimation of the localization of the robot may jump between different hypotheses which are very far from each other. Although the parametrization of the indoor localization method enables us to minimize the multi-hypotheses problem, the system architecture we propose also permits choosing a different technique such as [SASL13] or [GN15] for indoor localization that alleviates this problem.

For outdoor zones, our proposal relies mainly on the GPS sensor and it is therefore critical to have good GPS coverage. As a consequence, the base station of the differential GPS system must be carefully placed so that the zone covered with the highest GPS quality is maximized. Moreover, GPS measurements are used to geolocalize the door of the warehouse, defining the spatial relation between the indoor map-based localization with the outdoor localization. Thus, to increase the accuracy of the localization during transitions, the quality of the GPS measurements for setting up the LOCAL reference frame must be as high as possible. However, due to the fact that this is a one time task, the GPS measurements for geolocalizing the door can be replaced by a probably costly topographic technique that provides a more precise and reliable estimation.

We have found that the differentiation of the scenario in indoor and outdoor zones is not only important for localization purposes. Many subsystems of the robot are affected by the fact that it is in an indoor or outdoor zone. For instance, the characteristics of the scenario dramatically change from one zone to another. As can be seen in Figure 3.25a, the corridors outdoors are significantly wider than the corridors indoors and thus, for a safe autonomous navigation, the speed must be adapted to the width. So the detection of the change of zone can be used to modify the parametrization of the navigation to adapt the maximum speed or the safety distances to the scenario.

As regards navigation, we can conclude from our experience, that the meaning of *obstacle avoidance* dramatically changes from one application to another. It is clear that the main task of obstacle avoidance is to prevent the robot from crashing into elements in the environment, thus any obstacle avoidance technique must provide this capability. However, besides this safety requirement, the definition of the expected behavior of a robot avoiding an obstacle depends on the scenario and application. When the robot finds an obstacle blocking the desired path to the goal, there are different approaches

[1] `https://youtu.be/viueseDp1pg`
[2] `https://youtu.be/vfQfohiRAgg`

to face the problem. One option can be to slow down and stop the robot, wait for the obstacle to move and then resume the movement as planned. Another option is to let the robot bypass the obstacle and return to the planned path once the obstacle is overtaken. A third option would be to recalculate a new optimal path to the goal taking into consideration the new obstacle encountered. The requirements of the scenario determine the design of the obstacle avoidance method so there is no universal technique valid for every application. In our case, the proposed obstacle avoidance method tries to bypass the obstacles encountered and, in very complex situations, a replanning is performed to reach the goal. However, in some scenarios where more predictable autonomous robots are preferred, it might be worth stopping the robot when an unexpected obstacle is found. An example of the complexity of defining a correct behaviour during navigation was mentioned above, when the robot was manually stopped while another vehicle, driven by a human, was maneuvering. In addition to obstacle avoidance, a protocol between vehicles is needed to solve these situations.

We can also extract some lessons from our experience in this kind of warehouses construction material. They are not as structured, clean and rigid as those designed for picking and packing, which are suitable for landmark based solutions. It is also crucial for the application the interaction between robot and humans. Robots are easier to deploy in spaces without human presence because people are source of unpredictable events for the robot. These events affect many of the tasks of the robot. For instance, humans cause changes in the environment that are not critical for their performance but they may be decisive for the robot localization and mapping. Also navigation is affected if free zones are not respected.

3.6 Conclusions

This chapter describes a seamless localization technique for indoor-outdoor environments and a reactive navigation method. This system has been applied in an autonomous robot deployed in logistics scenarios.

A unified framework for continuous localization in large environments is presented. The most suitable and accurate sensors in each moment are integrated to provide a continuous localization without discontinuities in the estimations and having a limited uncertainty, even when transitions indoors-outdoors or vice versa occur. A progressive weighting during transitions based on the quality of the measurements of the different sensors allows a smooth integrated estimation. The proposed localization framework is open to a variety of indoor localization methods that can fit the requirements of the environment.

Regarding navigation, we have developed a technique for car-like vehicles that makes use of the advantages of two well known reactive navigation techniques but avoids their drawbacks. The Dynamic Window (DW) and Nearness Diagram (ND) techniques have been integrated and adapted in order to comply with the kinodynamic and shape constraints of the kind of robots considered.

The whole system integrating localization and navigation has been evaluated in real large-scale environments. Tests in different conditions have been carried out and conclusions reached about the behavior and limitations of the system. Deadlocks during motion only appear when very narrow passages for maneuvering are encountered in the trajectory which have not been detected at the long-term global planning time in changing scenarios.

As stated above, there are many solutions to the problem of obstacle avoidance. The kind of scenario considered so far is static, but obstacles appear occasionally in the environment. The solution proposed considers that the robot follows the path computed by a global planner and reacts to unexpected obstacles by temporarily modifying the path to avoid the obstacle. However, this solution may result in unsuccessful navigation performance if there are many obstacles moving around. In such scenario, planners for static environments and purely reactive avoidance techniques are no longer valid, because the dynamism of the environment is not considered. Taking the motion of the obstacles into consideration for planning allows obtain a better navigation behavior. Therefore, the rest of this thesis focuses on the navigation issue in dynamic environments, and it is assumed that an accurate localization estimation is always available during navigation, applying any existing localization technique, as the one developed in this chapter for example.

Chapter 4

Modeling the dynamic environment

The previous chapter introduced an autonomous robotic system deployed in an eminently static environment. Whenever a dynamic obstacle appears in the environment, the obstacle avoidance module modifies the path to avoid the obstacle. Then, the robot returns to the planned path.

However, in highly dynamic environments, where most of the obstacles move around, it is essential to explicitly consider the obstacle's motion in order to compute a safe trajectory towards the goal. This chapter focuses on the definition of a model that takes into account the future behavior of the obstacles and the kinodynamic constraints of the robot, mapped in the robot's Control space, which can be used for planning motions. Chapters 5 and 6 will be devoted to explain the planning and navigation techniques developed based on this modeling approach.

The work presented through this chapter is published in an article [LOM18] in the International Journal of Robotics Research.

4.1 Introduction

Representing the environment into the robot's Control space allows to compute motions (velocities) straightforwardly. This chapter defines a new model representing the dynamism of the environment which contains information about the motion evolution of obstacles, *the Dynamic Object Velocity-Time Space* (*DOVTS*). This work extends and formalizes the model of a preliminary work [OM05]. A precomputed set of trajectories in the Configuration space is represented as a set of velocity commands in *DOVTS*. It will contain free and collision-leading velocities within the space horizon available, for example the sensor's field of view. The *DOVTS* model exploits the concept of *estimated arriving time* of an object to compute the times and velocities for future potential collisions. As a result, several velocity obstacles will be represented in the Control space of the robot.

Section 4.2 establishes some remarks regarding the type of robot under consideration. Section 4.3 formally defines the *DOVTS* model for different kinds of obstacle trajectories and static obstacles. A complexity analysis of the model and a comparison with respect to other well-known techniques in this field closes the section. Finally, section 4.4 presents the conclusions of the chapter.

4.2 Problem Statement

In order to compute the free Control space, circular robot trajectories for non-holonomic robots have been selected for the long-term planning. This is not a strong constraint because, as it will be seen later, the planner is launched every sampling time, leading to different kind of trajectories that drive the robot quickly towards the goal. The circular trajectory initially planned is only applied during the next sampling time, and it is replanned in the following periods.

We consider a non-holonomic robot $\mathcal{R}$ moving in a dynamic environment, where it has to safely reach its goal avoiding collisions with the static and moving objects $\mathcal{O}_i$ around it. They all share workspace $\mathcal{W} = \mathbb{R}^2$. Let $\mathcal{O}(t)$ be the set of points in $\mathcal{W}$ occupied by all the obstacles $\mathcal{O}_i$ at instant t, i.e., $\mathcal{O}(t) = \cup \mathcal{O}_i(t)$.

The state of the robot is defined by its position and orientation at instant t, $\mathcal{R}_t = (x, y, \theta)$. Let $\mathcal{R}(\mathbf{v}, t)$ denote the state reached by the system $\mathcal{R}$ under the action of control $\mathbf{v} = (\omega, v)$ applied at time t. In our case, the controls are the angular (ω) and linear (v) velocity for a differential-drive robot. The motion model for the robot can be expressed by the well-known equations 4.1.

$$\dot{x} = v * cos(\theta); \quad \dot{y} = v * sin(\theta); \quad \dot{\theta} = \omega \tag{4.1}$$

From these states and motion model we describe in the next section the *DOVTS* environment model.

4.3 Environment modeling in the control space

The model is constructed by mapping the information from the workspace to the control space of the robot, denominated *Velocity Space* ($\mathcal{V}$), which is the set of velocities (ω, v) reachable by the robot limited by the maximum and minimum velocity constraints. This space also contains sets of velocities leading to future collision if the robot executes them. In some way these collision velocities represent obstacles mapped into $\mathcal{V}$, so we name them *Dynamic Object Velocity* (*DOV*).

The basic concept to build the model is the *time of collision* between robot and obstacle trajectories. The intersection points are computed from a robocentric representation, resulting in associated information of time and velocity that indicate the instants at which an obstacle reaches them, and the velocities for the robot to pass before colliding or after the object passes, avoiding collision. These time-velocity data are then transferred to the control space of the robot to represent the *Dynamic Object Velocity-Time* (*DOVT*) obstacles in the environment, conforming the *Velocity-Time Space* ($\mathcal{VT}$) of the robot.

In this work, for the sake of clarity, the method is presented for linear and circular obstacle motion, but other kind of obstacle trajectories could be used, without loss of generality. The information mapped both in $\mathcal{V}$ and $\mathcal{VT}$ can be used to compute commands without collision for the robot. The techniques developed will be covered through Chapters 5 and 6, respectively.

One of the advantages of this technique is that the model maps the safe robot velocities for a space horizon only limited by the field of view of the onboard sensors. This allows motions to be planned at every instant for the whole time horizon corresponding to the space horizon and not only for the next sampling period in a purely reactive way. Additionally, due to computational burden or to practical navigation issues, this space horizon could be artificially limited, enabling reasoning about the behaviour of only the closest obstacles, which are the ones that will impose the immediate manoeuvring decisions.

4.3.1 The Dynamic Object Velocity-Time Space

In this section the *Dynamic Object Velocity-Time Space* (*DOVTS*) is defined. Appendix B contains the equations for computing the model for linear and circular obstacle trajectories together with a more detailed explanation of the computation process. Next, a brief description is provided.

The model results from computing the times of collision between different feasible robot trajectories and obstacle trajectories in a robocentric reference. Figures 4.1a – 4.1c show this idea. In Fig. 4.1a the workspace at a specific instant is depicted. The information in the workspace is mapped to a robocentric representation in the configuration space (Fig. 4.1b) to $\mathcal{V}$ (Fig. 4.2b). The robot is reduced to a point and the obstacles are enlarged with the radius of the robot. They are modeled as wrapping squares. It is also assumed that the future trajectory of each obstacle $\mathcal{O}_i$ is known or estimated.

In Fig. 4.1b the $\mathcal{CB}_i$ (*CollisionBand*) swept by an enlarged obstacle $\mathcal{O}_i$ moving in a straight line is depicted. A circular trajectory γ_j for the robot is also shown. Let P_{1j} and P_{2j} be the intersection points between γ_j and the outline of $\mathcal{CB}_i$. These points represent the collision points in $\mathcal{W}$ between $\mathcal{R}$ and $\mathcal{O}_i$, and will determine the opportunities for the robot to pass before (P_{2j}) or behind the obstacle (P_{1j}) by following trajectory γ_j. Then, the times (t_{2j} and t_{1j}) at which the obstacle reaches those points, $\mathcal{O}_i^1$ and $\mathcal{O}_i^2$, respectively, are computed. These times are the *estimated arriving times* of the object, which indicate *times of collision.* Thus, the minimum and maximum velocities that allow the robot to pass before or after the obstacle, respectively, are:

$$\begin{aligned}\omega_{kj} &= \frac{\theta_{kj}}{t_{kj}} = \frac{\text{atan2}\;(2\,x_{kj}\,y_{kj},\; x_{kj}^2 - y_{kj}^2)}{t_{kj}} \\ v_{kj} &= r_j\,\omega_{kj}, \quad k = 1, 2\end{aligned} \tag{4.2}$$

where θ_{kj} is the angular displacement on γ_j for the robot to reach $P_{kj} = (x_{kj}, y_{kj})$, $k = 1..2$. See Appendix B for details.

These calculations are extended to a discretized set of n feasible circular trajectories Γ in $\mathcal{W}$, so $\gamma_j \in \Gamma$, $j = 1..n$ (see Fig. 4.1c). Note that hereinafter, γ_j is used indistinctly to refer to both a circular trajectory and its radius. The computed velocities and times to

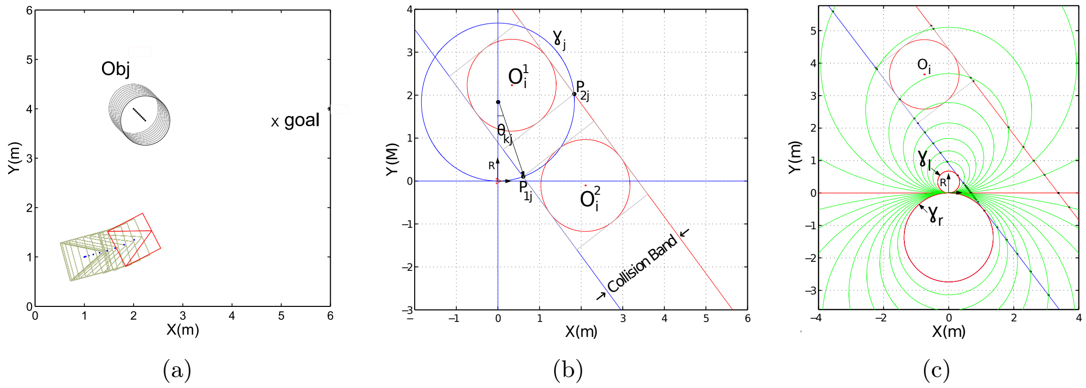


Figure 4.1: (a) Workspace; (b) Collision Band swept by object $\mathcal{O}_i$, object $\mathcal{O}_i$ in positions $\mathcal{O}_i^1$ and $\mathcal{O}_i^2$, path γ_j, and collision points P_{1j} and P_{2j} in the robocentric (R) Configuration Space; (c) multiple paths Γ through the collision band.

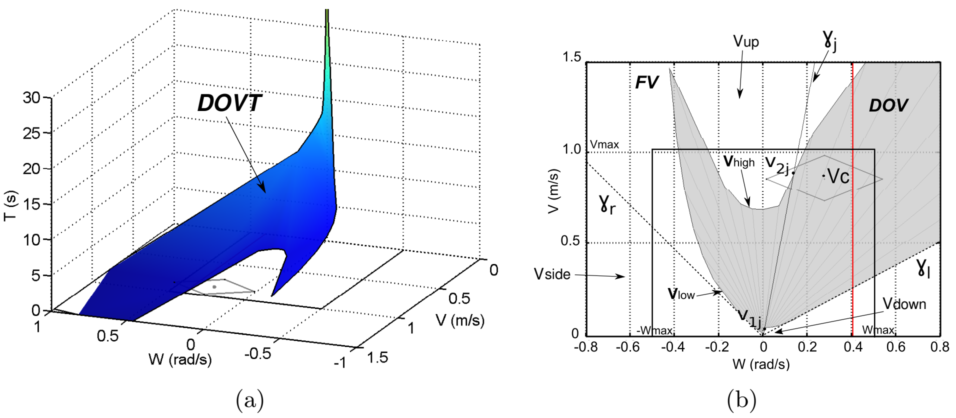


Figure 4.2: (a) Velocity-time space $DOVTS$ and velocity-time obstacle $DOVT$; (b) projection of $DOVTS$ on the plane (w, v), $DOVS$, and DOV (projection of $DOVT$).

collision for Γ are mapped in the *Velocity-Time Space.* Figure 4.2a shows the $\mathcal{VT}$ space of the robot including the resultant velocities and times of collision. These sets correspond to the limits of *Dynamic Object Velocity-Time* ($DOVT$), and the corresponding *Dynamic Object Velocity* (DOV) in $\mathcal{V}$ (Fig. 4.2b), which will be defined formally later. The velocities in between lead to a collision with the obstacle at some time. Note also that the circular paths are transformed into straight lines in the velocity space, with slope $\gamma_j = v_j/\omega j$, $\gamma_j \in \Gamma$, as shown in Fig. 4.2b. In Fig. 4.3 the robocentric representation of an obstacle following a circular trajectory and its corresponding DOV are plotted.

We now formally define the space for the robot and its characteristic variables.

Definition 1 (Dynamic Object Velocity-Time). *The Dynamic Object Velocity-Time (DOVT) for a particular moving object* $\mathcal{O}_i$ *with respect to the set of feasible trajectories*

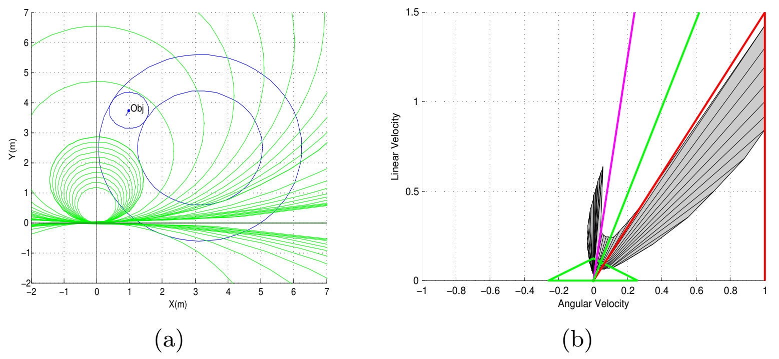


Figure 4.3: (a) Robocentric representation of an obstacle circular trajectory (blue), and in green a set of circular trajectories for the robot. (b) The corresponding model in *DOVS*.

Γ *is defined as the set of velocities that produce a collision with* $\mathcal{O}_i$ *at time* t*,*

$$DOVT(O_i,\Gamma) = \{(\omega, v, t) \ \in \ \mathcal{V} \times \mathbb{R} \mid \exists \gamma_j \in \Gamma, \frac{v}{\omega} = \gamma_j, \ \boldsymbol{v}_j = (\omega, v), \ \mathcal{R}(\boldsymbol{v}_j, t) \cap \mathcal{O}_i(t) \neq \emptyset\} \tag{4.3}$$

Definition 2 (**Dynamic Object Velocity**). *The Dynamic Object Velocity (DOV) for a particular moving object* $\mathcal{O}_i$ *with respect to the set of feasible trajectories* Γ *is defined as the projection of its DOVT, i.e., the set of velocities that produce a collision with* $\mathcal{O}_i$ *at any time,*

$$DOV(O_i,\Gamma) = \{(\omega, v) \ \in \ \mathcal{V} \mid \exists \gamma_j \in \Gamma, \ \frac{v}{\omega} = \gamma_j, \exists t \in \mathbb{R}, \ \boldsymbol{v}_j = (\omega, v), \ \mathcal{R}(\boldsymbol{v}_j, t) \cap \mathcal{O}_i(t) \neq \emptyset\} \tag{4.4}$$

Accordingly, we define $DOVT(\mathcal{O},\Gamma)$ and $DOV(\mathcal{O},\Gamma)$ (henceforth $DOVT$, DOV) with respect to all the obstacles in the environment,

$$DOVT(\mathcal{O},\Gamma) = \cup_{\mathcal{O}_i} DOVT(\mathcal{O}_i,\Gamma) \tag{4.5}$$
$$DOV(\mathcal{O},\Gamma) = \cup_{\mathcal{O}_i} DOV(\mathcal{O}_i,\Gamma) \tag{4.6}$$

Definition 3 (**Free Velocity-Time**). *The Free Velocity-Time (FVT) is the set of velocities outside DOVT,*

$$FVT = \{(\omega, v, t) \ \in \ \mathcal{V} \times \mathbb{R} \mid (\omega, v, t) \notin DOVT, \boldsymbol{v}_j = (\omega, v), \ \mathcal{R}(\boldsymbol{v}_j, t) \cap \mathcal{O}(t) = \emptyset\} \tag{4.7}$$

Definition 4 (**Free Velocity**). *The Free Velocity (FV) is the set of velocities outside DOV,*

$$FV = \{(\omega, v) \ \in \ \mathcal{V} \mid \boldsymbol{v}_j = (\omega, v) \notin DOV, \ \forall t \in \mathbb{R}, \mathcal{R}(\boldsymbol{v}_j, t) \cap \mathcal{O}(t) = \emptyset\} \tag{4.8}$$

Definition 5 (**Dynamic Object Velocity-Time Space**). *The Dynamic Object Velocity-Time Space (DOVTS) is defined as the control space of the robot which contains controls belonging to DOVT and FVT,*

$$DOVTS = \{DOVT \cup FVT\} \tag{4.9}$$

Definition 6 (**Dynamic Object Velocity Space**). *The Dynamic Object Velocity Space (DOVS) is defined as the control space of the robot which contains controls belonging to DOV and FV,*

$$DOVS = \{DOV \cup FV\} \tag{4.10}$$

Next, the definitions provided are with respect to *DOVTS*. However, they are straightforwardly applicable to its projection *DOVS*.

Definition 7. *Let $\boldsymbol{V}_{low}$ and $\boldsymbol{V}_{high}$ respectively be the set of maximum and minimum velocities and times computed for all paths in Γ (Fig. 4.2),*

$$\begin{aligned}\boldsymbol{V}_{low}(O_i,\Gamma) = \{(\omega,v,t) \in DOVT(O_i,\Gamma)\ |\boldsymbol{v}_j = (\omega,v) = \boldsymbol{v}_{1j},\ t = \boldsymbol{t}_{1j},\\ \mathcal{R}(\boldsymbol{v}_j,t) \cap \mathcal{O}_i(t) \neq \emptyset\}\end{aligned} \tag{4.11}$$

$$\begin{aligned}\boldsymbol{V}_{high}(O_i,\Gamma) = \{(\omega,v,t) \in DOVT(O_i,\Gamma)\ |\boldsymbol{v}_j = (\omega,v) = \boldsymbol{v}_{2j},\ t = \boldsymbol{t}_{2j},\\ \mathcal{R}(\boldsymbol{v}_j,t) \cap \mathcal{O}_i(t) \neq \emptyset\}\end{aligned} \tag{4.12}$$

Note that $\mathbf{V}_{low}$ and $\mathbf{V}_{high}$ represent the contour of the *DOVT* (*DOV*) for a particular object, i.e.,

$$(\omega,v,t) \in \mathbf{V}_{low}(\mathcal{O}_i,\Gamma) \leq (\omega,v,t) \in DOVT(O_i,\Gamma) \leq (\omega,v,t) \in \mathbf{V}_{high}(\mathcal{O}_i,\Gamma) \tag{4.13}$$

Definition 8. *Let $\boldsymbol{V}_{down}$, $\boldsymbol{V}_{up}$ and $\boldsymbol{V}_{side}$ be the sets of velocities and times in FVT eligible for the robot to move safely. In Fig. 4.2b, they represent respectively the velocity commands under $\boldsymbol{V}_{low}$, above $\boldsymbol{V}_{high}$, and below the bounds of DOV, respectively. Let γ_l, γ_r be the left most and right most radius shaping DOVT (DOV) in DOVTS (DOVS).*

$$\begin{aligned}\boldsymbol{V}_{down} = \{(\omega,v,t) \in FVT \mid \forall\gamma_j \in \Gamma,\ \frac{v}{\omega} = \gamma_j, \boldsymbol{v}_j = (\omega,v) < \boldsymbol{v}_{1j},\\ \forall t \in \mathbb{R},\ \mathcal{R}(\boldsymbol{v}_j,t) \cap \mathcal{O}(t) = \emptyset\}\end{aligned} \tag{4.14}$$

$$\begin{aligned}\boldsymbol{V}_{up} = \{(\omega,v,t) \in FVT \mid \forall\gamma_j \in \Gamma,\ \frac{v}{\omega} = \gamma_j, \boldsymbol{v}_j = (\omega,v) > \boldsymbol{v}_{2j},\\ \forall t \in \mathbb{R}, \mathcal{R}(\boldsymbol{v}_j,t) \cap \mathcal{O}(t) = \emptyset\}\end{aligned} \tag{4.15}$$

$$\begin{aligned}\boldsymbol{V}_{side} = \{(\omega,v,t) \in FVT \mid \forall\gamma_j \in \Gamma,\ \frac{v}{\omega} = \gamma_j \notin [\gamma_l,\gamma_r], \boldsymbol{v}_j = (\omega,v),\\ \forall t \in \mathbb{R},\ \mathcal{R}(\boldsymbol{v}_j,t) \cap \mathcal{O}(t) = \emptyset\}\end{aligned} \tag{4.16}$$

From these definitions we can obtain some important properties which must be used by any planner based on this model to select safe robot commands ensuring collision avoidance in a dynamic environment.

Property 1. *Selecting velocity commands in $\boldsymbol{V}_{down}$ yields no collision trajectories. Formally,*

$$\forall \boldsymbol{v}_j \in \boldsymbol{V}_{down} \Rightarrow \mathcal{R}(\boldsymbol{v}_j, t) \cap \mathcal{O}(t) = \emptyset, \forall t \in \mathbb{R}.$$

Proof. By definition of $\mathbf{V}_{down}$, if $\mathbf{v}_j \in \mathbf{V}_{down}$, then $\mathbf{v}_j < \mathbf{v}_{1j}$, where $\gamma_j = \frac{v_j}{w_j} = \frac{v_{1j}}{w_{1j}}$. Then, $\exists t \in \mathbb{R}$ such that $\mathcal{R}(\mathbf{v}_j, t) = P_{1j}$, the intersection point with the collision band. Given that $\mathbf{v}_j < \mathbf{v}_{1j}$, then $t > t_{1j}$. At time t the object has passed P_{1j}, and a collision cannot be produced on γ_j. $\square$

Property 2. *Selecting velocity commands in $\boldsymbol{V}_{up}$ yields no collision trajectories. Formally,*

$$\forall \boldsymbol{v}_j \in \boldsymbol{V}_{up} \Rightarrow \mathcal{R}(\boldsymbol{v}_j, t) \cap \mathcal{O}(t) = \emptyset, \forall t \in \mathbb{R}.$$

Proof. By definition of $\mathbf{V}_{up}$, if $\mathbf{v}_j \in \mathbf{V}_{up}$, then $\mathbf{v}_j > \mathbf{v}_{2j}$, where $\gamma_j = \frac{v_j}{w_j} = \frac{v_{2j}}{w_{2j}}$. Then, $\exists t \in \mathbb{R}$ such that $\mathcal{R}(\mathbf{v}_j, t) = P_{2j}$, the intersection point with the collision band. Given that $\mathbf{v}_j > \mathbf{v}_{2j}$, then $t < t_{2j}$. At any time before t_{2j} the object has not yet arrived at P_{2j}, and a collision cannot be produced on γ_j. $\square$

Property 3. *Selecting velocity commands in $\boldsymbol{V}_{side}$ yields no collision trajectories. Formally,*

$$\forall \boldsymbol{v}_j \in \boldsymbol{V}_{side} \Rightarrow \mathcal{R}(\boldsymbol{v}_j, t) \cap \mathcal{O}(t) = \emptyset, \forall t \in \mathbb{R}.$$

Proof. If $\mathbf{v}_j \in \mathbf{V}_{side}$, then $\frac{v_j}{w_j} = \gamma_j \notin [\gamma_l, \gamma_r]$, by definition of $\mathbf{V}_{side}$. Thus, no collision could be produced as the resultant trajectory γ_j does not intersect the bounds of the $\mathcal{CB}_i$ at any time. $\square$

Property 4. *Selecting velocity commands under DOVT yields collision trajectories. Formally,*

$$\forall \boldsymbol{v}_j \in DOVT \Rightarrow \exists t \in \mathbb{R},\ \mathcal{R}(\boldsymbol{v}_j, t) \cap \mathcal{O}(t) \neq \emptyset.$$

Proof. If $\mathbf{v}_j \in DOVT$, then $\mathbf{v}_j > \mathbf{v}_{1j}$ and $\mathbf{v}_j < \mathbf{v}_{2j}$, by definition of $DOVT$. Then, $\exists t_1, t_2 \in \mathbb{R}$ such that $\mathcal{R}(\mathbf{v}_j, t_1) = P_{1j}$ and $\mathcal{R}(\mathbf{v}_j, t_2) = P_{2j}$. Given that $\mathbf{v}_j > \mathbf{v}_{1j}$, then $t_1 < t_{1j}$, which means that at time t_1 the object is still at P_{1j} and a collision is produced. Likewise, given that $\mathbf{v}_j < \mathbf{v}_{2j}$, then $t_2 > t_{2j}$, which means that at time t_2 the object has already reached P_{2j} and a collision is produced. Then, a collision could be produced sometime within the interval $t \in [t_{2j}, t_{1j}]$. $\square$

Property 5. *Selecting velocity commands in DOV yields collision trajectories. Formally,*

$$\forall \boldsymbol{v}_j \in DOV \Rightarrow \exists t \in \mathbb{R},\ \mathcal{R}(\boldsymbol{v}_j, t) \cap \mathcal{O}(t) \neq \emptyset.$$

Proof. DOV is the projection of $DOVT$ in the plane (ω, v). Thus, the same conditions as in Property 4 are fulfilled. $\square$

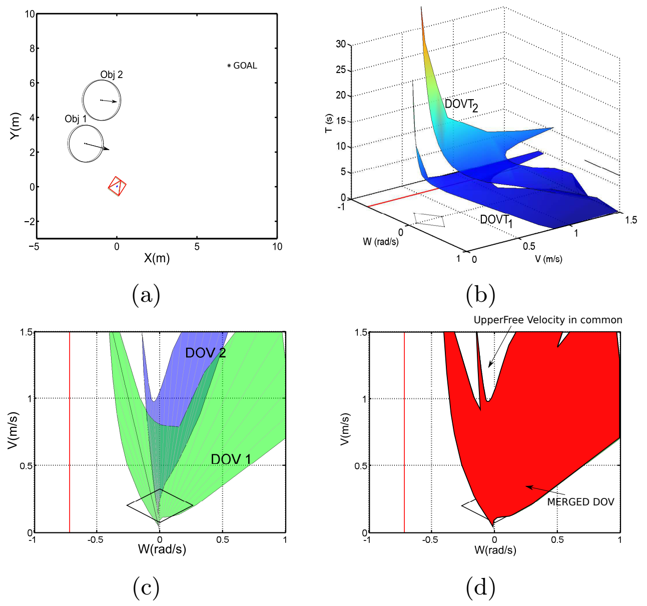


Figure 4.4: (a) A situation with two moving objects near to each other in $\mathcal{W}$. (b) The situation represented in *DOVTS*. (c) Projection of both moving objects in *DOVS*. (d) The merged *DOV* object.

Summarizing the properties, when a velocity command is chosen out of *DOVT* or *DOV*, collision avoidance is guaranteed as long as the command is maintained and the object motion hypotheses are met. Considering the 2D projection *DOVS*, commands can be safely changed every sampling control period, as long as they remain outside *DOV*. In the event that a sudden unforeseen change of an object motion results in the current velocity being inside a *DOV*, it still could be possible, time permitting, to escape collision by correctly choosing subsequent velocity commands. By using *DOVTS*, the robot can use the set of free commands under *DOVT* surface to compute a motion, which increases the robot maneuverability with respect to using *DOVS*, in which only velocities out of any *DOV* are permitted to ensure collision avoidance.

4.3.2 Dealing with multiple objects in *DOVTS*

Figure 4.4 shows a situation with two objects in $\mathcal{W}$, *DOVTS* and *DOVS*. In the *DOVTS* space, moving objects are represented as their corresponding *DOVT* surfaces. Clearly, the highest surface in Fig. 4.4b corresponds to the farthest or the slowest object with respect to the robot, and the lowest surface to the nearest or the quickest obstacle. Working directly in *DOVTS* would allow to utilize the velocity-time room between both surfaces for maneuvering among the objects. Figure 4.4c shows the projection of both surfaces in *DOVS*. When this space is used to plan trajectories, the *DOV* objects can be geometrically merged to obtain one compound *DOV* object (Fig. 4.4d). Reasoning in this space to compute the motion commands in the free velocity space is made using the

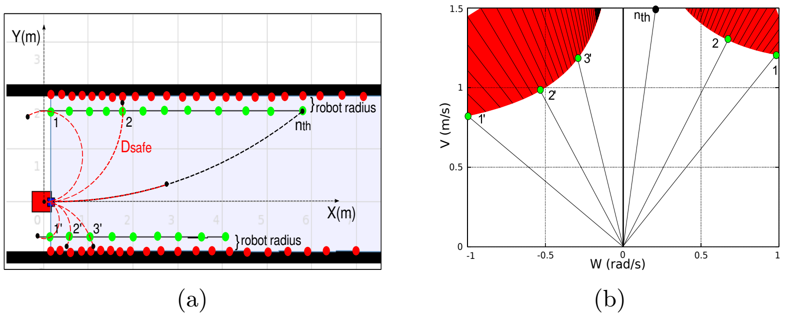


Figure 4.5: (a) Mapping a corridor into *DOVS*. The red points correspond to scan laser points for example, and the green points are the points after extending the walls with the robot radius. (b) The corridor mapped into *DOVS* showing the maximum velocity reachable for the robot on each circular path considered.

merged object, which will be explained in Chapter 5. This leads to more conservative navigation decisions than if the whole *DOVTS* was used, which is the objective in Chapter 6.

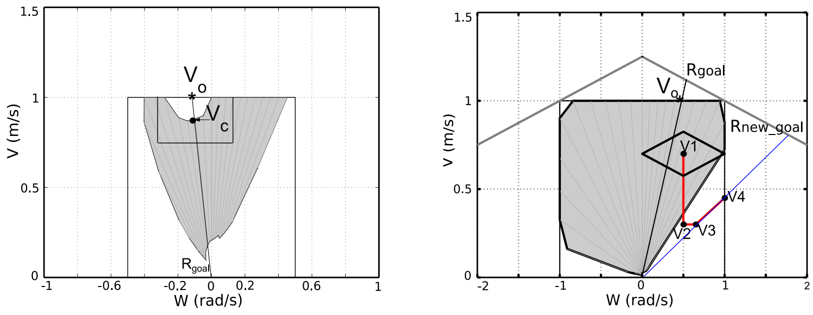
4.3.3 Static Objects in *DOVTS*

Static obstacles can also be represented in *DOVTS*. Let D_{safe} be the displacement the robot requires on a path to reach zero speed from its current velocity. If the distance to a static object is greater than D_{safe}, then the circular path in *DOVTS* is mapped as free of collision. Otherwise, the maximum velocity that the robot should have to stop before a collision, $\mathbf{v}_{stop}$, is calculated and mapped into *DOVS*. This velocity should be forbidden at any time in *DOVTS*. Any higher velocity is mapped as a collision one. Figure 4.5 depicts the result of mapping a static corridor-like object into *DOVS*.

4.3.4 The Kinodynamic Constraints in *DOVTS*

The maximum linear and angular velocity constraints are explicitly represented in the model by the external square (omnidirectional robot) or the external rhombus (differential drive robot) in Fig. 4.6. So far, it has been assumed that any $\mathbf{v}_o$ could be reachable in one sampling step from the current velocity, which depends on the robot's maximum acceleration. The dynamic constraints of the robot define the velocity window (VW) centered in its current velocity $\mathbf{v}_c$ and bounded by the acceleration constraints (a_m, α_m). That is, $(v_c \pm a_m \Delta t, w_c \pm \alpha_m \Delta t)$, the inner rectangle and rhombus in Fig. 4.6a-b, respectively. The shape of this window also represents the kinematic constraints corresponding to the type of the robot model.

Figure 4.6a shows that $\mathbf{v}_o$ can only be reached in one step from the velocities inside VW, due to the dynamic constraints. In Fig. 4.6b it can be seen that $\mathbf{v}_o$ cannot be reached in one step from $\mathbf{v}_1$. To deal with this problem, we compute first the number of sampling periods needed to reach the limits $\mathbf{V}_{low}$ and $\mathbf{V}_{high}$ from the current velocity in $DOVT(DOV)$, adding them to $\mathbf{t}_{low}$ and $\mathbf{t}_{high}$ to obtain $\mathbf{t}'_{low}$ and $\mathbf{t}'_{high}$ respectively.



(a) (b)

Figure 4.6: Kinodynamic constraints: (a) holonomic robot, $\mathbf{v}_o$ is reachable only inside VW, a square window; (b) non holonomic robot, $\mathbf{v}_o$ is not reachable within VW, a rhombus window. Circular, clothoid and anti-clothoid trajectories are shown in red. R_{new_goal} represents a new radius, computed in the free zone, to be followed until the true R_{goal} becomes free.

These times are used to recompute the new velocity limits ($\mathbf{V}'_{low}$, $\mathbf{V}'_{high}$) using equations 4.2.

4.3.5 Trajectories in *DOVTS*

Different kinds of trajectories can be selected to execute the motion plan and maneuver the robot. In this work we consider bounded velocity and acceleration for real robots, and apply extremal actions (maximum acceleration or deceleration) motivated by results presented in [RP94], introduced in section 2.2. As described in previous work by [OM05], this type of control action yields rectilinear, clothoid (only angular acceleration) and anti-clothoid (only linear acceleration) paths, allowing a curvature continuity. Moreover, as shown in section 4.3.1, circular and linear trajectories are used for motion planning and execution.

These mentioned trajectories are directly mapped on *DOVS* (see Figure 4.6b). Linear trajectories in the Workspace appear as a straight line in the v axis ($\omega = 0$). Circular paths are mapped as straight lines with a slope, which represent different radii ($\gamma = v/\omega$) ($v_3 - v_4$ red line). Clothoids are depicted as straight horizontal lines ($v = constant$) ($v_2 - v_3$ red line), and anti-clothoids as vertical lines ($\omega = constant$) ($v_1 - v_2$ red line).

4.3.6 Complexity of modeling algorithms

A comparison is performed to analyze the complexity of the algorithms presented for modeling the dynamic environment and its use for planning in other techniques proposed in the literature, specifically those based on *VO* and *ICS*, with respect to our proposal.

Computing the set of *ICS* of a robot for different control trajectories and several obstacles involves a complexity of $\mathcal{O}(mnt_{max}o)$, where m is the number of obstacles, n the number of evasive maneuvers, t_{max} a look-ahead for practical use of the approach and o the sum of vertexes of the robot and the most complex object, which are involved in the

Minkowski difference operation required between each pair robot-obstacle. Parameter t_{max} is the number of sampling periods, with a duration of Δt each, considered for computing the *ICS* set, and thus the total time of such states being *ICS*. It is assumed that the workspace is bounded, and the robot and obstacles will exit it at t_{max}. By using Graphics Processing Unit (GPU) the complexity can be reduced to $\mathcal{O}(mnt_{max})$ (see [MG10]).

The *VO* approach concerns mainly three operations: computing each individual VO_j for $j = 1..m$ obstacles, the multiple velocity obstacle *MVO*, and the set of safe velocities. The most complex operation is the second one, because it requires updating the points of each VO_i, $i = j + 1..m$, with the intersection points between VO_j and each VO_i, and ordering them clockwise, which leads to a complexity of $\mathcal{O}(m^2)$. This can be reduced to $\mathcal{O}(m \log m)$ by storing the points within a more efficient structure (see [Fio95]). The need for a t_{max} parameter to consider a look-ahead for the obstacles around is also present in this approach.

In our technique, the complexity of modeling each moving object in its DOV_j, $j = 1..m$, is $\mathcal{O}(mn)$, where m is the number of obstacles and n the maximum number of the trajectories considered (Γ). The set of trajectories selected for computing a DOV_j depends on the position of the obstacle and its motion with respect to the robot, and on the desired discretization. The calculation for collision times and forbidden velocities is therefore focused on the area where the obstacle will be moving. Thus, several robot trajectories are considered for each obstacle. Furthermore, the complexity for mapping objects moving in linear and non-linear trajectories (in particular circular paths have been developed in this work) is the same, because only intersection points with the collision band are needed, not altering the computation burden. This approach provides enough information for computing plans in the defined workspace horizon, and it is not affected by the parameter t_{max} of the other methods given that the time for which obstacles are considered is implicit within the model construction.

4.4 Conclusions

A *DOVTS* model to represent the dynamism of the environment in the Control space of the robot is developed. Any motion planner existing in the literature that uses the Velocity space as a Control space can benefit from the model herein defined. The following Chapters present the planning and navigation techniques proposed by the author by using the free velocity space, exploiting the properties from *DOVTS* and its projection *DOVS* on the Velocity plane (ω, v). Chapter 5 bases on *DOVS* to define a set of strategies for planning and navigation that optimizes the time to reach a goal in the presence of moving obstacles. The approach in Chapter 6 reasons on *DOVTS*, which allows to obtain a higher maneuverability for the robot with respect to *DOVS*, at the cost of increasing the computation time.

Chapter 5

A model-based robocentric planner on *DOVS*

Having a proper definition of the environment is a key issue for making navigation decisions. The previous Chapter introduced the description of the *DOVTS* model, which takes into account the motion evolution of the obstacles and the kinodynamic constraints of the robot on the Control space so that a safe and feasible motion towards the goal can be computed. The model contains information about eligible and forbidden velocities, as well as the times at which a collision may occur.

The work developed here addresses a new robocentric technique of motion planning and navigation for differential-drive robots in dynamic environments, by using the data mapped on the projection *DOVS*. The method considers safety and maneuverability to plan near time-optimal motions within the visibility space horizon available, not only for the current sampling period. A general planning strategy is built upon the ideas of avoiding moving obstacles by slowing down to let them pass or speeding up to overtake them.

These results are published in an article [LOM18] in the International Journal of Robotics Research.

5.1 Introduction

As already mentioned, considering the motion evolution of the obstacles for planning motions in dynamic environments is paramount. The *DOVS* model captures the dynamism of the environment and the robot kinodynamic constraints in a horizon within the sensor's field of view on-board the robot. Having this greater visibility horizon at the current instant allows the decision process to select the best motion from an improved and informed search, without needing to simulate the motion several steps ahead. However, only one velocity command is applied during the next sampling period, and a new plan is computed in order to adapt to new situations in case hypotheses are no longer satisfied. Recomputing the plan every sampling time yields clothoid and anti-clothoid trajectories, providing continuous curvature trajectories.

Section 5.2 explains the decision making approach, and the planning and navigation strategies. Through this section, a strategies-based approach to compute safe motion commands (velocities) for the robot is developed, by using the *DOVS* model. Navigation decisions are made based on identifying a *situation* among a discrete set, evaluating the *decision variables* computed from the model of the environment *DOVS*. Section 5.3 evaluates the method in randomly generated simulated scenarios, based on metrics defined using safety and time-to-goal criteria. An evaluation in real-world experiments is also presented. In section 5.4 some conclusions and future work are exposed.

5.2 Decision making for planning and navigation

This section explains the decision making process based on *DOVS*. The characteristic properties derived in 4.3.1 with respect to *DOVS* will serve as the foundation for the technique developed.

The problem is similar to a situation in which a pedestrian intends to cross a road while cars are passing. The pedestrian must decide how to do this safely. In such a situation, the main decision the robot must take is whether to pass before or after the obstacle.

Most of the proposed techniques for navigation in dynamic environments limit the number of possible robot trajectories or maneuvers, or simply slow down or stop the robot to avoid a collision by allowing the moving obstacle to pass first. The method presented in this work enables lighter computation of long-term safe command sequences in the free-velocity space, and thus the corresponding trajectories to the goal. These planned trajectories are not time-optimal, but are used as a first seed to guide the robot towards the goal in the free space. In the following steps, the recomputed extremal control commands will lead the robot to move in near time-optimal trajectories to the goal.

The design is based on the paradigm of situated activity ([Ark98]). The robot decides the motion plan within a set of finite but general situations identified from the perception system. Each situation has an associated motion strategy, which exploits the information provided by the model of the environment. Two main criteria are considered to define the strategy: i) the trajectories generated have to be safe; ii) near time-optimal trajectories to the goal have to be generated. The first is ensured by selecting velocity commands inside the free-velocity FV of the control space, as explained in the previous section. The second criterion is accomplished by the strategies defined for each situation, computing velocity commands complying with the robot kinodynamic constraints. Each motion leads to a stretch of trajectory, which combined during navigation drive the robot to the goal maintaining the highest velocities possible.

Figure 5.1 reflects this idea. Essentially, there are two main planning decisions: passing before the object ($RobotFront$) and passing after the object (*RobotBehind*). In turn, each is decomposed into different sub-decisions depending on whether the robot is before, inside or after the collision band. The execution of each motion strategy depends in turn on several *decision variables*, which drive the hierarchical decision process implemented

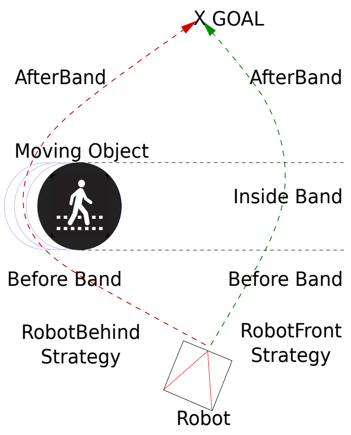


Figure 5.1: Trajectory in green when the robot passes before the moving obstacle and in red when the robot passes after the moving obstacle.

in a decision tree. These variables are formally defined in the next subsection.

5.2.1 Decision variables

Table 5.1 summarizes the *decision variables*, some in the workspace $\mathcal{W}$ and others in $DOVS$, including a brief explanation about their meaning and their use in the different situations. An important decision variable is GD (Goal Direction), which maps a circular trajectory from the current robot location towards the goal. If this trajectory is followed, reaching the goal is ensured. Commands always have to be selected in FV (free-velocity) space. Then, when GD lies inside a DOV, it has to be moved to FV (GD_{new} in Fig. 5.2a). To that end, a nearby velocity sub-goal that avoids the obstacles, not only in the next sampling period but also within the visibility horizon, is chosen. Fig. 5.2b shows a situation in which there are not $UpperFree$ velocities, so only $\mathbf{V}_{down}$ and $\mathbf{V}_{side}$ velocities can be selected.

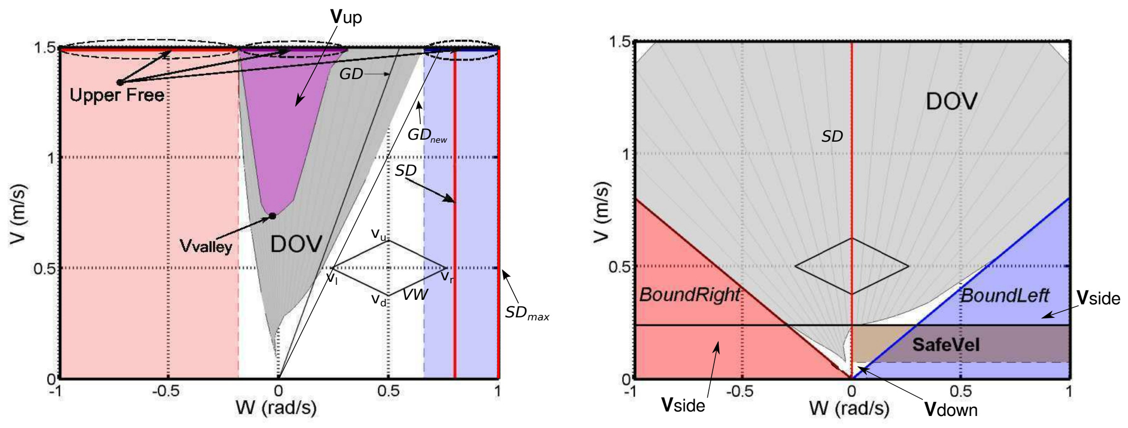
5.2.2 Optimization based on decision variables

The decision making process described above is implemented by means of navigation strategies, which use the defined decision variables. The objective of the strategies is to navigate towards the goals in dynamic scenarios, executing near time-optimal trajectories whilst ensuring safety (no collisions). Unlike in static environments, in dynamic environments where obstacle motion is uncertain or it might unexpectedly change, computing global time-optimal trajectories is not feasible.

To apply the decision making strategies based on $DOVS$ model, the direction to the goal is projected in $DOVS$ as the decision variable GD, representing a steering direction which drives directly to the goal, planning initially a circular trajectory γ_{GD} from the current robot location. This trajectory is not time-optimal, so it will only serve as an initial reference to move towards the goal. Two kinds of extremal controls are chosen

Table 5.1: Decision variables in W and *DOVS*.

Variable	**Meaning**	**Situation**
W		
$RelPos$	relative position before, in or after the collision band	all
$AngDis$ (θ_m)	angular distance robot-goal	all
DOVS		
FV - *DOV*	free (FV) and non-free (*DOV*) velocities	all
$\mathbf{V}_{high}$	minimum velocities to pass before object	PassingBefore, SlowingDown
$\mathbf{V}_{low}$	maximum velocities to give way to object	PassingBefore, SlowingDown
$\mathbf{V}_{up}$ - $\mathbf{V}_{down}$	high ($\mathbf{v} > \mathbf{V}_{high}$) - low ($\mathbf{v} < \mathbf{V}_{low}$) free velocities	all
$\mathbf{V}_{side}$	free velocities $\mathbf{v} = \{(\omega, v) \mid \frac{v}{w} = \gamma_j \notin [\gamma_l, \gamma_r]\}$	all
$\mathbf{v}_{Valley}$	$(\omega, v)_{Valley} = min(\mathbf{V}_{high})$	PassingBefore
$UpperFree$	free angular velocities with maximum linear velocity	PassingBefore, PassingAligned
$SafeVel$	safe low velocities to avoid or escape from collision	SlowingDown
$BoundRight$ - $BoundLeft$	γ_r - γ_l mapped in velocity space, boundaries for $\mathbf{V}_{side}$	AvoidingObject
v_m, ω_m, a_m, α_m	maximum linear and angular velocities and accelerations	all
$VW(VelocityWindow)$	velocities that can be reached within next time interval	all
$GoalDirection(GD)$	$(\omega, v)_{GD}$ mapped direction in velocity space	all
$SteeringDir(SD)$	ω_{SD} mapped angular deviation to goal in velocity space	all



(a) (b)

Figure 5.2: Decision variables in *DOVS*. (a) Situation in which there are *UpperFree* velocities (magenta, blue and pink); given that GD lies inside the DOV, the closest *UpperFree* to GD will be chosen, computing GD_{new}. (b) Situation in which *UpperFree* is occupied with velocities leading to collision (gray), only low velocities in $\mathbf{V}_{down}$ and $\mathbf{V}_{side}$ can be selected.

to be applied every sampling time to reach GD, either maximum angular acceleration to optimally align first the robot to goal position, or maximum linear acceleration for optimally approaching the goal. This kind of trajectories were derived in [RP94] as time-optimal ones for differential-drive robots with acceleration constraints in free space. We have adapted that technique for dynamic environments, obtaining near time-optimal trajectories. They lead to clothoid and anti-clothoid trajectories, respectively (see fig. 5.3a-b). Moreover, we have to take into account the kinodynamic constraints of the vehicle, which limit the acceleration and the velocity to be applied during every control sampling time.

The time-to-goal for a clothoid, $t_{G_w}(\mathbf{v}_c, \theta_m, d_G)$, and for an anti-clothoid, $t_{G_v}(\mathbf{v}_c, \theta_m, d_G)$, depends on several *decision variables.* Since these trajectories are computed from *Fresnel* functions, which analytic solution cannot be obtained for all the decision variables, we have analyzed by simulation both time functions (t_{G_w}, t_{G_v}) for different ω_0 initial conditions. In Appendix C, these functions are described. Figure 5.3 represents them. The conclusion of this study is that for an ample range of ω_0 and for near and far goals the clothoids result in lower times, mainly for low values of ω_0. So, it can be deduced that it is always better initially to increase the angular velocity up to the maximum if possible, rather than keeping the initial angular velocity, until an angular deceleration is needed for aligning. When the robot is nearly aligned to the goal, a linear acceleration is applied to speed up to maximum velocity. Algorithm 5.1 implements this method, and is used in the strategies to avoid moving obstacles.

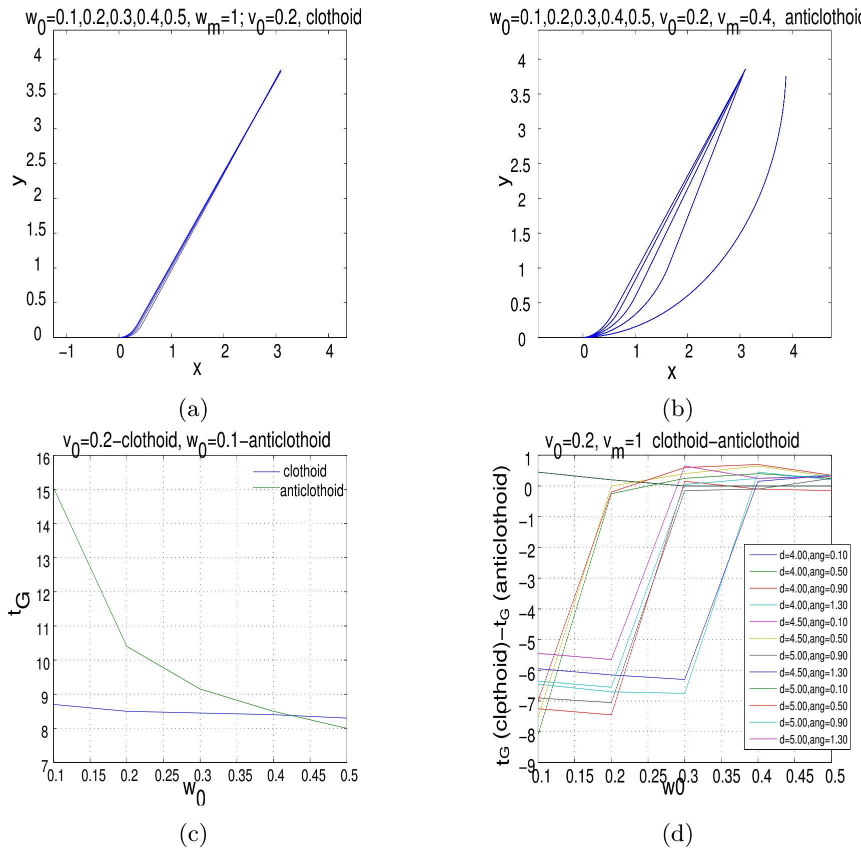


Figure 5.3: (a) Clothoid, and (b) anti-clothoid trajectories for different initial values of ω_0. The higher ω_0, the quicker the aligning to the goal. (c) Times to goal t_G for trajectories in a and b. (d) Time to goal difference t_G(clothoid) $-t_G$(anticlothoid) for different goal distances (d) and angular deviations (ang) with high and low maximum linear velocities.

5.2.3 Situation based navigation

The planner reasons in the *DOVS* space described in section 4.3 to represent the dynamism of the environment. In a scenario with obstacles, the trajectories introduced before do not lead to time-optimal trajectories because they may result in collisions. But it is possible to find near time-optimal and safe trajectories computed in the free velocity space FV, easily identifiable in *DOVS*. So, the methodology presented in section 5.2.2 is applicable in this space, avoiding obstacles whilst preserving low time-to-goal motions and safety.

The strategies are associated to *situations* detected. Figure 5.4 shows the situation tree, where the leaves correspond to all the situations that can be identified. Each of the *situations* has an associated *action*; the sequence of actions applied to reach the goal constitutes a strategy, which implements the general decision making process described in section 5.2. The robocentric planner executes cyclically, performing the strategies and so adapting to the changing environment in real-time. The planner establishes a

Algorithm 5.1 Align the robot to the goal

```
Require:
 1: $\mathbf{v}_c$, the current velocity of the robot
 2: $(\omega, v)_{GD}$, max. vel. to goal direction in $DOVS$
 3: $\theta_m$, angular deviation to goal from current robot position
 4: $d_G$, distance to goal from current robot position
 5: function ALIGNING($\mathbf{v}_c$, $(\omega, v)_{GD}$, $\theta_m$, $d_G$)
 6:     if $\mid \omega_c \mid \neq \mid \omega_{GD} \mid$ then                ▷ robot not aligned (clothoid)
 7:         $\omega_i = argmin_{\mathbf{v} \in VW}(t_{G_w}(\mathbf{v}_c, \theta_m, d_G)) =$
 8:             $= argmin_{\mathbf{v} \in VW}(w_m - \omega_{GD})$
 9:         $v_i = v_c$
10:     else                                               ▷ robot is aligning (anti-clothoid)
11:         $\omega_i = \omega_c$
12:         $v_i = argmin_{\mathbf{v} \in VW}(t_{G_v}(\mathbf{v}_c, \theta_m, d_G)) =$
13:             $= argmin_{\mathbf{v} \in VW}(v_c - v_{GD})$
14:     end if
15:     return $(\omega_i, v_i)$
16: end function
```

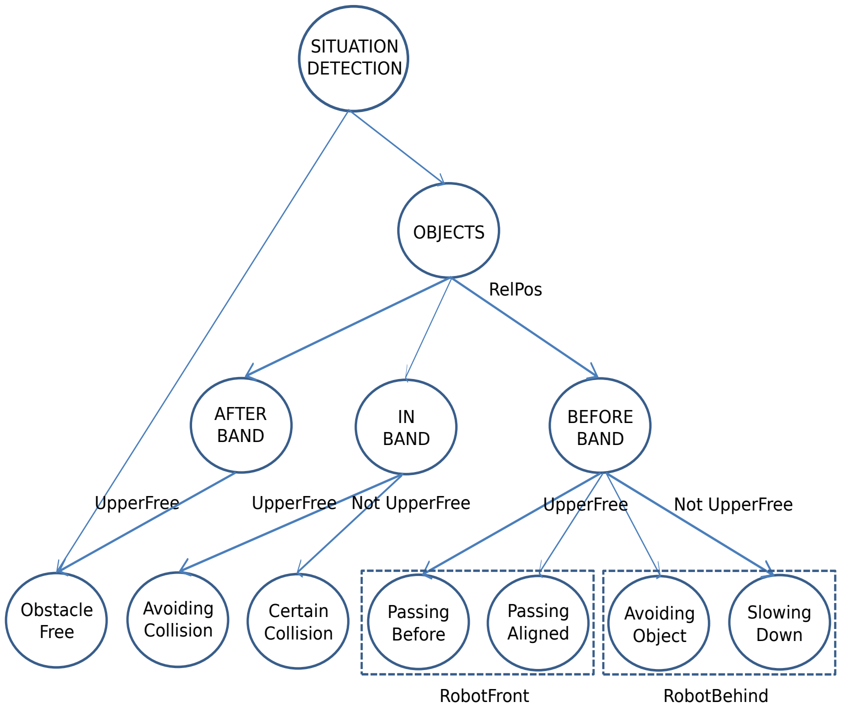


Figure 5.4: Situation Tree. Leaves represent all the situations that can be identified, each of which has an associated navigation strategy.

Algorithm 5.2 Basic **FreeMotion** for the robot

```
Require:
1: $\mathbf{v}_c$, the current velocity of the robot
2: $(\omega, v)_{GD}$, max. vel. to goal direction in $DOVS$
3: $\theta_m$, angular deviation to goal from current robot position
4: $d_G$, distance to goal from current robot position
5: function FreeMotion($\mathbf{v}_c$, $(\omega, v)_{GD}$, $\theta_m$, $d_G$)
6:     $\mathbf{v}_1 = \mathbf{v}_c$
7:     $\mathbf{v}_2 = Aligning(\mathbf{v}_c, (\omega, v)_{GD}, \theta_m, d_G)$
8:     $\mathbf{v}_3 = (\omega_i = 0,\ v_i = v_{i-1} + a_m\ \Delta t)$          ▷ $a_m$, max. linear acc.
9:     $\mathbf{v}_4 = (\omega_i = 0,\ v_i = v_m)$          ▷ $v_m$, max. linear vel.
10:    return $(\mathbf{v}_1 - \mathbf{v}_2 - \mathbf{v}_3 - \mathbf{v}_4)$
11: end function
```

long-term plan within the horizon defined by the field of view of the sensors and not only for the next sampling period. In this sense, this local planner is not purely reactive. It computes a safe trajectory for that space horizon. The plan is re-computed for each sampling period, so only the immediate action of the strategy required by the current situation is applied at each sampling time.

Table 5.2 shows the actions associated with each situation. Roughly speaking, the method consists in applying a sequence of maximum angular and linear accelerations to reach the maximum linear velocity in the free velocity space (FV), following the command computation developed in section 5.2.2. Before doing it, if the goal direction mapped in the velocity space, GD, is inside a dynamic obstacle DOV, a close new goal GD_{new} is mapped in FV in each situation. This way the near time-optimal command sequence is applied to reach that goal, avoiding the moving obstacle until GD is in FV again.

Algorithm 5.2 represents the basic motion to be applied in all the situations, as shown in Table 5.2. The only difference between situations is the election of a new GD_{new} in the free velocity space FV. Figures 5.5–5.9 describe situations *ObstacleFree*, *PassingBefore*, *PassingAligned*, *AvoidingObject* and *SlowingDown*, its associated actions and the strategies derived from them. The *AvoidingCollision* situation is a consequence of the robot having entered the collision band of an obstacle and then having to execute a velocity in free space to avoid collision. The *CertainCollision* situation appears when the robot falls into an unavoidable collision situation, due either to an unexpected obstacle appearing on the scene, or because no free velocity is available (a blocking situation, e.g. objects surrounding the robot).

Table 5.2: Actions for the situations.$\mathbf{V}_{sol}$ represents the sequence of velocities to be applied in a situation, $\mathbf{v}_c = (\omega, v)_c$ is the current velocity of the robot, $(\omega, v)_{Valley} = \mathbf{v}_{Valley}$, $\mathbf{v}_{GD} = (\omega, v)_{GD}$ is the maximum velocity in $DOVS$ for the Goal Direction (GD), ω_{GD} is the angular component of GD bounded by maximum angular velocity mapped into $DOVS$, $v_{_{SafeVel}}(\mathbf{v}_1)$ is the nearest linear free velocity with respect to $\mathbf{v}_1$, and $VW = \mathbf{v} \in FV \cap \{\mathbf{v}_l, \mathbf{v}_u, \mathbf{v}_r, \mathbf{v}_d\}$, where $\mathbf{v}_l = (\omega_c - \alpha_m \Delta t, v_c)$, $\mathbf{v}_r = (\omega_c + \alpha_m \Delta t, v_c)$, $\mathbf{v}_u = (\omega_c, v_c + a_m \Delta t)$ and $\mathbf{v}_d = (\omega_c, v_c - a_m \Delta t)$. GD_{new} is the new Goal Direction in Free Velocity (FV) closest to the true GD.

Situation	**Actions (velocities)**
$ObstacleFree$ (Fig. 5.5)	$\mathbf{V}_{sol} = \textbf{FreeMotion}(\mathbf{v}_c, GD, \theta_m, d_G)$
RobotFront Strategy	
$PassingBefore$ (Fig. 5.6)	$\mathbf{V}_{sol} = \textbf{FreeMotion}(\mathbf{v}_c, GD_{new}, \theta_m, d_G) \mid \mathbf{v}_{GD_{new}} = (\omega_{valley}, v_m)$
$PassingAligned$ (Fig. 5.7)	$\mathbf{V}_{sol} = \textbf{FreeMotion}(\mathbf{v}_c, GD_{new}, \theta_m, d_G) \mid \mathbf{v}_{GD_{new}} = (\omega_{GD_{new}}, v_m) \in FV,$ $\omega_{GD_{new}} = \min_{\mathbf{v} \in UpperFree}(\lvert\omega\rvert)$
RobotBehind Strategy	
$AvoidingObject$ (Fig. 5.8)	$\mathbf{V}_{sol} = \textbf{FreeMotion}(\mathbf{v}_c, GD_{new}, \theta_m, d_G) \mid \mathbf{v}_{GD_{new}} = (\omega_{GD_{new}}, v_m) \in FV,$ $\omega_{GD_{new}} = \min_{\mathbf{v} \in UpperFree}(\lvert\omega - \omega_{GD}\rvert)$
$SlowingDown$ (Fig. 5.9)	**If** $(\omega, v)_c \in DOV$ **then** $\mathbf{v}_1 = \mathbf{v}_c$, $\mathbf{v}_2 = (\omega_2 = \omega_1, v_2 = argmin_{\omega=\omega_2}(v_c - v_{_{SafeVel}}(\mathbf{v}_1)))$ **Else** $\mathbf{V}_{sol} = \textbf{FreeMotion}(\mathbf{v}_c, GD_{new}, \theta_m, d_G) \mid \mathbf{v}_{GD_{new}} = (\omega, v)_{GD_{new}} \in SafeVel$ **EndIf**

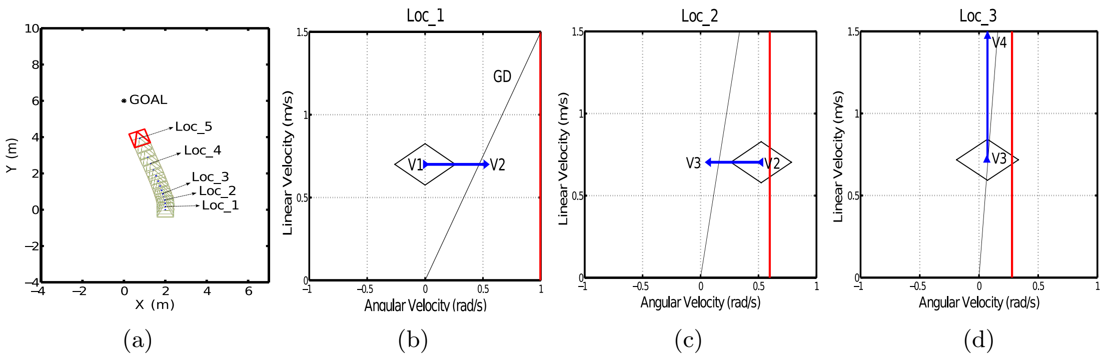


Figure 5.5: Evolution of ***ObstacleFree*** situation. Algorithm 5.2 implements the motion for this situation. (a) Part of the trajectory followed by the robot. (b)-(d) Velocity space of the robot and the applied actions in three representative locations of the trajectory. The vertical red line represents the Steering Direction SD and the black line the Goal Direction GD, to be reached for alignment to the goal. In Loc_1 a maximum angular acceleration ($\mathbf{v}_1 - \mathbf{v}_2$) is applied to reach SD every sampling period (the angular deviation of the robot is high). In Loc_2 a maximum angular deceleration ($\mathbf{v}_2 - \mathbf{v}_3$) is applied to reach GD (the angular deviation has decreased). In Loc_3 the robot applies commands at maximum linear acceleration ($\mathbf{v}_3 - \mathbf{v}_4$), while in Loc_4 and Loc_5, a straight line at maximum linear velocity $\mathbf{v}_4$ is achieved.

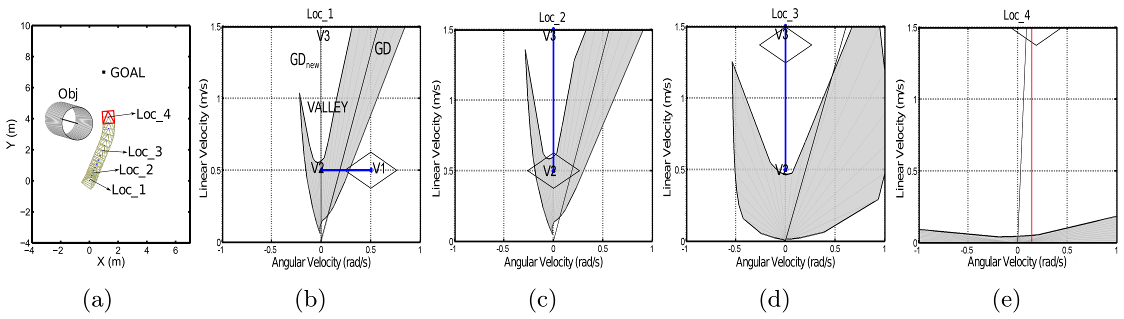


Figure 5.6: Evolution of ***PassingBefore*** situation. (a) Trajectory of the robot. (b)-(e) *DOVS* at four relevant instants during the trajectory. In this situation the robot can cross the collision band to pass before the object. There is a set of free velocities in the *UpperFree* zone which can be reached inside the bounds of the *DOV* ($\mathbf{V}_{up}$), identified as a *valley*. The depth of the valley ($\mathbf{v}_{valley}$ in table 5.1) provides certain knowledge about the level of safety for the robot to perform the strategy. The deeper the valley, the greater its width, and the safer it is to traverse the *DOV* to reach the free velocities in $\mathbf{V}_{up}$. A new GD_{new} is computed close to the initial GD in $\mathbf{V}_{up}$, such that it contains velocity $\mathbf{v}_{valley}$. A sequence of clothoid ($\mathbf{v}_1 - \mathbf{v}_2$) at maximum angular deceleration (b), anti-clothoid ($\mathbf{v}_2 - \mathbf{v}_3$) at maximum linear acceleration (c,d), and straight line at maximum linear velocity is applied (e).

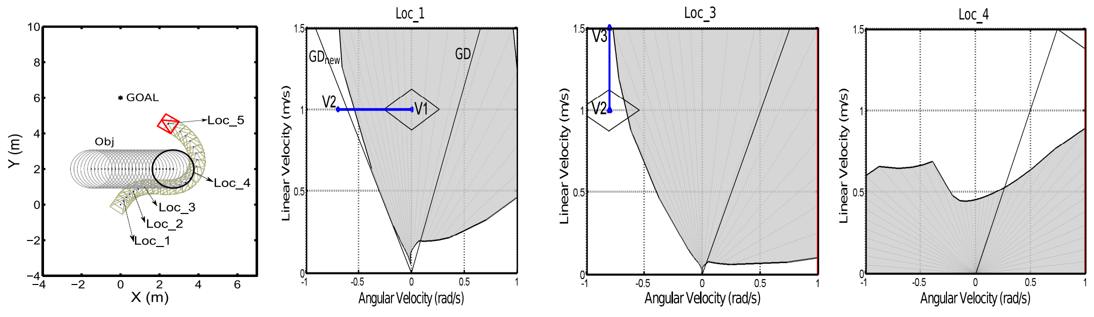


Figure 5.7: Evolution in ***PassingAligned*** situation. (a) Trajectory performed by the robot. (b)-(d) *DOVS* for different motions of the robot. The set of velocities in *UpperFree* leads the robot to move in the same direction as the object. In this case, the robot lies inside the *DOV* and *GD* is not free. GD_{new} is defined to escape from dangerous velocities, applying an angular acceleration motion (clothoid $\mathbf{v}_1 - \mathbf{v}_2$). Then, maximum linear acceleration ($\mathbf{v}_2 - \mathbf{v}_3$) is applied to move in the same direction as the obstacle (Loc_3). If the robot moves faster than the object, a valley appears inside *DOV*, which allows the robot to change the strategy to a *PassingBefore* situation. Finally, *AvoidingCollision* (Loc_4) and *ObstacleFree* (Loc_5) situations are identified when the robot lies inside the collision band and when it exits it, respectively.

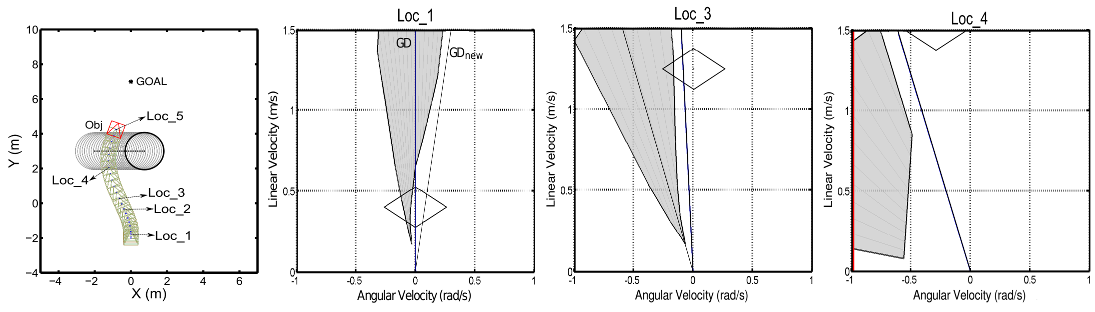


Figure 5.8: Evolution of ***AvoidingObject*** situation. (a) Partial trajectory of the robot. (b)-(d) *DOVS* representation at different instants. In this situation, the *DOV* of a moving object occupies the middle zone of *DOVS*, leaving two zones of velocities in *UpperFree* which belong to $\mathbf{V}_{side}$ (b). A new GD_{new} close to the current *GD* is selected in $\mathbf{V}_{side}$, to then apply the sequence of motions computed by Algorithm 5.2. The robot maneuvers to pass behind the object.

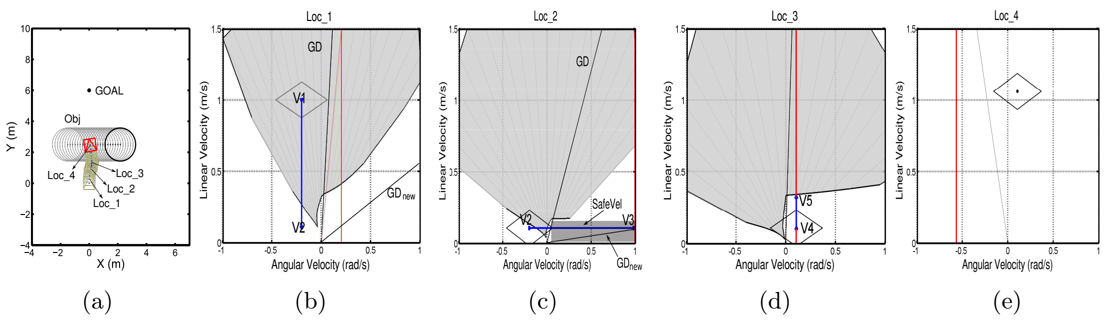


Figure 5.9: Evolution in ***SlowingDown*** situation. (a) Trajectory of the robot. (b)-(e) *DOVS* information at different locations of the robot. There are no $UpperFree$ velocities in *DOVS* (see Fig. 5.9a-b). In this case, the velocity of the robot is inside the *DOV* of the object, and the robot has to slow down to avoid a collision. A sequence of anti-clothoid trajectories (deceleration) is applied to escape from the dangerous velocities ($\mathbf{v}_1 - \mathbf{v}_2$). Then, a sequence of clothoid trajectories ($\mathbf{v}_2 - \mathbf{v}_3 - v_4$) is computed to maintain the robot at a low safe velocity in the $SafeVel$ zone until the object passes (c), following a new GD_{new} computed in $SafeVel$. The set $SafeVel$ defines the velocities which will be freed first, given the motion of the object. Finally an *ObstacleFree* situation appears (e), and the corresponding strategy is applied.

5.3 Experimental evaluation and metrics

5.3.1 Simulation results

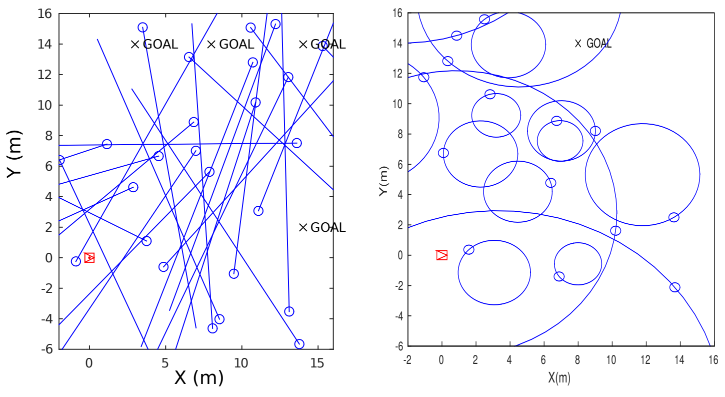
In this section we evaluate the proposed technique qualitatively and quantitatively. We have performed several simulations where the robot traverses different kind of scenarios to reach different goals[3] [4].

In these scenarios only moving objects appear around the robot. We have considered obstacle occupancies of 4% and 7% (Fig. 5.10), given that the 4% value is a commonly found scenario in the literature ([BVdB15], [MGF09]). Although these occupancies are seemingly not very dense when compared to static benchmark scenarios, the large impact of moving obstacles in the effective space for safe planning (see Fig. 5.10) makes them suitably challenging. This effective space has been computed as the mean of the forbidden velocity areas in *DOVS*, a way to quantify the limited robot maneuverability. See table 5.3.

We have tested navigation for three different types of scenario during simulation: *Linear*, *Non-Linear*, both for moving obstacles, and *Complete*, having static and moving obstacles. 40 different scenarios are simulated in *Linear* and *Non-Linear* sets. In each scenario the robot has to reach four different goals. In total, we have run 1440 simulations, 160 for each % of obstacle occupancy (4% and 7% for *Linear*, 4% for *Non-Linear*), for the three sets of velocities V_1, V_2, V_3 described in Table 5.3, and for a maximum linear robot velocity of 1.5 $m.s^{-1}$.

[3] Simulations to show the navigation performance of the robot in several scenarios and conditions can be accessed from `https://youtu.be/fqqUApjBRVo`

[4]An extensive video for simulations in random-generated scenarios is available from `https://youtu.be/gOUKIQpm7Pw`



(a) (b)

Figure 5.10: Scenarios with different densities of occupancy: (a) 7% of static density corresponds to 46% of occupancy in terms of safe velocity space available to be chosen in *Linear* scenario. (b) 4% of static occupancy represents about 35% of forbidden velocities in the *Non-Linear* scenario with obstacles moving in circular trajectories.

Table 5.3: Mean of forbidden velocity area in *DOVS* during simulations for different obstacle velocity ranges. The values of 4% and 7% for obstacle occupancy were measured as the area of the workspace occupied by the obstacles at a given instant (as if the obstacles were static). However, if we measure the occupancy in terms of forbidden velocities in *DOVS* as the area of each *DOV*, these values increase considerably.

	V_1 $v = 0.2$	V_2 $v = 0.5$	V_3 $v = 0.6..0.9$
Linear		$\omega = 0$	
4% occupancy	34.85%	40.12%	42.67%
7% occupancy	46.08%	53.98%	58.19%
Non Linear	$\omega = -0.15..0.15$	$\omega = -0.38..0.38$	$\omega = -0.45..0.45$
4% occupancy	34.97%	39.17%	40.06%

We have evaluated the technique in scenarios where the density of obstacles remains the same, so static bounds were established to define the workspace and simulate the movement of the obstacles cyclically, once the random movement has been generated. Table 5.4 reflects the collisions produced during simulation, as explained later. Column *Bounded(V_1)* indicates the percentage of collisions in scenarios with static bounds. A first observation is that the number of collisions is greater than in open scenarios, represented in the other three columns, due to the robot being trapped when it is close to them. Therefore, to focus on the evaluation of the robot navigating among moving obstacles, we have discarded the static bounds to evaluate the performance based on metrics. Another reason for collisions is because new obstacles appear from outside the limits of

Table 5.4: Percentage of collisions during simulations. Notice that in the highest occupancy situations and with the scenario bounds acting as static obstacles, many trapping situations can appear so some inevitable collisions occur.

	Bounded(V_1)	V_1	V_2	V_3
Linear				
4% occupancy	5%	0%	2.5%	3.75%
7% occupancy	32.5%	5%	10%	16.875%
Non Linear				
4% occupancy	10.6%	3.75%	15%	11.875%
NL-Aprox				
4% occupancy		10.62%	34.37%	46.25%

the scenario, so the robot might not have the reaction capability from kinodynamic constraints to avoid collision. Additionally, we have evaluated the method when conditions about obstacle motion is uncertain, scenario *NL-Aprox*. So, we have performed simulations in scenarios with obstacles moving with non linear trajectories and approximating the *DOVS* model with the linear approach. The collisions greatly increase, as can be seen seen in table 5.4. Therefore, the method benefits from the greater look-ahead if there is a proper estimation of the obstacle motion. Whenever the motion changes, the calculations should be recomputed and, in this case, our method behaves like a reactive one.

Figure 5.11 illustrates several iterations of a simulation of the *Linear* scenarios. The robot maneuvers around the obstacles while maintaining safety, selecting high velocities in the free velocity sub-space FV.

Figs. 5.12 and 5.13 show several metrics evaluated during navigation for the overall simulations in *Linear* scenarios, for each of the occupancy densities considered and for each set of velocities. As expected, in a scenario where fewer obstacles appear, the robot can move at higher velocities (Figs. 5.12a and 5.13a). Safety is evaluated from the distance measured between the robot and each of the obstacles (5.12b and 5.13b). Figures 5.12c and 5.13c illustrate the time-to-goal with respect to the optimal value for each goal, computed for free space. Longer trajectories are produced when there are more obstacles in the environment. Figures 5.12d and 5.13d describe the strategies selected during navigation. *AvoidingObject* and *AvoidingCollision* are the most frequently applied strategies. *PassingBefore* strategy occurs more frequently than *SlowingDown*, which means that the robot tries to search for free spaces to pass before objects at high velocity, rather than slowing down to give way to objects. It can be seen that for 7% occupancy *ObstacleFree* and *PassingBefore* appear less frequently than for 4% occupancy, as was predictable.

Regarding simulations for *Non-Linear* scenarios, Fig. 5.14 plots the same metrics as for the *Linear*. Similar results are obtained, so the technique adapts well to different

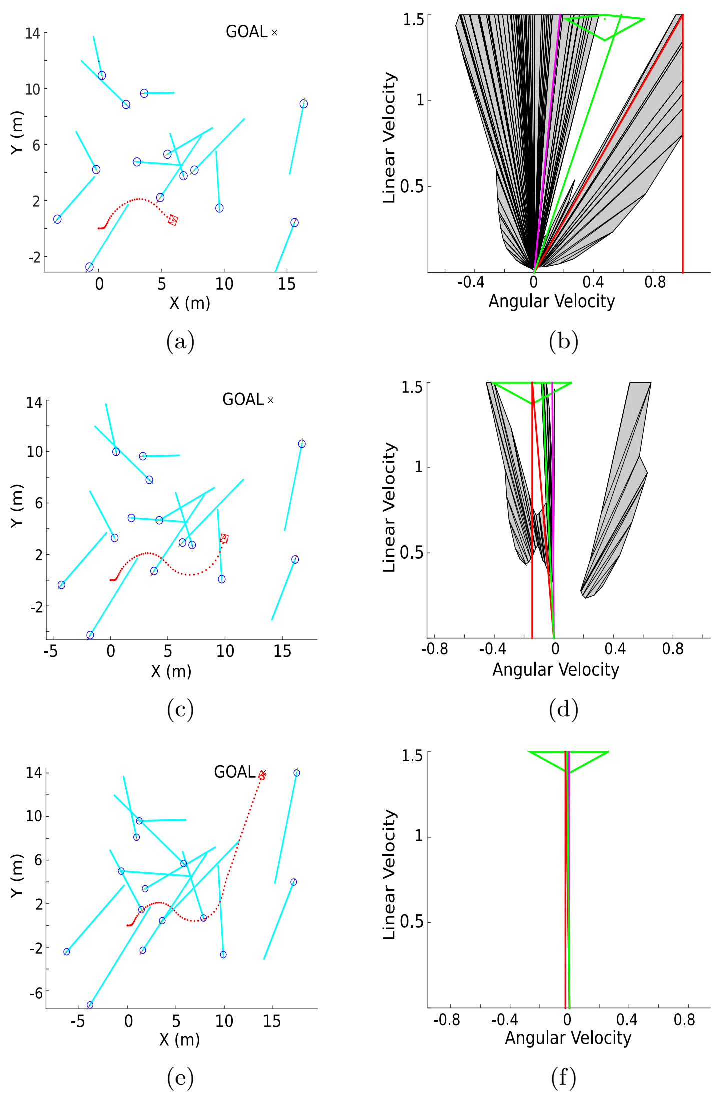


Figure 5.11: Simulation in a *Linear* scenario for 4% obstacle occupancy and velocities in V_1. In fact, this percentage increases in a dynamic scenario, as a result of the reduction on the available workspace and velocity space. The workspace (left) and *DOVS* (right) are represented at different instants. The planner searches for free velocity sub-spaces so that the robot (red) can maintain high linear velocities (rhomboid window) during navigation.

kinds of obstacle trajectories.

Moreover, the technique developed is conservative in the sense that any velocity in free velocity space FV is safe, but the velocities inside a DOV do not always or instantly lead to an unavoidable collision; they can be selected but only during a certain time without collision. This improvement is the subject of the work developed in Chapter 6.

Finally, in the *Complete* scenario shown in Fig. 5.15 there are static and moving

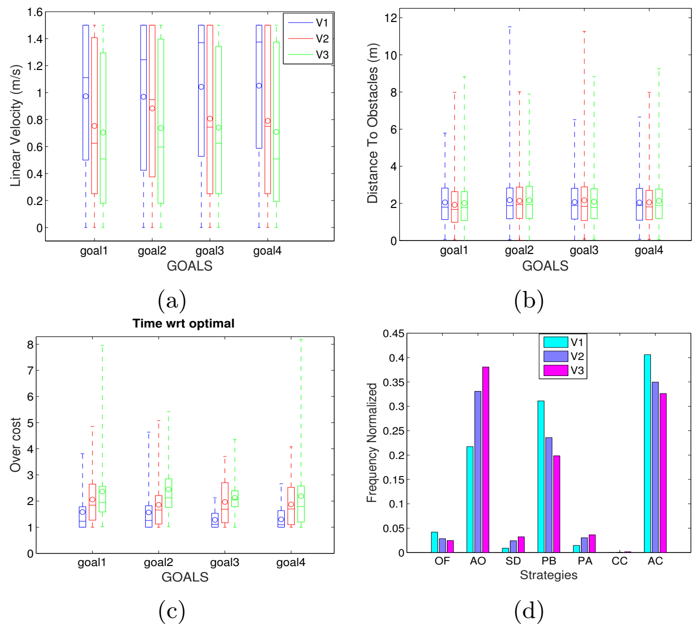


Figure 5.12: Metrics for 4% occupancy scenarios with obstacles moving linearly. Percentile, mean, and median for: (a) linear velocity, (b) distance to obstacles, (c) time with respect to optimal for free space, (d) histogram of the situations.

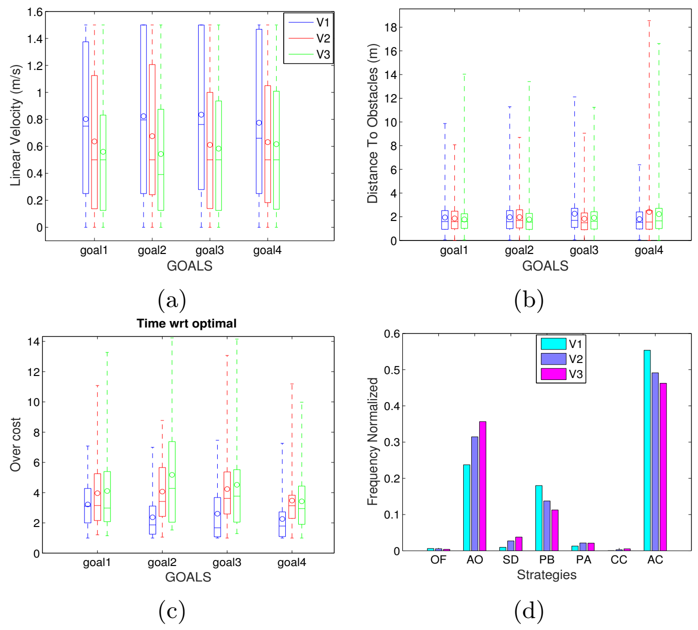


Figure 5.13: Metrics for 7% occupancy scenarios, with obstacles moving linearly.

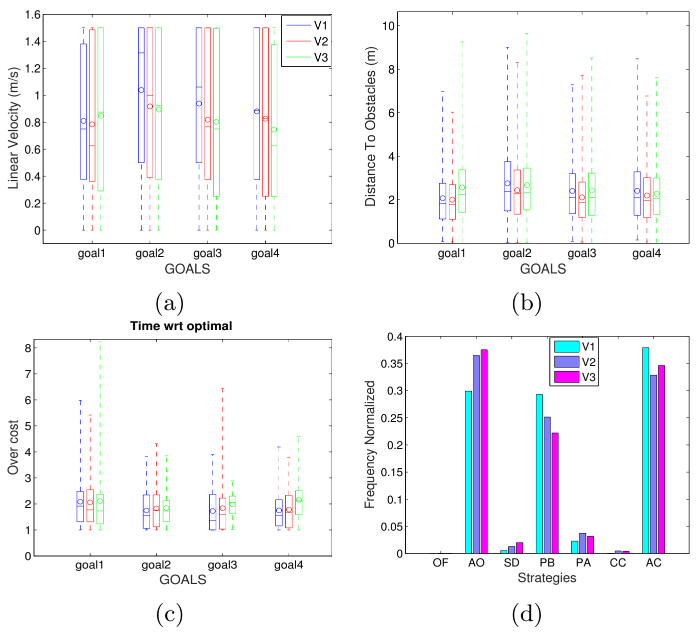


Figure 5.14: Metrics for simulations for 4% occupancy of non linear obstacles.

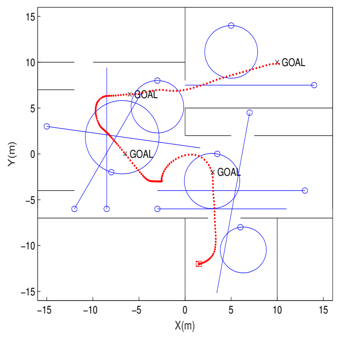


Figure 5.15: *Complete* scenario with static (black), linear and non linear obstacles describing their trajectory (blue), and the trajectory performed by the robot (red) to achieve the different goals.

obstacles (following linear and non-linear trajectories) to illustrate a general situation in which a robot has to move from one room to another, navigating among several objects and four subgoals (waypoints). Figure 5.16 represents the metrics evaluated: linear and angular velocity profiles, distance to obstacles and situations detected. In this scenario, where many static objects are present and appear in *DOVS*, the *ObstacleFree* situation is not encountered, so there are not many velocities free to maneuver.

5.3.2 Comparison with a *one-step* approach

We show here through simulation how our look-ahead planner obtains a better behavior and lower time-to-goal than a method which considers only the next-period candidate

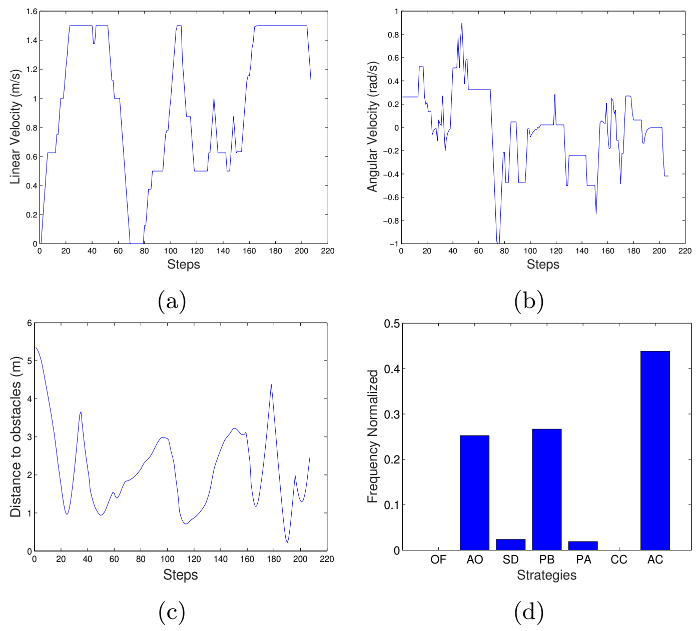


Figure 5.16: Metrics for simulation of the *Complete* scenario. Most of the time the robot moves at maximum velocity, slowing down when it becomes considerably disorientated from the goal (bigger changes observed for angular velocity). Strategy *PassingBefore* is applied many times, that is the robot moves many times at high velocities to pass before the obstacles, and no *CertainCollision* situation appears.

velocities. This local planning approach is denominated *one-step* approach, and it is the decision process used in [GSR09] and [SGF10] to develop a near time-optimal motion planner. The basic idea is that the planner chooses collision-free velocities from the velocity window (VW) that generate a time-optimal trajectory to the goal, evaluated in absence of obstacles. In our case, for a non-holonomic robot, the near time-optimal motions used are the ones introduced in 5.2.2. The planner proposed in this work computes collision-free commands from the velocity window based on the future evolution of the obstacles, look-ahead information previously considered to plan a safe trajectory to the goal, by choosing a *GoalDirection* (GD) outside DOV.

The evaluation carried out is performed in randomly generated scenarios, where the robot has to reach four different goals, during a set of 10 simulations. The range of velocities considered for the obstacles are: 0.2, 0.5, 0.8, 1.2. Figure 5.17 shows the mean values of the time-to-goal for each of the methods. It can be seen that for low obstacle velocities, our approach gets better results, while similar values are obtained with higher velocities. This is because the obstacles spend more time occupying the workspace: The *one-step* approach remains in the time-optimal for the current VW, while our approach favors the selection of commands towards bigger areas of free space. Therefore, both approaches obtain near time-optimal trajectories, but ours can get lower times to the goal.

Figure 5.18a,b plot a scenario for one of the simulations designed, where the robot

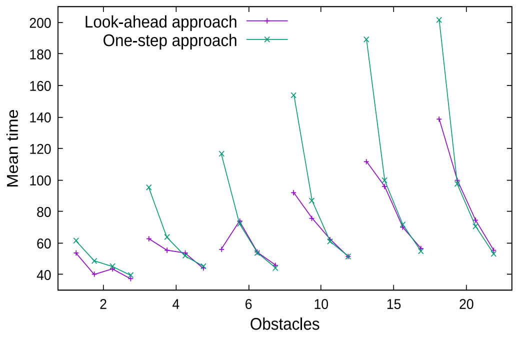


Figure 5.17: Comparison of the mean of the time-to-goal for our look-ahead planner and the *one-step* local method, for the velocity range considered.

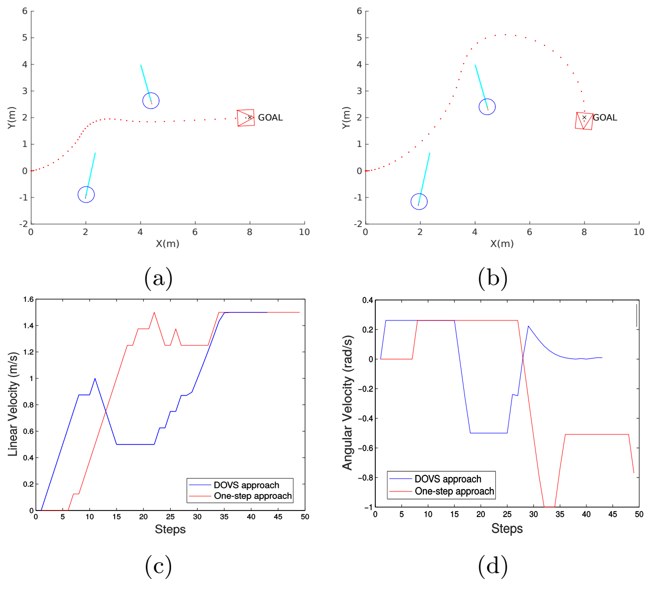


Figure 5.18: Trajectories followed by the robot reasoning within *DOVS* visibility (a), and *one-step* visibility (b). Profiles of linear (c) and angular (d) velocities during navigation. Looking further ahead at a single instant (*DOVS*) provides shorter and lower time-to-goal trajectories than using the theoretical time-optimal control to the goal at every sampling time (*one-step* approach).

behavior using both methods to move around the obstacles is also shown. As can be seen, our method makes the robot to pass between both obstacles, whilst the *one-step* approach computes a longer trajectory due to the implicit reactivity of the local planning. As a consequence, the time-to-goal and the trajectory length are reduced, Figs. 5.18c,d.

Figure 5.19 describes several steps of the simulation for both methods. *DOVS* look-ahead plans long-term maneuvers in the velocity free space, which permits to find safe commands when re-planning every sampling time to move between both obstacles, unlike *one-step* planner which only re-plans locally to follow a theoretical time-optimal trajectory towards the goal in the free space.

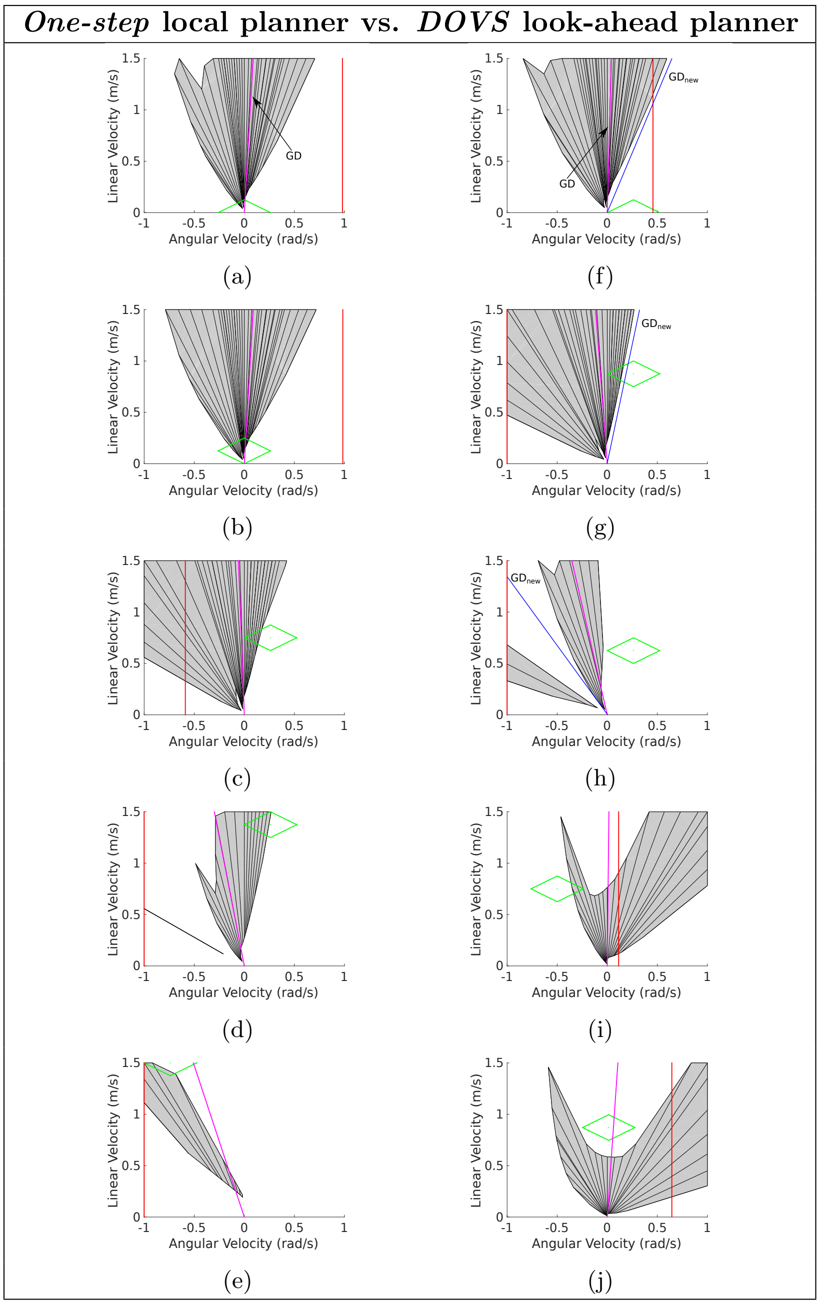


Figure 5.19: (a-e) *One-step* local planner: the velocity which would drive the robot to the goal applying time-optimal commands from the current position-velocity in free space (mapped *GD*) is selected. In (d) the velocity obstacle prevents the robot from choosing a velocity to maneuver towards the goal, until the moving object passes and *GD* becomes free to be followed (e). (f-j) *DOVS* look-ahead planner: it plans safe velocities out of the velocity obstacle for the sensor visibility horizon, driving the robot to the new mapped goal GD_{new} (f-h) instead of *GD*, leading the robot to slow down and maneuver to finally cross between both obstacles through the velocity valley (i,j).

Complexity of the planning approaches. In section 4.3.6 we provided a complexity analysis of different modeling techniques with respect to our approach. Now, we focus on the planning operation. *One-step* approach uses information only for computing the next control command as constrained to the acceleration window. If a further plan should be obtained, a tree search would be performed by expanding the nodes for the look-ahead considered t_{max}, which is the time horizon considered for modeling the obstacles. Then, the complexity would be $\mathcal{O}(q^{t_{max}/\Delta t} m \log m)$, where q is the fixed number of velocities that can be reached within the next sampling period, and $t_{max}/\Delta t$ represents the number of levels achieved during tree expansion.

The *DOVS* lookahead approach presented in this paper deals with multiple objects by merging all $VO_j\ j = 1..m$ obstacles (see 4.3.2). This operation requires for an object to compute the collision times and velocities for the robot trajectories considered for the other objects, and to repeat this for each object. The complexity is then $\mathcal{O}(nm^2)$. This approach provides enough information for computing plans in the defined workspace horizon, so the complexity is not affected by the parameter t_{max}, since a look ahead simulation is not needed for planning safe motions.

5.3.3 Real-world Experiments

We have evaluated the navigation strategies in two real scenarios. The first scenario corresponds to the kind of scenarios shown in the simulations with obstacles moving randomly in all directions. The second is a more structured scenario, an overtaking manoeuvre.[5]

Experimental setup. The navigation strategies have been tested on three Pioneer robots. These are differential-drive robots equipped with a 2-D laser rangefinder Sick LMS-200 and on-board Intel Centrino duo at 1.6 GHz. The computation time of the navigation strategies was around 200 ms, the time for which a new laser measuring is available. The field of view of the laser rangefinder sensor was 180 degrees, 0.5 degrees of angular resolution and a maximum range of 8 meters. The maximum translational velocity was set to 0.3 $m.s^{-1}$ and the rotational velocity to 0.4 $rad.s^{-1}$, fitting the velocity window used. The method developed in [MMM08] is used for mapping static and moving obstacles from rangefinder sensors, and an EKF-based technique is applied to track the moving objects, in which the state vector included the location and the velocities of the tracked objects.

The work presented in [PSF01] developed a navigation system on a wheelchair equipped with sensors. The authors represent the space occupied at instant t with a *time stamp map*, and compare the corresponding information in the previous step to determine moving and static obstacles by using a nearest-neighbor criterion. The *VO* approach is used to compute safe velocities, selecting the highest feasible velocity in the direction of the goal or a maneuver-avoidance velocity with a specific heading. As a result, wall-following and obstacle-avoidance behaviors are observed in simulation and real-world crowed environments.

[5]A video showing the experiments is accessible from `https://youtu.be/wfqDm7srzCM`

Experiment 1. Figure 5.20a depicts the hall-like scenario, with robot and object trajectories, not previously known. Fig. 5.20b-d shows the velocity profiles and the strategies. *ObstacleFree*, *AvoidingObject*, and *PassingBefore* are the most frequent situations, allowing to maintain the maximum linear velocity. Figure 5.21 illustrates several steps of the experiment.

Experiment 2. Figures 5.22a and 5.23 show an overtaking maneuver in a narrow corridor. Before overtaking obstacle 1, the robot slows down until obstacle 2 passes. When the maneuver is initiated, the maximum velocity is maintained until the end, as shown in Fig. 5.22b. *ObstacleFree*, *AvoidingObject* and *SlowingDown* are the situations which occur the most. The latter makes the robot reduce the velocity before starting the overtaking maneuver.

5.4 Conclusions

A robocentric technique for planning and navigation in dynamic environments has been developed. The *DOVS* model defined for mapping the dynamics of the environment allows the planning of safe motions within a space horizon, for instance the range of visibility of the on-board sensors. Several situations depending on the *decision variables* are identified applying specific actions associated to them, which implement the *RobotFront* or the *RobotBehind* strategies. The complexity of the method does not increase either by trajectories which are more complex than linear obstacle trajectories, as described in 4.3.6, or as a result of the space-time horizon for planning (5.3.2), unlike other techniques described in the literature. It has been shown that the long-term strategies selected every sampling time within the space horizon visibility improve the time to goal with respect to other techniques which are purely reactive. The method has been evaluated in simulation and in real-world experiments. In a dynamic scenario, the navigation difficulty can be measured from the velocity space area occupied by the obstacles, which somehow reflects the actual capability of the robot to maneuver. Sometimes collisions are unavoidable, mainly in cases where the robot is trapped near static obstacles or when obstacles appear in the scenario, due to robot's kinodynamic constraints.

From the lessons learned, we have observed that some improvements can be achieved. Using directly the *DOV* obstacles without merging them would make the processing more complex but enlarge the set of free and safe velocities to maneuver. This issue will be covered in Chapter 6, in which a planning technique reasoning on *DOVTS* is proposed. Blocking situations appearing in closed scenarios or with many static obstacles could be reduced achieving a different treatment for those static obstacles, by prioritizing free and safe velocities that move the robot away from them. However, this issue is not treated in the rest of the thesis and remains as future work.

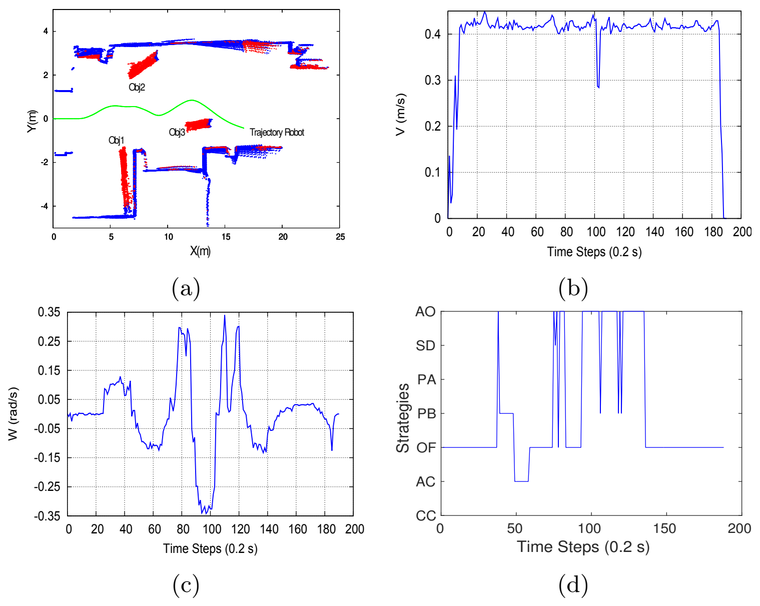


Figure 5.20: Hall scenario: (a) Trajectory of the robot (green) and environment perceived in experiment 1, with static and moving objects, (b) Linear velocity profile, (c) Angular velocity profile, (d) Situations.

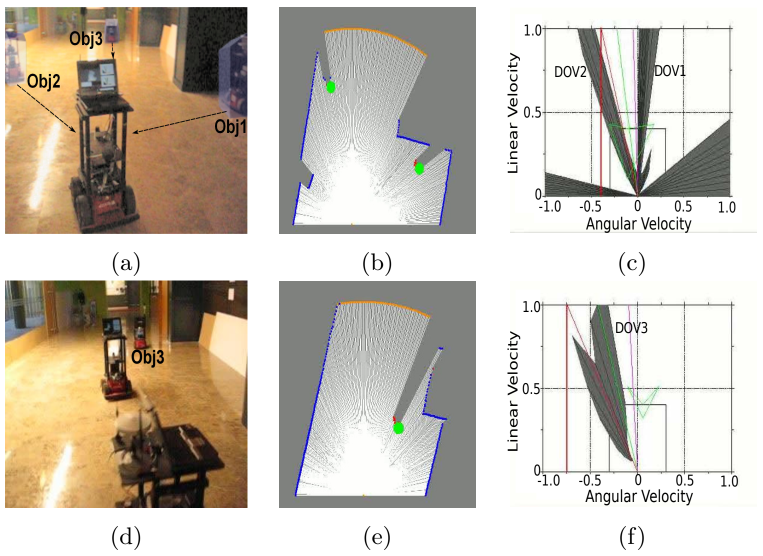


Figure 5.21: Experiment 1 in the *Hall* (*random*) scenario. The robot can pass between *Obj*1 and *Obj*2 (a-b) at maximum velocity (c), and avoids *Obj*3 (d-e) towards the side closest to the goal (f).

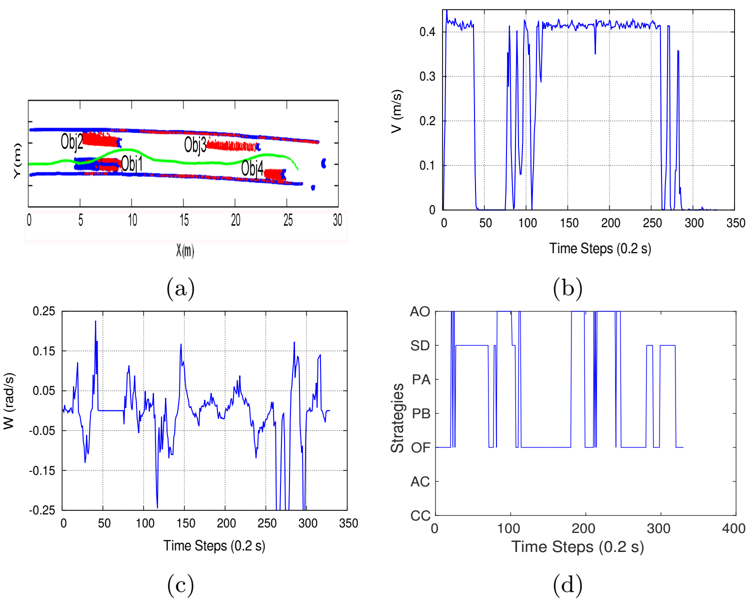


Figure 5.22: Overtaking scenario: (a) Trajectory of the robot (green) and environment perceived in experiment 2, (b) Linear velocity profile, (c) Angular velocity profile, (d) Situations.

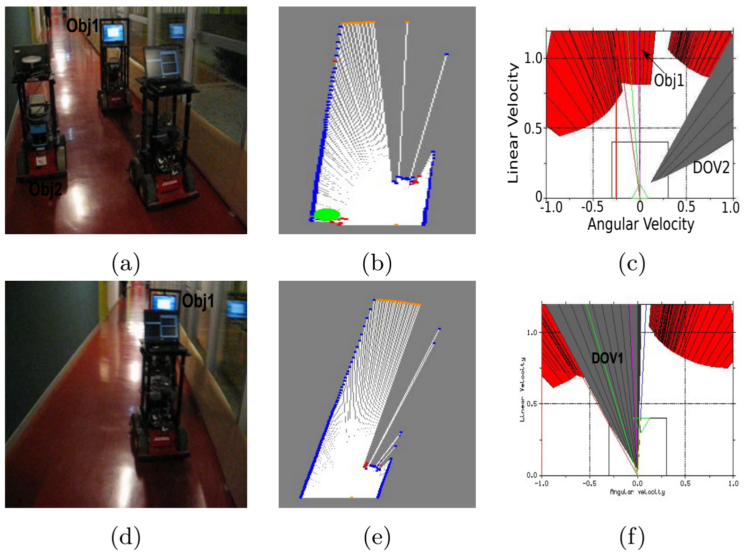


Figure 5.23: Experiment 2 in the *Overtaking* scenario. The robot has to slow down before overtaking obstacle *Obj*1. Red velocity obstacles (c,f) are the walls on both sides of the corridor and obstacle *Obj*1, which is detected as static given that both the robot and *Obj*1 are navigating with a similar velocity.

Chapter 6

A model-based robocentric planner on *DOVTS*

The model definition presented in Chapter 4 to represent the dynamic environment for a differential-drive robot allowed for researching into new approaches for planning and navigation. The previous chapter used the *DOVS* model to compute a motion for the robot. Searching on the Velocity space reduces the data processing but omits commands that may lead to a better navigation performance.

The aim of this chapter consists in improving the navigation behavior of the robot by explicitly using the information of time available in the model. A planning technique working on the Velocity-Time space is proposed to compute commands by, again, balancing safety and time-to-goal criteria. The evaluation of the method suggests a performance comparison with respect to using *DOVS*.

The work developed and the results obtained were presented at the European conference ECMR'17 [LM17].

6.1 Introduction

The main advantage of using *DOVTS* with respect to *DOVS* for planning has to do with the greater set of free velocities available for the robot. Velocities between different dynamic obstacles *DOVT*s can be exploited to obtain better time-to-goal trajectories. The Velocity-Time space of the robot is discretized in cells to search for motions. This allows to come up with two approaches: *Time-Aware* and *Conservative*. The first one respects the free velocities under *DOVT* surfaces, whereas the latter restricts them, like *DOVS*.

Next, section 6.2 introduces the details of the planning and navigation algorithm. Section 6.3 evaluates the method for the two approaches proposed in randomly generated simulated scenarios. And finally, in section 6.4, some conclusions and future work are presented.

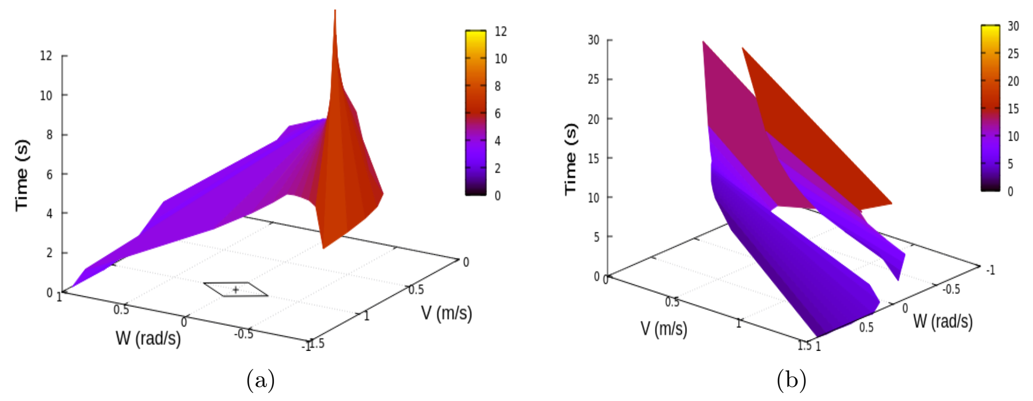


Figure 6.1: (a) Velocity-time space $DOVTS$ for one moving obstacle; the surface represents the corresponding velocity-time obstacle $DOVT$ including the dynamic information. (b) $DOVTS$ for two obstacles; free velocities can be selected between both surfaces representing the $DOVT$ obstacles.

6.2 Trajectory planning and navigation

This section introduces a motion planning and navigation technique that works directly on the $DOVTS$ model explained in Chapter 4. Figure 6.1 shows $DOVTS$ for one (a) and two (b) moving obstacles. Velocities in FVT, the free Velocity-Time space in the model, can be selected by a motion planner to compute safe motions without collision during a time horizon according to their collision time.

6.2.1 The planner

As the planner works in the Velocity-Time space, the goal location to be reached in the Workspace has to be projected in $DOVTS$ as a velocity goal, $\mathbf{v}_g$. From the current robot location in the configuration space, a circular trajectory towards the goal is computed, which is modeled into $DOVTS$ as the set of velocities which leads to the goal following Eq. (4.2), and bounded by the velocity constraints. The maximum velocity in the set refers to the velocity the robot should have to reach the goal at the next sampling step. Then, $\mathbf{v}_g$ corresponds to the maximum reachable velocity in the set within the bounds. Algorithm 6.1 reproduces the computation in pseudo-code. Then, the planner searches for a plan towards $\mathbf{v}_g$ along the whole available time horizon.

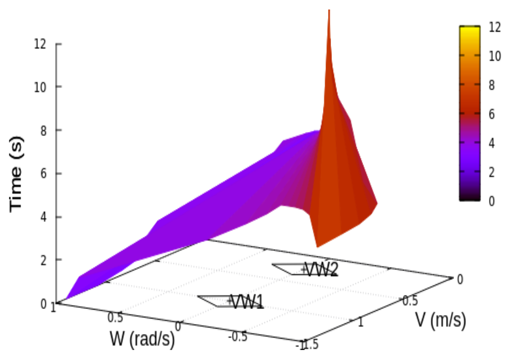
Algorithm 6.1 Compute Goal in the Velocity Space

```
Require:
1: $(x, y)_g$, goal position relative to robot in workspace
2: function ComputeVelocityGoal($(x, y)_g$)
3:     $\gamma_g$ = GetRadiusGoal($(x, y)_g$);
4:     $\mathbf{v}_g$ = GetVelocity($(x, y)_g$, $\gamma_g$)          ▷ Equation 4.2
5:     return $\mathbf{v}_g$
6: end function
```

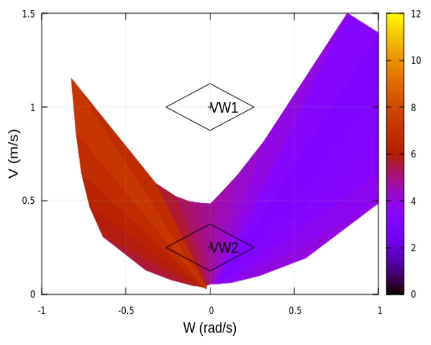
As already mentioned in Chapters 4 and 5, extremal controls can provide near time-optimal trajectories in environments with obstacles in order to maneuver the robot. These will be the controls selected to navigate, which are bounded by the dynamic constraints of the robot, leading to clothoid or anti-clothoid trajectories ([OM05]). These maximum accelerations will be used as the cell size in the grid explored by the planner.

The planner developed uses information mapped on *DOVTS* to select the more suitable velocities $\mathbf{v} = (\omega, v)$ for the robot. The planner searches for safe velocities in *FVT* and executes every sampling time to adapt to changes in the scenario. Simple circular trajectories are considered when planning, which may lead to non optimal solutions if they are applied in the long term. In addition, the *DOVTS* model offers a look-ahead at an instant which is valid while velocity conditions remain invariable. Not allowing the planner to be executed to update the changes observed in the environment would result in sub-optimal trajectories and be unsafe for the robot, requiring a forward simulation of the system, which is costlier. However, it could be taken into consideration, if some safety guarantees could be found. The selection of the best velocity command is conducted by safety and near-optimal time-to-goal criteria. To illustrate the idea, figure 6.2a shows a situation where an obstacle and two candidate velocities defining the robot in *DOVTS* are plotted, *VW1* and *VW2*. These represent two examples of velocities eligible for planning. The projection of this situation into *DOVS* is also shown in figure 6.2b to clearly state that *VW2* can only be temporally applied, as it lies under a *DOVT*.

For computing near-optimal time commands among safe velocities, an A*-like algorithm in the 3D space *DOVTS* is proposed. The planner constructs a tree by expanding velocity-time safe nodes in the search space ordered by a cost function $f(n) = g(n)+h(n)$, which estimates the cost to reach the velocity goal, $\mathbf{v}_g$. $g(n)$ is the cost to reach the current node n, and $h(n)$ is a heuristic cost which estimates the cost to get to the goal from n. In our method, the search space is defined as a velocity-time grid obtained from



(a)



(b)

Figure 6.2: Two alternative velocities for the robot, *VW1* and *VW2*, in (a) *DOVTS* and (b) its projection into the velocity space *DOVS*. They represent the kind of velocities eligible when planning, i.e. velocities which do (*VW2*) and do not (*VW1*) have a bounded time of application given by the time until collision with the obstacle. The planner will combine safety and time-to-goal criteria to select the best motion.

DOVTS, and the nodes of the tree correspond to the cells in the grid, $n = (\omega_n, v_n, t_n)$. The cost function $g(n)$ reflects the number of velocity changes taken and their closeness to the velocity goal. The algorithm selects commands that get the robot away from the obstacles, but as little as possible from the velocity goal. The heuristic term $h(n)$ is the time to goal, measured as the number of sampling steps to reach the goal velocity in absence of obstacles.

Formally, for a node $n = (\omega_n, v_n, t_n)$ being evaluated,

$$\begin{aligned} g(n) &= g(parent(n)) + 1 + OpCost(n) \\ h(n) &= t_{\mathbf{v}_g} - t_n \\ OpCost(n) &= 1+ \\ &\quad + \mathrm{dist}(cell_n, cell_{\mathbf{v}_g}) \\ &\quad - \mathrm{dist}(cell_{parent(n)}, cell_{\mathbf{v}_g}) \end{aligned}$$

The cost function $g(n)$ is interpreted as follows: the unity cost represents one velocity change opportunity at maximum acceleration (hence making shorter plans better); *OpCost* models an opportunity cost of staying as close to the velocity goal as possible, with the euclidean distance between cells given in grid coordinates (which represent velocity changes at maximum acceleration) used to prefer velocities closer to the goal and the 1 term used to ensure an always positive increase in cost.

In other words, the planner minimizes the velocity changes while trying to stay close to the velocity goal, simultaneously avoiding Velocity-Time collision configurations. The heuristic, in turn, represents the minimum possible remaining steps in the plan given the height in time steps of the search space, used to accelerate the search without having an influence in the optimal solution.

Algorithm 6.2 represents the process to compute the motion command (ω, v) to be applied for the robot during a sampling time, for the plan found at every sampling step.

Figure 6.3 depicts an example of different navigation solutions proposed. Workspace with two obstacles following linear trajectories (the collision bands are plotted), the trajectory of the robot, the corresponding *DOVTS* and velocity-time grid at several instants are represented. The colored grid cells correspond to the obstacles in *DOVTS*, i.e., forbidden velocity-time commands for the robot.

6.2.2 Navigation strategies

The planning and navigation algorithm can be applied using different strategies. We consider here two of them: the *Conservative* strategy, in which the velocities lying under any *DOVT* are considered as always occupied, thus they are not eligible (see figure 6.3a); and the *Time-Aware* strategy, in which all the possible free velocities are considered for planning, even those that could lead to collision at a future time (see figure 6.3b). Both strategies are evaluated in Section 6.3.

Conservative strategy could be associated to the approach proposed in Chapter 5, where the projection of *DOVTS* model is used. A fusion of the several *DOV*s was applied,

Algorithm 6.2 Compute Robot Motion

```
Require:
 1: $DOVTS$, Velocity-Time space of the robot
 2: $\mathbf{v}_c$, current velocity of the robot
 3: $\mathbf{v}_g$, velocity goal
 4: $s$, strategy used for planning (Conservative, Time-Aware)
 5: function ComputeMotion($DOVTS$, $\mathbf{v}_c$, $\mathbf{v}_g$, $s$)
 6:     mapGrid = ComputeGrid($DOVTS$)
 7:     $cell_r$ = CellGrid($\mathbf{v}_c$)
 8:     $cell_g$ = CellGrid($\mathbf{v}_g$)
 9:     mapGrid = mapGrid $\cup$ $cell_r$ $\cup$ $cell_g$
10:     $velPlan$ = MotionPlan(mapGrid, s)
11:     $(\omega, v)$ = FirstElement($velPlan$);
12:     return $(\omega, v)$
13: end function
```

so free velocities between DOVs were not admissible. Althought the specific planning techniques developed here and in Chapter 5 are different, having the free velocities under any $DOVT$ surface available for planning serves for comparison from the point of view of maneuverability.

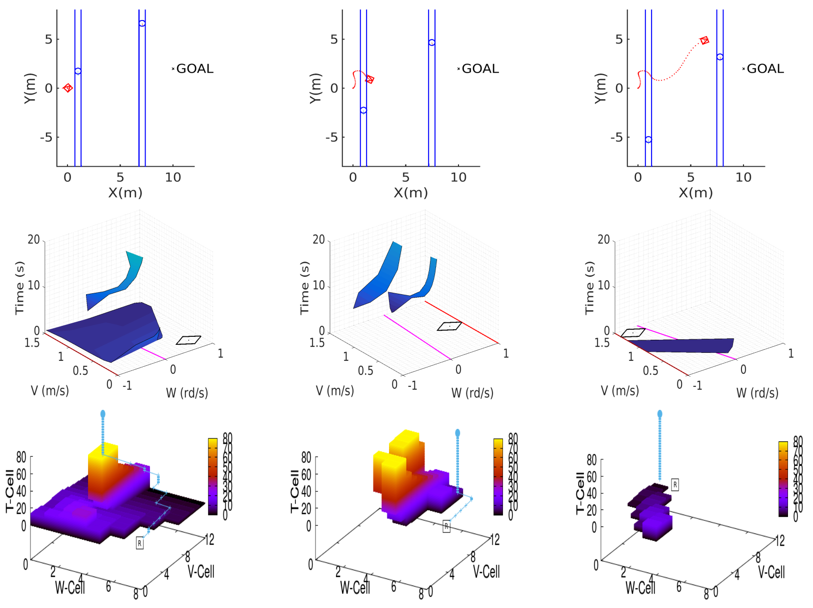
6.2.3 Grid parameters

The size of the cells in the grid is also an important issue, because it will determine the precision of the velocity obstacles in the grid, and the smoothness of the trajectories obtained from the command application. The computation time will fix also the minimum time step needed for planning. In the current implementation the sampling time is $\Delta t = 250ms$. This parameter, jointly with the maximum available robot velocities and the desired precision, determine the 3D velocity-time grid size. In this work, we have chosen to divide the velocity space $\mathcal{V}$ considering the maximum angular and velocity accelerations for the sampling time used, and the time dimension with respect to the sampling time. The time horizon considered during simulation ($20s$) is not a time horizon imposed in the method, it corresponds to the workspace horizon considered for modeling the obstacles in the field of view of the sensors. The maximum velocities for the robot are $v_{max} = 1.5\ m.s^{-1}$ and $[w_{min}, w_{max}] = [-1, 1]\ rad.s^{-1}$s.

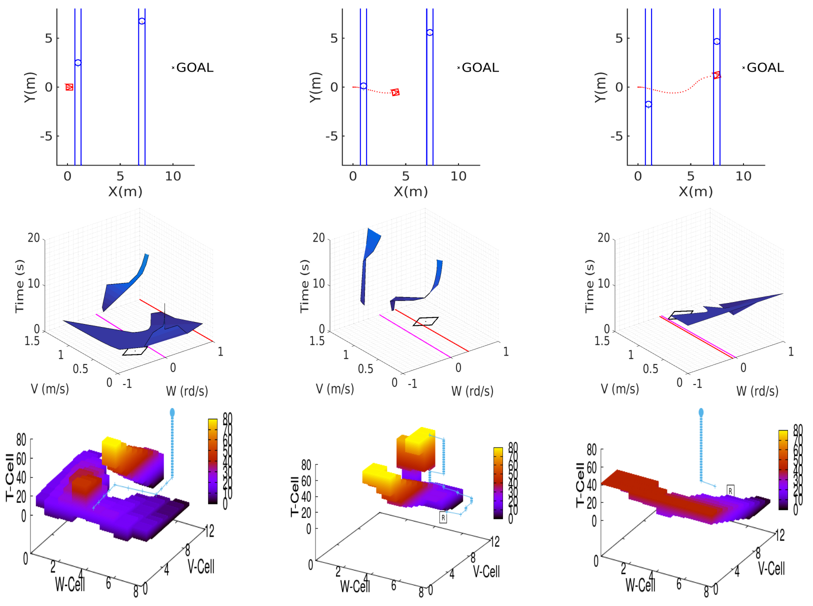
6.3 Simulation results

Several simulations in randomly generated scenarios have been performed to test robot navigation under different conditions. The performance of the planner is evaluated in terms of safety and time-to-goal criteria [6].

[6]A video showing different simulations can be accessed from `https://youtu.be/ODkTfR1JuR8`



(a) *Conservative* strategy. Any velocity which leads to collision at any future time is forbidden for planning. Thus, the planner computes motions above the first *DOVT* and around the second one. As a result, the robot deviates and waits until the obstacles pass, leading to longer trajectories.



(b) *Time-Aware* strategy. Velocities between the *DOVT*s can be selected for planning, which lead the robot to pass before both obstacles.

Figure 6.3: Navigation strategies. The workspace, which shows the trajectory followed by the robot and the moving obstacles, and the situation represented both in *DOVTS* and in the velocity-time grid are shown at different instants during navigation. The sequence of velocities computed in *FVT* from the current node (R) towards the goal are also plotted (blue line). Using information about time to collision (*Time-Aware* strategy) allows the robot reaching the goal in near-optimal time.

Two scenarios have been chosen for evaluation: the first one more structured, the *road-crossing*, in which the robot crosses an area where obstacles moving in linear trajectories traverse the scenario from one extreme to the other in both directions; in the second one, the *non-linear*, the obstacles move in random directions following non-linear trajectories.

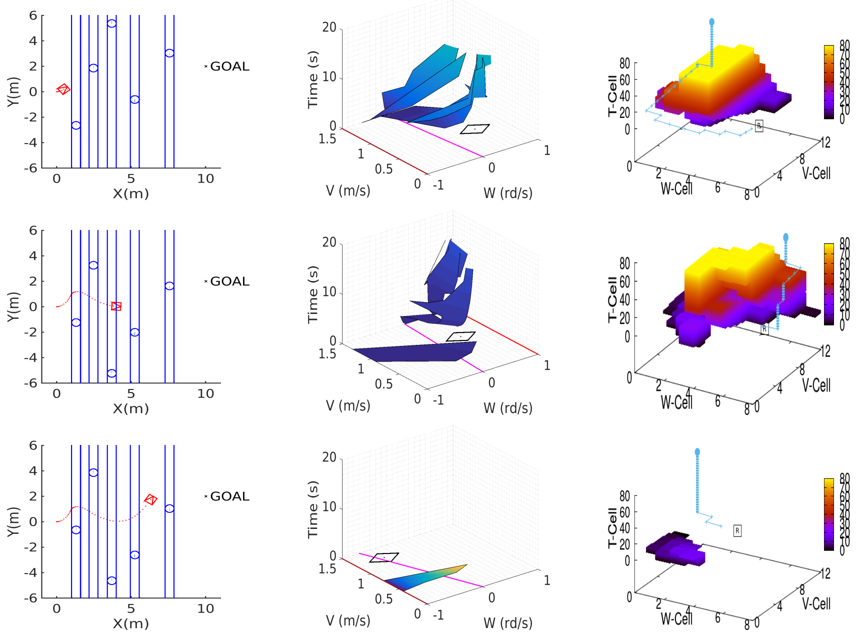
In addition, the number of moving obstacles and their velocities are changed so that we cover a range of conditions to generalize the evaluation performed. The density of moving obstacles is kept during all the simulations, in order to maintain the complexity of the scenario for navigation. For each condition we run 10 simulations, and for each same scenario we execute both *Time-Aware* and *Conservative* strategies of Section 6.2.2, to emphasize the improvement when time to collision is available for planning. For the *road-crossing* scenario, where obstacles move with non-zero linear velocity, the conditions defined are $V1 = \{\omega = 0, v = 0.2\}$ and $V2 = \{\omega = 0, v = 0.5\}$. In the *non-linear* scenario, the conditions established are $V1 = \{\omega = -0.15..0.15, v = 0.2\}$ and $V2 = \{\omega = -0.38..0.38, v = 0.5\}$. In both cases, the number of obstacles considered are 5 and 10.

Figure 6.4 shows several snapshots during simulation of the two scenarios designed. The trajectory followed by the robot and the collision band of each obstacle in the workspace are plotted, together with the obstacles modeled in *DOVTS* and the velocity-time grid used for planning. Once the information about the environment is modeled in *DOVTS*, it is mapped into the velocity-time grid.

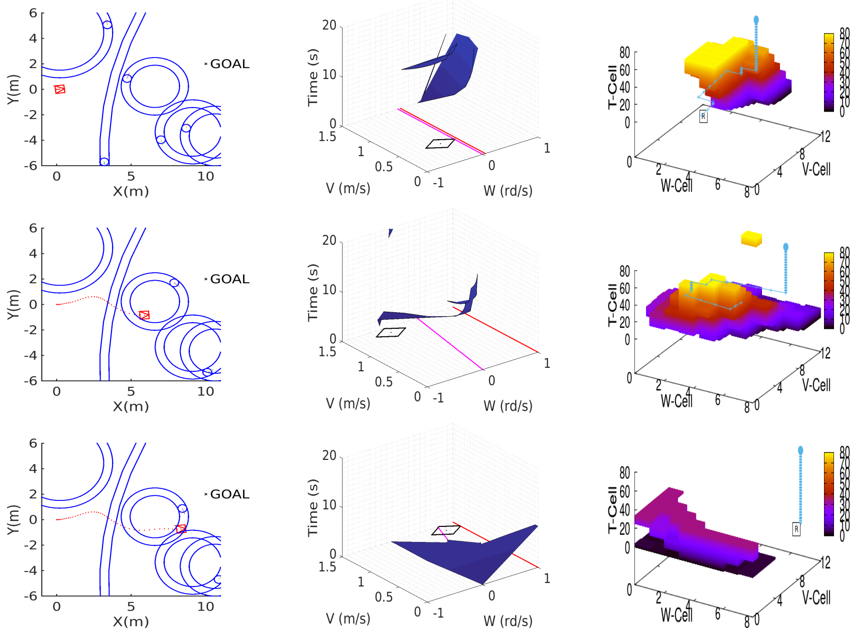
Figure 6.5 shows the time needed to traverse the scenario with respect to the optimal case, i.e. in absence of obstacles, comparing both *Time-Aware* and *Conservative* strategies, for each kind of scenario considered. Figures 6.6 and 6.7 illustrate the profiles for linear velocity and distance to the obstacles. As expected, *Time-Aware* strategy leads to quicker trajectories. The mean distance to obstacles is similar with both strategies for few obstacles. Although as the number of obstacles increases, this distance obviously decreases given the lower free space for the robot to move.

In general, collisions may appear during navigation. Table 6.1 shows the number of collisions produced out of the total simulations performed (10). The number of collisions using *Conservative* strategy are in general higher than with *Time-Aware* strategy. This is because, even though the method restricts the velocities which may lead to collision, this fact produces a great reduction of the free space to plan safe commands. In the *road-crossing* environment appear more collisions than in the *non-linear* case. In such a structured scenario, the robot has little room for crossing among the obstacles, mainly in the case of high density of obstacles and relative high obstacle velocities. It may get blocked in a situation where there is no opportunity for the robot to cross and reach the goal, ending in collision.

It can be concluded that, in general, the *Time-Aware* strategy performs better in the considered scenarios than the *Conservative* one, both in terms of safety and lower time-to-goal. This is because this strategy considers more velocity commands that can be selected to move among the obstacles. However, in very dense dynamic scenarios, the *Time-Aware* strategy may result more risky than the *Conservative* strategy. During robot navigation, *unnatural* motions, as spinning or turning back, are observed sometimes. This is due to the fact that the robot needs time to reach a velocity given its dynamic constraints,

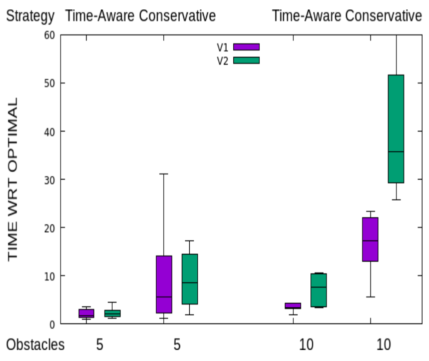


(a) *Road-crossing* scenario.

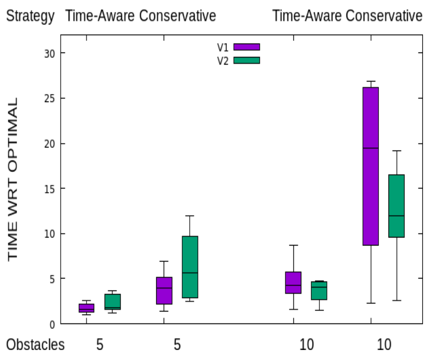


(b) *Non-linear* scenario.

Figure 6.4: Scenarios for evaluation. Simulations with 5 obstacles moving with velocities in set $V1$ and executing *Time-Aware* strategy. Workspace with the robot trajectory and the obstacle's collision bands, the *DOVTS* space and the velocity-time grid at different instants during robot navigation are illustrated. The planner searches for commands in *FVT* to drive the robot towards the goal. The goal trajectory is represented in *DOVTS* as the set of velocities which describe the circumference arc towards the goal (magenta line in the images). The goal velocity $\mathbf{v}_g$ is the maximum value of the set within bounds. The color of the cells in the grid identify the time to collision with an obstacle.

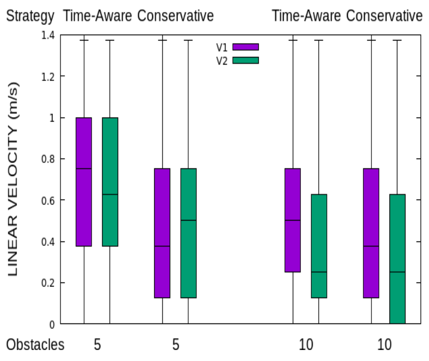


(a) *Road-crossing* environment.

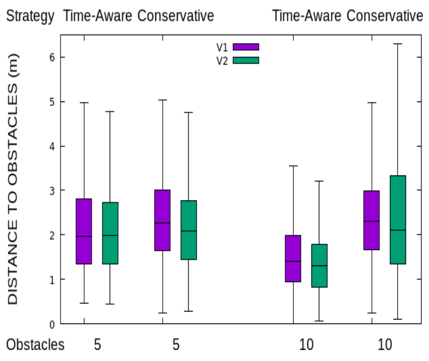


(b) *Non-linear* environment.

Figure 6.5: Time cost with respect to optimal case.

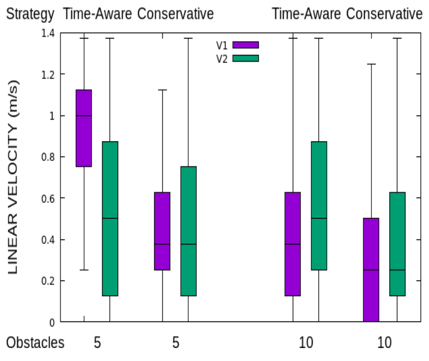


(a)

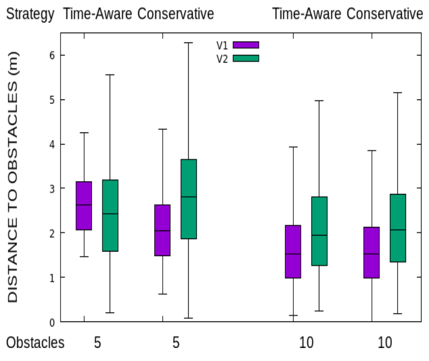


(b)

Figure 6.6: *Road-crossing* environment. (a) Linear velocity and (b) distance to obstacles during simulation, comparing *Time-Aware* and *Conservative* strategies.



(a)



(b)

Figure 6.7: *Non-linear* environment. (a) Linear velocity and (b) distance to obstacles during simulation, comparing *Time-Aware* and *Conservative* strategies.

Table 6.1: Number of collisions during simulation.

Scenario	*Road-crossing*				*Non-linear*			
Obstacles	5		10		5		10	
	v1	v2	v1	v2	v1	v2	v1	v2
Time-Aware	0	0	1	4	0	0	1	1
Conservative	2	2	3	2	0	2	1	2

especially when linear and angular velocities are high and the robot has to decelerate, and complete rotation is allowed when linear velocity is zero in order to explore potential free velocities to escape. This behavior could be reduced by limiting the set of highest velocities and the turning angle of the robot in this exploration. But it may result in the robot not finding ways of escape due to this limitation. Further work should be devoted to analyze and improve all these issues.

6.4 Conclusions

The planning technique presented in this chapter implements an A*-like algorithm in the *DOVTS* space. It optimizes a function that balances the time to goal and the time to collision to obtain a motion plan. In order to emphasize the differences when using *DOVTS* or *DOVS*, two alternatives for the algorithm are compared, *Time-Aware* and *Conservative*. The first one uses explicitly the remaining time until collision, providing quicker trajectories towards the goal among the obstacles. The second strategy avoids selecting any velocity which leads to collision at any time, similarly as using *DOVS*, thus restricting the set of velocities to plan a motion.

A general conclusion from the different techniques presented based on *DOVS* and *DOVTS*, is that, in general, planning in the Velocity-Time space leads to lower time-to-goal trajectories. But in dense and very dynamic scenarios provoke more risky motions since more velocities are free to be chosen to maneuver among the moving obstacles, increasing this way potential collisions.

Improving the technique to reduce the number of collisions in difficult situations found in the experiments, together with an analysis of the different options to smooth the *unnatural* motions observed during navigation are inherent issues during the development of the thesis.

Chapter 7

Conclusions

Through this thesis we have addressed several aspects involved in achieving autonomous robotic systems. The Introduction presented the main objectives to accomplish: a seamless localization in challenging environments, and a safe trajectory planning in dynamic environments. Providing a context of application, these objectives contributed to establish specific objectives to tackle. In Chapter 3, we developed localization and navigation techniques for deploying a car-like robot in an industrial logistic plant. As discussed earlier, this scenario raised some difficulties in computing the robot localization since the scenario comprises indoor and outdoor zones. Chapter 4 introduced a methodology for representing the environment of a differential-drive robot by computing free and collision-leading velocities with the obstacles around. This modeling approach acts as a framework for defining the planning and navigation approaches in Chapters 5 and 6.

Next, more detail is given to the main outcomes obtained:

- A localization system which integrates different techniques to obtain an estimation of the robot localization without discontinuities. The test scenario contains indoor and outdoor zones. Indoors, where any localization method described in the literature could be used, we employ the particle filter method proposed by [Fox03] with some adjustments of the parameterization which helped to better adapt the method to the scenario. The sensors used indoors are a laser rangefinder and an IMU/Odometry. Outdoors, the estimation of the robot position is obtained directly from GPS measurements and the IMU/Odometry. However, the robot orientation cannot be obtained from these data due to the magnetic material stored in the logistic plant and the low frequency of the GPS that is required to provide this information. Thus, the robot orientation should be estimated by integrating several GPS position data whenever the robot moves in a straight line. In addition, we perform an observability analysis of the GPS-based localization method, which assures the convergence and bounded uncertainty of the robot estimation under some reasonable assumptions. During the indoor-outdoor transition and vice versa, the applied approach computes the quality of the measurements, which are obtained from all the available sensors indoors and outdoors, as a function of their uncertainty, in order to have a smooth change between the different zones to avoid localization jumps. An exhaustive evaluation of the whole system, which also incorporates a reactive navigation module, is performed both in simulation and experimentation.

- Definition of a model, the *Dynamic Object Velocity-Time Space DOVTS*, to represent the environment for a differential-drive robot. The velocity and trajectory of the obstacles are taken into account to determine the set of velocities for the robot that might eventually lead to collision with the obstacles, as well as the collision time. These data are depicted in the Control space of the robot, the Velocity space, to discriminate between free and forbidden velocities. Some properties and characteristics are extracted from mapping the obstacles in the Velocity space of the robot, which settle the basis for the planning and navigation approaches defined within this thesis. A complexity analysis ends this definition with respect to other techniques in the literature. Another relevant aspect for planning is the computation of extremal (bang-bang) controls, bounded by the acceleration constraints, to obtain near-optimal time-to-goal trajectories. This fact results in the robot capacity to describe shaped paths like linear, circular, clothoid and anti-clothoid curves.

- A planning and navigation technique that uses the *DOVTS* model. The high-level navigation strategies used for locally planning the robot trajectories to avoid an obstacle are simple: robot passing before or after the obstacle. To achieve this, the robot situation with respect to the obstacle is identified and then, several variables are proposed as decision parameters for the motion strategy. The decision variables represent relevant information from the obstacle mapped in the Velocity space of the robot, and they allow to determine if the robot can pass the obstacle, or it should slow down, or it is in a dangerous zone like inside the collision band of the obstacle (area swept by the obstacle while moving). This methodology computes trajectories in a medium/long term not only for the next sampling time. This way, lower time-to-goal trajectories are obtained as compared to other techniques found in the literature, which only evaluate the commands for the next short-term period. The evaluation of the method introduces varying scenarios with moving and static obstacles, extensively in simulation and in real-world experimentation.

- Implementation of a planning and navigation technique based on the *DOVTS* model, which uses the time information explicitly. The Velocity-Time space is discretized in cells, thus having free and forbidden cells for planning, and an *A*-like* algorithm is proposed to compute the trajectories balancing safety (distance to the Velocity-Time obstacles *DOVT*) and time-to-goal criteria. This approach confers more maneuverability for the robot. Therefore, in order to show the improvement, the evaluation of the algorithm is performed considering two approaches: leaving cells under *DOVT* surfaces as free or marking them as forbidden. The latter is equivalent to using the *DOVTS* projection, like the approach stated in the previous point. As expected, lower time-to-goal trajectories are obtained when planning in *DOVTS*, but at the expense of increasing the computational cost. This may also result in a more risky robot behavior particularly due to the increase in the set of free velocities to be chosen from.

Some collisions are observed from the evaluation of the two planning techniques proposed in this thesis. Several circumstances explain this result: i) There are situations where the robot is trapped near static obstacles, this fact suggests incorporating high-level policies to avoid those areas in advance. ii) Other scenarios where unexpected obstacles might eventually appear in the scenario, thus the robot's kinodynamic constraints do not give enough time for the robot to avoid the collision. iii) Scenarios very structured with obstacles moving at high speed, which limits the free space for the robot to select velocities in such a way that there is no assurance in reaching the goal safely.

In addition, we should comment on the *unnatural* motions observed during navigation. Spinning motions appear when the robot executes high linear and angular velocities followed by slowing down velocities, whereas the turning back behaviors occur when the robot is stopped and searches for free velocities in a different direction of motion. Restricting the highest linear and angular velocities may help in the first case, again at the cost of limiting free Velocity space for planning. In the latter case, we could provide the robot with a high-level indication for motion direction.

7.1 Future work

Several lines for future research are open as a consequence of the outcomes obtained in this thesis. As previously stated, many works in the literature use the *Velocity Obstacle* formulation to represent the environment for holonomic robots, which has been applied extensively to different kinematic models of robots and applications. Therefore, it seems that the next step entails comparing any of these methods, like the approach in [AMBR$^+$13], with respect to an explicitly focused non-holonomic model, like the one presented in this thesis.

In a non-cooperative multi-robot context, where several mobile robots can make decisions without communicating any information, techniques developed within this work can be evaluated to study the limitations or virtues of our proposals by their direct application without inserting any collaboration rule. From the whole system's navigation behavior, some proactive decision-making criteria could be incorporated within the context that the robots should consider the reciprocity for the collision avoidance (assuming that all the robots use the same navigation algorithm). Within the available options in the literature, following the approach from [vdBGLM11] and derivative publications results appealing, where a sharing of the effort for the collision avoidance is proposed. Usually, a fixed percentage of the effort is determined in advance. This leads to suggest a more sophisticated way to assign the effort by, for example, an utility function which assesses certain parameters during navigation to give priority to some robots. In addition, this methodology would help to solve some unusual behaviors observed in symmetrical scenarios.

The work developed here considers the other moving entities as robots. However, for a real coexistence with humans, it is mandatory to understand human reaction to robot motions and develop techniques to smoothly include robots in our daily life.

Chapter 8

Conclusiones

A lo largo de esta tesis hemos abordado diferentes aspectos con el objetivo de conseguir sistemas robóticos autónomos. La Introducción presenta los objetivos principales que se desean cumplir: conseguir una localización continua en entornos complejos, y una planificación de trayectorias seguras en entornos dinámicos. Estos objetivos generales pretenden proporcionar un contexto de aplicación que ayude a establecer objetivos más concretos. En el Capítulo 3 se desarrollan técnicas de localización y navegación para el despliegue de un robot tipo coche en una planta industrial. Como ya se ha mencionado, este escenario plantea ciertas dificultades a la hora de calcular la localización del robot dado que contiene zonas de interior y exterior. El Capítulo 4 presenta una metodología para representar el entorno de un robot diferencial, calculando velocidades libres y velocidades que llevan a colisión con los obstáculos, estableciendo así un marco formal sobre el que definir los métodos de planificación y navegación desarrollados en los Capítulos 5 y 6.

A continuación, se proporciona una descripción más detallada sobre los resultados obtenidos:

- Un sistema de localización que integra diferentes técnicas para obtener una estimación de la localización del robot sin discontinuidades. El escenario de prueba contiene zonas de interior y exterior. En el interior, donde cualquier método de localización existente en la literatura puede utilizarse, empleamos el método de filtro de partículas propuesto por [Fox03], con algunos ajustes sobre la parametrización para adaptar el método al escenario. Los sensores utilizados en el interior son un láser y un IMU/Odometría. En el exterior, la estimación de la posición del robot es obtenida directamente a partir de las medidas GPS y el IMU/Odometría. Sin embargo, la orientación del robot no puede conseguirse de estos datos debido al material magnético almacenado en la planta logística y la baja frecuencia del GPS cuando se requiere que proporcione esta información. Por tanto, la orientación del robot debe estimarse integrando varias posiciones GPS siempre que el robot se mueva en línea recta. Además, realizamos un análisis de observabilidad del método de localización basado en GPS, que asegura la convergencia e incertidumbre limitada de la estimación del robot bajo ciertas asunciones razonables. Durante la transición interior-exterior y viceversa, el método considera la calidad de las medidas obtenidas de todos los sensores disponibles tanto en interior como exterior, calculándola como una función de su incertidumbre, para tener un cambio suave entre las diferentes zonas para evitar saltos en la localización. Una evaluación exhaustiva del sistema completo, el cuál incorpora también un módulo de navegación reactiva con maniobras, se realiza tanto en simulación como en experimentación real.

- Definición de un modelo, *Dynamic Object Velocity-Time Space DOVTS*, para representar el entorno dinámico de un robot no-holónomo, en nuestro caso un robot diferencial. Se consideran tanto la velocidad como la trayectoria de los obstáculos para determinar el conjunto de velocidades que llevan al robot a colisionar con los obstáculos, así como el tiempo de colisión. Estos datos se plasman en el espacio de Control del robot, el espacio de Velocidad, que divide el espacio en velocidades libres y prohibidas para el robot. Algunas propiedades y características se extraen del modelo resultante, que establecen las bases para las técnicas de planificación y navegación definidas en la tesis. Un análisis de complejidad con respecto a otras técnicas en la literatura cierra esta formulación. Otro aspecto relevante para la planificación es el cálculo de controles extremos (bang-bang), limitados por las restricciones de aceleración, para obtener trayectorias cuasi-óptimas en tiempo hacia el objetivo. Esto lleva a que el robot describa curvas de tipo lineal, circular, clotoides y anti-clotoides.

- Una técnica de planificación y navegación que utiliza el modelo *DOVTS*. A alto nivel, las estrategias de navegación que se utilizan para planificar las trayectorias del robot localmente son sencillas y comunes: pasar por delante o detrás de un obstáculo. Para lograrlo, primero identificamos la situación del robot con respecto al obstáculo, para después proponer varias variables como parámetros de decisión sobre la estrategia de movimiento a ejecutar. Las variables de decisión representan información relevante del obstáculo modelado en el espacio de Velocidad del robot, y permiten determinar si el robot puede superar el obstáculo, o debería frenar, o si está en una zona peligrosa como dentro de la banda de colisión del obstáculo (área barrida por el obstáculo cuando se mueve). Esta metodología calcula trayectorias en un plazo de tiempo medio/largo, no sólo para el siguiente periodo de tiempo. De esta manera, se obtienen trayectorias de menor tiempo hacia el objetivo con respecto a otras técnicas encontradas en la literatura, que sólo evalúan los comandos para el siguiente periodo. La evaluación del método se realiza en escenarios diferentes con obstáculos móviles y estáticos, tanto en simulación como experimentación real.

- Implementación de una técnica de planificación y navegación basada en *DOVTS*, que utiliza explícitamente la información del tiempo. El espacio de Velocidad-Tiempo se discretiza en celdas, dando lugar a celdas libres y ocupadas para la planificación. Se propone un algoritmo A* adaptado al modelo para calcular las trayectorias, evaluando criterios de seguridad (distancia a los obstáculos de

Velocidad-Tiempo *DOVT*) y tiempo hacia objetivo. Utilizar la información del tiempo permite más maniobrabilidad para el robot. Para mostrar la mejora, la evaluación del algoritmo se hace considerando dos enfoques: mantener las celdas bajo la superficie de cualquier *DOVT* como libres o marcándolas como prohibidas. Esta última es equivalente a utilizar la proyección de *DOVTS*, como en el método del punto anterior. Como era de esperar, se consiguen trayectorias de menor tiempo cuando se planifica en *DOVTS*, pero a costa de incrementar el coste computacional. Además, el comportamiento del robot puede resultar más arriesgado debido al aumento de velocidades libres para escoger.

Algunas colisiones se observan a partir de la evaluación de las técnicas propuestas en la tesis. Varias circunstancias explican este resultado: i) Hay situaciones en las que el robot está atrapado cerca de obstáculos estáticos, lo que sugiere incorporar políticas de alto nivel para evitar aquellas áreas de antemano. ii) Otras situaciones donde aparecen obstáculos inesperados en el escenario, en las que las restricciones dinámicas del robot no le permiten tener tiempo suficiente para evitar la colisión. iii) Escenarios muy estructurados y obstáculos moviéndose a alta velocidad, limitando el espacio libre para que el robot pueda seleccionar velocidades de tal manera que no se asegura que pueda alcanzar el objetivo de forma segura.

Por otro lado, debemos comentar los movimientos antinaturales observados durante la navegación. Los movimientos giratorios aparecen cuando el robot ejecuta velocidades lineales y angulares altas seguido por velocidades de frenado, mientras que los giros hacia atrás ocurren cuando el robot está parado y busca velocidades libres hacia una dirección de movimiento diferente. Restringir las velocidades lineales y angulares más altas puede ayudar en el primer caso, a costa de limitar el espacio de Velocidad para planificación. En el último caso, se pueden incorporar indicaciones de alto nivel para definir una dirección clara de movimiento.

8.1 Trabajo futuro

Varias líneas de investigación quedan abiertas como resultado del trabajo realizado en la presente tesis. Como ya se ha comentado anteriormente, varios trabajos en la literatura utilizan la formulación *Velocity Obstacle* para representar el entorno dinámico de robots holónomos, y lo aplican a diferentes modelos cinemáticos de robots. Por ello, el siguiente paso nos llevaría a la comparación de cualquiera de estos métodos, como el enfoque propuesto en [AMBR$^+$13], con el modelo presentado aquí, definido explícitamente para robots no-holónomos.

En un contexto multi-robot no-cooperativo, en el que varios robots móviles pueden tomar decisiones sin intercambiar información, las técnicas desarrolladas en la tesis pueden ser evaluadas para estudiar las limitaciones o virtudes de las mismas, aplicándolas directamente sin añadir ninguna regla de colaboración. A partir de la navegación resultante del sistema completo, podríamos incorporar algunos criterios para la toma de decisión proactiva, en la que los robots deben considerar la reciprocidad para la evitación de colisión (asumiendo que todos los robots utilizan el mismo algoritmo de navegación). En este contexto, de entre las opciones disponibles en la literatura, resulta interesante el enfoque propuesto en [vdBGLM11] y publicaciones derivadas, donde se establece una compartición del esfuerzo para la evitación de colisión. Normalmente, se define un porcentaje fijo de esfuerzo. Sin embargo, proponer un método más sofisticado de asignación de esfuerzo podría mejorar el comportamiento global del sistema, por ejemplo, mediante una función de utilidad que evalúe ciertos parámetros durante la navegación para dar prioridad a algunos robots. Además, esta metodología podría resolver ciertos comportamientos inusuales observados en escenarios simétricos.

El trabajo desarrollado en la presente tesis considera las demás entidades móviles como robots. Sin embargo, para una coexistencia real con humanos, es imprescindible entender la reacción humana a los movimientos de los robots y desarrollar técnicas adecuadas para incluir robots en nuestro día a día.

Appendix A

Maneuvering

As stated in section 3.4.1, the kinematic model of a car-like robot restricts the turning radius capability of the robot, which may lead not to reach some desired positions. Moreover, unforeseen obstacles appearing in the scenario during navigation may force the robot to change motion in a non-forward direction to avoid collision.

Figure A.1 shows some considerations for a car-like robot to maneuver. Figure A.1a shows the optimal values of ϕ_{goal} to reach a goal in the absence of obstacles computed using the equation in line 6 of Algorithm 3.2 (section 3.4.1), where $(goal_x, goal_y)$ is the position of the goal relative to the robot reference frame, and D is the distance between the two axes of the robot. White zones are forbidden as they require steering angles out of the kinematic bounds, which in our case is $\phi_{max} = 0.4\ rad$. Figure A.1b shows the zone where the obstacles are unavoidable, no matter what steering angle is used if the robot keeps moving forward. This zone is defined by R_{min}, the minimum turning radius of the center of the robot, and R^*, the minimum turning radius of the furthest point of the shape of the robot.

To overcome these limitations in the kinematic abilities of a car-like robot, a reactive maneuvering technique is proposed in the next section.

A.1 Maneuvering

The reactive maneuvering technique proposed in this section is designed to overcome the limitations in the movement ability of the robot given by the kinematic constraints. It integrates with the NDW reactive technique of section 3.4.1. Optimal solutions for maneuvering in clear navigation zones or with restricted obstacles are well known. However, general case solutions rely on the exploration of the combination of available maneuvers, which require computationally expensive algorithms. We propose a real-time reactive solution to deal with maneuvering in zones with unrestricted obstacles, so that motion can be recomputed when the environment suddenly changes.

We adopt a state machine based design (Figure A.2) where, by default, the obstacle avoidance method NDW is used to control the motion towards the goal. Whenever one of the two above-mentioned problematic situations is detected (an unreachable goal or an unavoidable obstacle), the navigation strategy is changed.

Unreachable Goals

When the goal of the robot is inside an unreachable zone, the robot must perform forward and backward movements in such a way that the goal moves to a reachable zone in the robocentric reference (as depicted in Figure A.1a). We also consider as unreachable any goal behind the robot, so that the first movement is to turn the robot around to face the goal.

To make this movement as short as possible, the robot moves backwards with the maximum steering angle allowed. Depending on the position of the goal, the sign of the steering angle is set so that the robot is reoriented towards the goal.

$$\phi_{turn} = \begin{cases} \phi_{max} & \text{if } goal_y < 0 \\ -\phi_{max} & \text{otherwise} \end{cases} \qquad v_{turn} = -v_{slow} \tag{A.1}$$

The speed profile during the turning maneuver is kept constant to a low value, v_{slow}, so that the movement is safe and can be stopped at any time. Note that, because of the presence of an obstacle in the turning trajectory of the robot, the desired movement to get the goal to a reachable position cannot be completed. In this case, when the distance from the robot to the obstacle in the trajectory is under a safe distance, the navigation state changes to TURN FORWARDS (see Figure A.2). In this situation, the steering angle and the speed change their sign with respect to (A.1). This navigation mode continues alternating backward and forward motions until the goal is again in a reachable position with respect to the robot.

Unavoidable Obstacles

When an obstacle appears in the unavoidable zone (see Figure A.1b) a backwards maneuver is required to move this obstacle out of the unavoidable zone and to keep the robot going to the goal. As in the case of unreachable goals considered above, to make this maneuver as short as possible, the steering angle is set to the maximum value. In this case, to discriminate the direction of the maneuver, we use ϕ^*, the optimal steering angle computed by Algorithm 3.2 if we consider only the avoidable obstacles. The idea behind this value ϕ^* is to know which side of the unavoidable obstacles it is better to pass by.

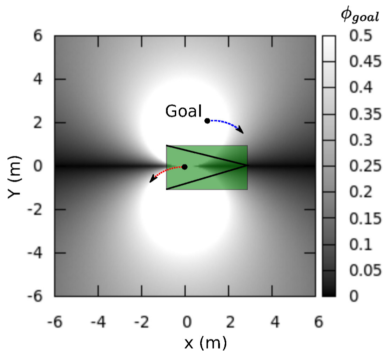
$$\phi_{maneuver} = \begin{cases} \phi_{max} & \text{if } \phi^* < 0 \\ -\phi_{max} & \text{otherwise} \end{cases} \qquad v_{turn} = -v_{slow} \tag{A.2}$$

$$\phi^* = \textsc{Navigation}(goal, AvoidableObstacles) \tag{A.3}$$

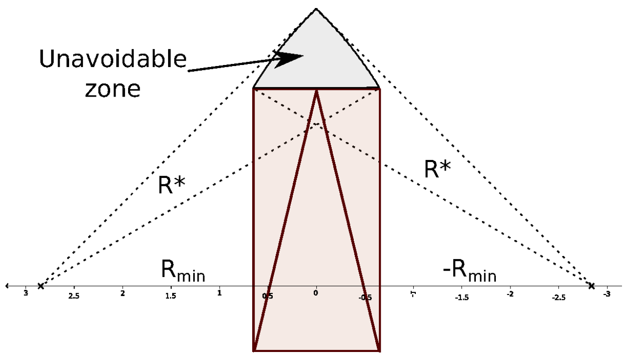
The speed v_{turn} is kept at low values to make the maneuver safer, as in the case of unreachable goals.

In the case that, because of the presence of obstacles, the maneuver cannot be completed backwards, the direction is switched (from MANEUVER BACKWARDS to MANEUVER

FORWARDS in Figure A.2) and the maneuver continues. In this new situation, the commands are also obtained from equations (A.2), but changing the sign. This maneuvering mode continues until there are no more unavoidable obstacles and thus the normal GO TO GOAL navigation is resumed.



(a) Required steering angle to reach a goal depending on its position relative to the robot. The lighter the point, the bigger the steering angle needed to reach the goal. If the goal is in the white zone, a maneuver (in red) at the maximum steering angle is performed so that the goal leaves the forbidden zone (in blue).



(b) Obstacles in unavoidable (in grey) zones require maneuvering. This zone is defined by R_{min}, the minimum turning radius of the robot, and R^*, the turning radius of the furthest point of the robot.

Figure A.1: Analysis of the maneuver capabilities of a car-like vehicle.

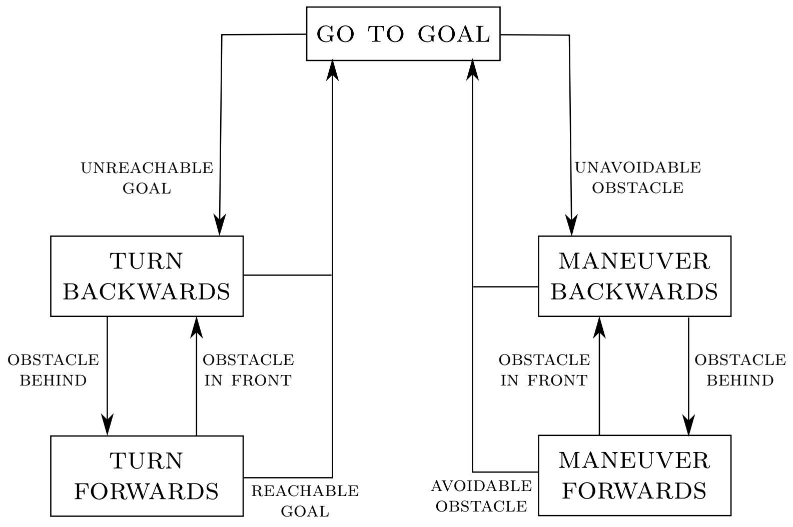


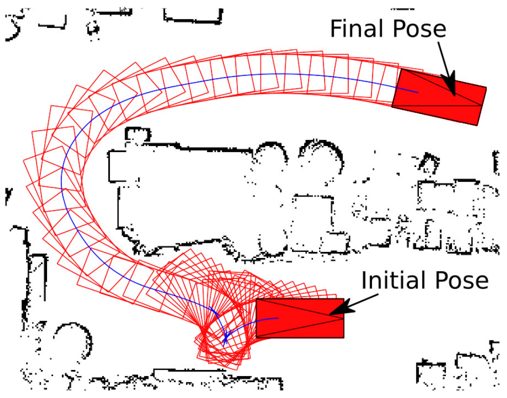
Figure A.2: State machine for navigation that switches between navigation modes: go-to-goal, maneuvering and turning.

A.2 Results

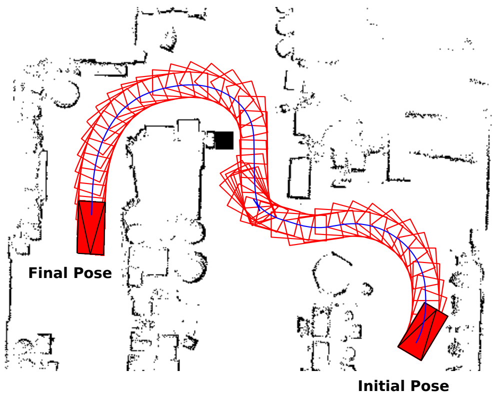
To test the maneuvering technique, we use the scenario shown in Figure 3.23. Table A.1 shows the statistical results of the 50 random goals experiment. Turn maneuvers refers to maneuvers needed to transform unreachable into reachable goals (see Figure A.3a), avoidance maneuvers are produced by obstacles in the way that are not considered in the path planning (see Figure A.3b), and direction changes are caused when the maneuvers are divided into different movements because of the presence of obstacles. We can verify that, provided there is enough maneuvering space at goal locations, the robot can move around the scenario without deadlocks and oscillations.

Table A.1: Statistical navigation results

Goals Assigned	50
Goals Reached	50
Turn Maneuvers	32
Avoidance Maneuvers	33
Direction Changes	118



(a) Turn maneuver when the goal is behind the robot. The robot moves forward and backward so as to make the goal reachable and to avoid colliding with the obstacles.



(b) Trajectory of the robot during an avoidance maneuver where an unmapped obstacle (black square) is detected and avoided.

Figure A.3: Examples of maneuvering trajectories with unreachable goals and unavoidable unmapped obstacles.

Appendix B

Computation of the robot collision velocities

B.1 Obstacle linear trajectories

Figure 4.1b in section 4.3 illustrates a moving object describing a linear trajectory, positioned at locations ($\mathcal{O}_i^1$ and $\mathcal{O}_i^2$). The positions are related to the instants at which the robot reaches points P_{1j} and P_{2j}, following a circular trajectory γ_j. P_{1j} is computed as the position at which the robot should arrive after the object has just passed it, at time t_{1j} ($\mathcal{O}_i(t_{1j}) = \mathcal{O}_i^2$). P_{2j} is the position at which the robot should arrive just before the object reaches it, at time t_{2j} ($\mathcal{O}_i(t_{2j}) = \mathcal{O}_i^1$).

From object location $\mathcal{O}_i(t_0) = (x_o, y_o, \phi_o)$ in the robot local reference and its velocity $\mathbf{v}_o$, the points of collision $P_{1j}(x_{1j}, y_{1j})$ and $P_{2j}(x_{2j}, y_{2j})$, and the corresponding times t_{1j} and t_{2j} are calculated by solving the following equations characterized by the curvature radius r_j of path γ_j and the center $(0, y_j)$ in the robot reference frame R. For a path γ_j, t_{ij} $(i = 1, 2)$ is computed from equations B.1, obtaining two solutions for each collision point P_{ij}.

$$\begin{aligned}
x_{ij} &= x_0 + v_0 \cos(\phi_0) t_{ij} \\
y_{ij} &= y_0 + v_0 \sin(\phi_0) t_{ij} \\
{r_j}^2 &= x_{ij}^2 + (y_{ij} - y_j)^2 \\
t_{ij} &= (-B \pm \sqrt{B^2 - 4AC})/2A \\
A &= v_0^2 \\
B &= 2v_0(x_0 \cos(\phi) + y_0 \sin(\phi)) - 2v_0 \sin(\phi) y_j \\
C &= x_0^2 + y_0^2 + y_j^2 - 2y_0 y_j - r_j^2.
\end{aligned} \tag{B.1}$$

From xy-coordinates of points P_{1j} and P_{2j}, and times t_{ij}, the velocities $\mathbf{v}_{ij}$ are computed, as in section 4.3.1. Two cases can occur. When the robot is out of the collision band, we select t_{ij} as the solution corresponding to the first intersection point (the lower value of t_{ij}) for both P_{1j} and P_{2j}. When the robot is in the collision band, the strategy

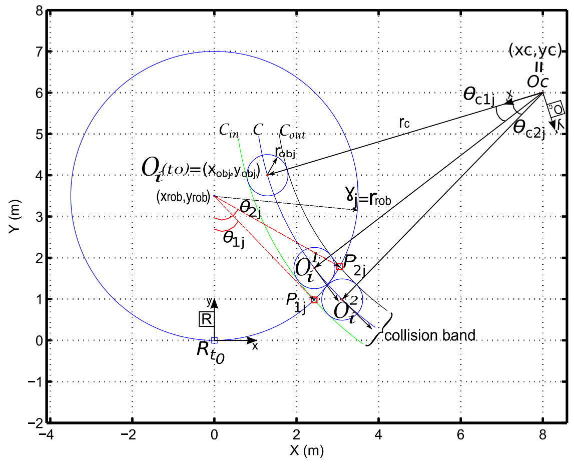


Figure B.1: Collision band, path γ_j with curvature radius r_{rob} and collision points P_{1j} and P_{2j} with a circular object that moves along a circular trajectory in the robocentric (R) Configuration space.

is to escape from the band to avoid a collision. In this case only the escape point P_{2j} and velocity $\mathbf{v}_{2j}$ are calculated, using the previous equations. A lower velocity would result in collision.

B.2 Obstacle circular trajectories

In a similar way, the intersection points $P_{1j}(x_{1j}, y_{1j})$ and $P_{2j}(x_{2j}, y_{2j})$ between a circular robot trajectory and a circular collision band swept by the obstacle (delimited by $\mathcal{C}_{in}$ and $\mathcal{C}_{out}$), and the corresponding times t_{1j} and t_{2j}, are computed. These points are computed from intersections between circles C_{in} and C_{out} and the robot circular path γ_j, following equations B.2 (see Fig.B.1).

$\mathcal{O}_i^1$ and $\mathcal{O}_i^2$ are the object positions for computing the collision times t_{2j} and t_{1j}, respectively. Applying tangency conditions between robot trajectory γ_j and the circles corresponding to the object in both positions (equation B.3), the times to collision are calculated from the angular displacement θ_{cij} and the angular velocity of the object ω_{obj} (equations B.4). Then, the velocities $\mathbf{v}_{ij}$ are computed as in section 4.3.1.

$$\begin{aligned}
(x - x_c)^2 + (y - y_c)^2 &= r_{C_{out}}^2 = (r_c - r_{rob})^2 \\
(x - x_c)^2 + (y - y_c)^2 &= r_{C_{in}}^2 = (r_c + r_{rob})^2 \\
(x - x_{rob})^2 + (y - y_{rob})^2 &= r_{rob}^2
\end{aligned} \tag{B.2}$$

$$(x - x_{obj})^2 + (y - y_{obj})^2 = r_{obj}^2 \tag{B.3}$$

$$\begin{aligned}
\theta_{cij} &= atan2(2\, x_{obj_i}\, y_{obj_i}, x_{obj_i}^2 - y_{obj_i}^2) \\
t_{ij} &= \theta_{cij}/w_{obj}, \quad i = 1, 2
\end{aligned} \tag{B.4}$$

Appendix C

Computation of time-to-goal from the current robot location

Trajectories are composed of combinations of straight lines, circular, clothoid and anti-clothoid curves. The time-to-goal functions (t_G) are computed as the sum of several terms, the switching times: time for angular acceleration (t_α), time for linear acceleration at constant angular non-zero velocity (t_{aw}), time at constant angular velocity (t_w), time for angular deceleration until alignment with the goal ($t_{-\alpha}$), time for linear acceleration once the robot is aligned (t_a), and time at maximum linear velocity towards the goal (t_v). The total time to goal in both cases is:

$$\begin{aligned} t_{Gw} &= t_\alpha + t_w + t_{-\alpha} + t_a + t_v && \text{(clothoid)} \\ t_{Gv} &= t_{aw} + t_{-\alpha} + t_a + t_v && \text{(anti-clothoid)} \end{aligned}$$

The switching time for starting angular deceleration is computed when $\theta_a = \omega_c^2/2{*}\alpha_m$ is reached, being $\omega = 0$ when the robot is aligned to the goal (x_G, y_G). The switching times are computed as follows:

a) Time for angular acceleration (t_α)/deceleration ($t_{-\alpha}$)

$$\begin{aligned} & t_\alpha = (w_\alpha - w_c)/\alpha_m \\ & \omega_\alpha = min(\omega_{top_w}, \omega_m); \quad \omega_{top_w} = w_c + \alpha_m * t_{top_w} \\ & t_{-\alpha} \simeq w_c/\alpha_m; \quad \omega_{t_{-\alpha}} = w_c - \alpha_m * t_{-\alpha} \simeq 0 \end{aligned}$$

Velocity ω_α will be the angular velocity reached (ω_{top_w}) when the value $\theta_a = \omega_c^2/2{*}\alpha_m$ with respect to the goal direction is measured or ω_m is reached, whilst v_c is kept. Then an angular deceleration starts until aligning the goal. The coordinates at the end this stage are ($\beta = \{\alpha, -\alpha\}$):

$$\begin{aligned} x_\beta &= v_c * \int_{t_0}^{t_\beta} \cos(\theta(t))dt \\ y_\beta &= v_c * \int_{t_0}^{t_\beta} \sin(\theta(t))dt \\ \theta_\beta &= w_c * t_\beta + (\alpha_m * t_\beta^2)/2 \end{aligned}$$

These are non integrable equations, so they are numerically solved.

b) Time at constant angular velocity (t_w)

$$t_w = (\theta_c - \theta_a)/\omega_c$$

Time until the orientation reaches θ_a from the initial θ_c. Both the angular and linear velocities (v_c, ω_c) are kept. The coordinates at the end of this stage are:

$$\begin{aligned} x_\omega &= (v_c/\omega_c) * \sin(\theta_c - \theta_a) \\ y_\omega &= (v_c/\omega_c) * (1 - \cos(\theta_c - \theta_a)) \\ \theta_\omega &= \theta_c - \theta_a \end{aligned}$$

c) Time at maximum linear acceleration and constant angular velocity (t_{aw})

$$\begin{aligned} t_{aw} &= (v_{aw} - v_c)/a_m; \quad v_{aw} = min(v_{top_v}, v_m) \\ v_{top_v} &= v_c + a_m * t_{top_v} \end{aligned}$$

This stage finishes when the angular position with respect to the goal direction (θ_a) is reached. The coordinates reached at the end of this stage are:

$$\begin{aligned} x_{aw} &= \int_{t_0}^{t_{aw}} v(t) * \cos(\theta(t)) \; dt \\ y_{aw} &= \int_{t_0}^{t_{aw}} v(t) * \sin(\theta(t)) \; dt \\ \theta_{aw} &= w_c * t_{aw} \end{aligned}$$

d) Time at maximum linear acceleration (t_a)

$$t_a = (v_m - v_c)/a_m; \quad v_a = v_c + a_m * t_a$$

The stage starts when the robot is aligned $(\omega = 0, \theta_a = 0)$, and accelerates up to the maximum linear velocity. The coordinates reached are:

$$\begin{aligned} x_a &= (v_c + (a_m * t_a)/2) * cos(\theta_g) * t_a \\ y_a &= (v_c + (a_m * t_a)/2) * sin(\theta_g) * t_a \end{aligned}$$

e) Time at maximum linear velocity (t_v)

At the start of this stage, the current location of the robot is given by:

$$x_c = \sum_i x_i; \quad y_c = \sum_i y_i; \quad i = \alpha, w, a_w, -\alpha, a$$

Being d_G the distance to the goal from (x_c, y_c), the time to goal and velocities are:

$$t_v = d_G/v_v; \quad v_v = v_m; \quad w_v = 0$$

Bibliography

[AAWB10] D. Althoff, M. Althoff, D. Wollherr, and M. Buss. Probabilistic collision state checker for crowded environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1492–1498, May 2010.

[AKWB12] Daniel Althoff, James J. Kuffner, Dirk Wollherr, and Martin Buss. Safety assessment of robot trajectories for navigation in uncertain and dynamic environments. *Autonomous Robots*, 32(3):285–302, 2012.

[AMBR$^+$13] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. volume 83, pages 203–216, 01 2013.

[AP99] H. Abbott and D. Powell. Land-vehicle navigation using gps. *Proceedings of the IEEE*, 87(1):145–162, Jan 1999.

[Ark98] Ronald C. Arkin. *An Behavior-based Robotics.* MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[BFAS14] S. Bouraine, T. Fraichard, O. Azouaoui, and H. Salhi. Passively safe partial motion planning for mobile robots with limited field-of-views in unknown dynamic environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3576–3582, May 2014.

[BFS12] Sara Bouraine, Thierry Fraichard, and Hassen Salhi. Provably safe navigation for mobile robots with limited field-of-views in dynamic environments. *Autonomous Robots*, 32(3):267–283, 2012.

[BK00] O. Brock and O. Khatib. Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 550–555 vol.1, 2000.

[BK07] K. E. Bekris and L. E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 704–710, April 2007.

[BM02] Devin J. Balkcom and Matthew T. Mason. Time optimal trajectories for bounded velocity differential drive vehicles. *The International Journal of Robotics Research*, 21(3):199–217, 2002.

[BMGF10] A. Bautin, L. Martinez-Gomez, and T. Fraichard. Inevitable collision states: A probabilistic perspective. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4022–4027, May 2010.

[BTK09] Kostas E. Bekris, Konstantinos I. Tsianos, and Lydia E. Kavraki. Safe and distributed kinodynamic replanning for vehicular networks. *Mobile Networks and Applications*, 14(3):292–308, 2009.

[BVdB13] Daman Bareiss and Jur Van den Berg. Reciprocal collision avoidance for robots with linear dynamics using lqr-obstacles. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3847–3853, May 2013.

[BVdB15] Daman Bareiss and Jur Van den Berg. Generalized reciprocal collision avoidance. *The International Journal of Robotics Research*, 34(12):1501–1514, 2015.

[CDPV06] Francois Caron, Emmanuel Duflos, Denis Pomorski, and Philippe Vanheeghe. Gps/imu data fusion using multisensor kalman filtering: Introduction of contextual aspects. *Inf. Fusion*, 7(2):221–230, June 2006.

[CKZ08] Nicholas Chan, James Kuffner, and Matthew Zucker. Improved motion planning speed and safety using regions of inevitable collision. In *17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control (RoManSy'08)*, July 2008.

[Cle15] Clearpath. Otto, heavyload material transporter. `http://www.clearpathrobotics.com/otto/`, 2015.

[CRS09] J. Collier and A. Ramirez-Serrano. Environment classification for indoor/outdoor robotic matching. In *Proceedings of Canadian Conference on Computer and Robot Vision*, pages 276–283, 2009.

[CT96] J. A. Castellanos and J. D. Tardós. *Laser-based segmentation and localization for a mobile robot Robotics and manufacturing: Recent trends in research and applications.* ASME Press, New York, 1996.

[FA03] T. Fraichard and H. Asama. Inevitable collision states. a step towards safer robots? In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 388–393 vol.1, Oct 2003.

[FBT97] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine*, 4(1):23–33, 1997.

[FDF01] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. In *American Control Conference, 2001. Proceedings of the 2001*, volume 1, pages 43–49 vol.1, 2001.

[Fio95] Paolo Fiorini. *Robot Motion Planning Among Moving Obstacles.* PhD thesis, Los Angeles, CA, USA, 1995. UMI Order No. GAX95-19816.

[Fox03] Dieter Fox. Adapting the sample size in particle filters through KLD-sampling. *The International Journal of Robotics Research*, 22(12):985–1003, December 2003.

[Fra98] Thierry Fraichard. Trajectory Planning in Dynamic Workspace: a 'State-Time Space' Approach. Technical Report RR-3545, INRIA, October 1998.

[Fra07] T. Fraichard. A short paper about motion safety. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1140–1145, April 2007.

[FS98] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[GES08] Dongqing Gu and N. El-Sheimy. Heading accuracy improvement of mems imu/dgps integrated navigation system for land vehicle. In *Proceedings of Position, Location and Navigation Symposium*, pages 1292–1296, 2008.

[GN15] Matthew Gadd and Paul Newman. A framework for infrastructure-free warehouse navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2015.

[GRS99] P. Goel, S.I. Roumeliotis, and G. Sukhatme. Robust localization using relative and absolute position estimates. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1134–1140, 1999.

[GSB07] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23:34–46, 2007.

[GSR09] Oren Gal, Zvi Shiller, and Elon Rimon. Efficient and safe on-line motion planning in dynamic environments. In *Proceedings-IEEE International Conference on Robotics and Automation*, pages 188–93, Kobe, Japan, May 2009.

[HK77] R. Hermann and Arthur J. Krener. Nonlinear controllability and observability. *Automatic Control, IEEE Transactions on*, 22(5):728–740, Oct 1977.

[HKLR02] David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

[HWW08] M. Hentschel, O. Wulf, and B. Wagner. A gps and laser-based localization for urban and non-urban outdoor environments. In *Proceedings of Internation Conference on Intelligent Robots and Systems (IROS)*, pages 149–154, 2008.

[JU07] Simon J. Julier and Jeffrey K. Uhlmann. Using covariance intersection for SLAM. *Robotics and Autonomous Systems*, 55(55):3–20, 2007.

[KKK$^+$08] J. Kang, D. Kim, E. Kim, Y. Kim, S. Yoo, and D. Wi. Seamless mobile robot localization service framework for integrated localization systems. In *Proceedings of 3rd International Symposium on Wireless Pervasive Computing (ISWPC)*, pages 175–179, May 2008.

[KO16] Mingeuk Kim and Jun-Ho Oh. Study on optimal velocity selection using velocity obstacle (ovvo) in dynamic and crowded environment. *Autonomous Robots*, 40(8):1459–1470, 2016.

[KUK15] KUKA. Kuka navigation solution. `http://www.kuka-robotics.com/res/robotics/Products/PDF/EN/KUKA_Navigation_Solution_EN.pdf`, 2015.

[KvdP07] M. Kalisiak and M. van de Panne. Faster motion planning using learned local viability models. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2700–2705, April 2007.

[LaV06] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[LK01] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[LLS05] Frédéric Large, Christian Laugier, and Zvi Shiller. Navigation among moving obstacles using the nlvo: Principles and applications to intelligent vehicles. *Autonomous Robots*, 19(2):159–171, Sep 2005.

[LM17] M. T. Lorente and L. Montano. Robot navigation balancing safety and time to goal in dynamic environments. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–8, Sept 2017.

[LOM18] María-Teresa Lorente, Eduardo Owen, and Luis Montano. Model-based robocentric planning and navigation for dynamic environments. *The International Journal of Robotics Research*, 37(8):867–889, 2018.

[LSSL02] F. Large, S. Sekhavat, Z. Shiller, and C. Laugier. Using non-linear velocity obstacles to plan motions in dynamic environments. In *IEEE International Conference on Control, Automation, Robotics and Vision, ICARV02*, pages 734–739, December 2002.

[MG10] Luis Martinez-Gomez. *Safe Navigation for Autonomous Vehicles in Dynamic Environments: an Inevitable Collision State (ICS) Perspective.* Theses, Université de Grenoble, November 2010.

[MGF09] L. Martinez-Gomez and T. Fraichard. Collision avoidance in dynamic environments: An ics-based solution and its comparative evaluation. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 100–105, May 2009.

[MK10] Daniel Maier and Alexander Kleiner. Improved gps sensor model for mobile robots in urban terrain. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4385–4390. IEEE, 2010.

[MLP16] T. Mercy, W. Van Loock, and G. Pipeleers. Real-time motion planning in the presence of moving obstacles. In *2016 European Control Conference (ECC)*, pages 1586–1591, June 2016.

[MM04] J. Minguez and L. Montano. Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios. *Transactions on Robotics and Automation*, 20(1):45–59, 2004.

[MM05] Javier Minguez and Luis Montano. Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios. *Robotics and Autonomous Systems*, 52(4):290 – 311, 2005.

[MMM08] Luis Montesano, Javier Minguez, and Luis Montano. Modeling dynamic scenarios for local sensor-based motion planning. *Autonomous Robots*, 25(3):231–251, 2008.

[OM05] E. Owen and L. Montano. Motion planning in dynamic environments using the velocity space. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2833–2838, Aug 2005.

[PCKL09] J. Park, J. S. Choi, J. Kin, and B. H. Lee. Moving obstacle avoidance for a mobile robot. In *2009 IEEE International Conference on Control and Automation*, pages 367–372, Dec 2009.

[PF05] S. Petti and T. Fraichard. Safe motion planning in dynamic environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2210–2215, Aug 2005.

[PSA$^+$06] E. B. Pacis, B. Sights, G. Ahuja, G. Kogut, and H. R. Everett. An adapting localization system for outdoor/indoor navigation. In *Proceedings of SPIE Proc. 6230: Unmanned Systems Technology VIII, Defense Security Symposium*, Orlando, EEUU, April 2006.

[PSBF07] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli. Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Transactions on Robotics*, 23(6):1170–1183, Dec 2007.

[PSF01] E. Prassler, J. Scholz, and P. Fiorini. A robotics wheelchair for crowded public environment. *IEEE Robotics Automation Magazine*, 8(1):38–45, Mar 2001.

[RF97] M. Renaud and J. Y. Fourquet. Minimum time motion of a mobile robot with two independent, acceleration-driven wheels. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2608–2613 vol.3, Apr 1997.

[RGYU99] K. Reif, S. Gunther, E. Yaz, and R. Unbehauen. Stochastic stability of the discrete-time extended kalman filter. *Automatic Control, IEEE Transactions on*, 44(4):714–728, Apr 1999.

[RP94] David B. Reister and François G. Pin. Time-optimal trajectories for mobile robots with two independently driven wheels. *The International Journal of Robotics Research*, 13(1):38–54, 1994.

[RSS11] Illah Reza Nourbakhsh Roland Siegwart and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots.* The MIT Press, 2011.

[SASL13] J. Saarinen, H. Andreasson, T. Stoyanov, and A.J. Lilienthal. Normal distributions transform monte-carlo localization (ndt-mcl). In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 382–389, Nov 2013.

[SB02] C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 508–513 vol.1, 2002.

[SGF10] Zvi Shiller, Oren Gal, and Thierry Fraichard. The Nonlinear Velocity Obstacle Revisited: the Optimal Time Horizon. In *Guaranteeing Safe Navigation in Dynamic Environments Workshop*, Anchorage, United States, May 2010.

[SLS01] Z. Shiller, F. Large, and S. Sekhavat. Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 3716–3721 vol.4, 2001.

[SP07] M. Seder and I. Petrovic. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1986–1991, April 2007.

[SSSS13] Zvi Shiller, Sanjeev Sharma, Ishai Stern, and Asher Stern. Online obstacle avoidance at high speeds. *The International Journal of Robotics Research*, 32(9-10):1030–1047, 2013.

[TBF05] Sebastian Thrun, Worfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[ULVM13] P. Urcola, M. T. Lorente, J. L. Villarroel, and L Montano. Seamless indoor-outdoor robust localization for robots. In *Proceedings of ROBOT 2013: First Iberian Robotics Conference*, November 2013.

[ULVM14] P. Urcola, M. T. Lorente, J. L. Villarroel, and L Montano. Seamless robot localization and navigation in indoors-outoors for logistics in warehouses. In *2014 European Robotics Forum (ERF)*, pages 1–8, November 2014.

[ULVM17] Pablo Urcola, María-Teresa Lorente, José L. Villarroel, and Luis Montano. Robust navigation and seamless localization for carlike robots in indoor-outdoor environments. *Journal of Field Robotics*, 2017.

[vdBGLM11] Jur van den Berg, Stephen Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. volume 70, pages 3–19, 04 2011.

[VdBO08] Jur Van den Berg and Mark Overmars. Planning time-minimal safe paths amidst unpredictably moving obstacles. *The International Journal of Robotics Research*, 27(11-12):1274–1294, 2008.

[WvdBM09] D. Wilkie, J. van den Berg, and D. Manocha. Generalized velocity obstacles. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5573–5578, Oct 2009.