CARLOS III UNIVERSITY OF MADRID

DEPARTMENT OF SYSTEMS ENGINEERING AND AUTOMATION

MASTER'S THESIS

ADVANCED APPLICATIONS OF THE FAST MARCHING SQUARE PLANNING METHOD

Author: Javier V. Gómez González

Advisor: Dr. Santiago Garrido Bullón

OFFICIAL MASTER'S PROGRAM IN ROBOTICS AND AUTOMATION

LEGANÉS, MADRID

NOVEMBER 2012

CARLOS III UNIVERSITY OF MADRID
OFFICIAL MASTER'S PROGRAM IN ROBOTICS AND AUTOMATION

The committee approves the Master's thesis titled "**Advanced Applications of the Fast Marching Square Planning Method**" done by **Javier V. Gómez González**.

Date: November 2012

Committee:

Dr. Luis Moreno Lorente

Dr. Concepción A. Monje Micharet

Dr. Beatriz López Boada

To those who trusted me without knowing me and to those who know me and still do it.

To PR2, Justin, Stanley, Junior, PetMan, Asimo, Nao, Qbo and many others... because your performance keeps me awake.

Contents

Tables Index

List of Figures

Acknowledgments

I want to acknowledge all those people who helped me during my research the last year and a half.

First of all, thanks to Diego Rodríguez-Losada, Alberto Valero, Miguel Hernando and Pablo San Segundo, I am here thanks to you.

Of course, thanks to Santiago Garrido, Luis Moreno and Dolores Blanco for giving me this great opportunity.

Thanks also to my laboratory mates: David Álvarez, Alejandro Lumbier, Fransico Rascón, Antonio Flores, Dorin Copaci, César Arismendi and many others. My work would never have been that good without your very useful help.

Also, thanks also to Alberto Vale, the IPFN team and Pedro Lima. It has been such a pleasure to work with you and I hope to continue during many many years.

Moreover, thanks to Frode Eika Sandnes and Nikolaos Mavridis. The experiences I shared with you are enough motivation to continue working.

Last, but not least, thanks to my family and friends. Now you know why I was so missing the last months.

This work was supported by the project number DPI2010-17772 and HYPER project funded by CONSOLIDER-INGENIO 2010, both from the Spanish Ministry for Science and Innovation.

Resumen

Con frecuencia algunos investigadores consideran el problema de la planificación de trayectorias resuelto. Estas peligrosas afirmaciones no pueden estar más lejos de la realidad. Aunque este campo ha sido investigado desde hace más de 30 años, aún quedan multitud de problemas abiertos. Los investigadores deben centrarse en ellos con el objetivo de acercar más los robots a aplicaciones reales. Para ello los algoritmos deben ser robustos y fiables ante situaciones cuyo modelado es realmente complejo.

Esta Tesis de Máster recopila algunos de los problemas de gran interés para la comunidad en la actualidad. Por ejemplo, se estudian los sistemas multirrobot, se aborda el aprendizaje por demostración y se analiza el complejo caso del proyecto ITER (International Thermonuclear Experimental Reactor) y sus operaciones de manipulación remota. En todos estos casos se usa el algoritmo de planificación Fast Marching Square que, como se detalla en las siguientes páginas, proporciona una robustez y calidad de trayectorias que pocos algoritmos son capaces de igualar.

Palabras clave: Planificación de trayectorias, Fast Marching, Fast Marching Square, Aprendizaje, Planificación de Formaciones de Robots, Proyecto ITER.

Abstract

It is common that many researchers consider the path planning problem solved. These dangerous sentences are far away from reality. Although this field has been explored during more than 30 years, there are still many unsolved problems. Researchers should focus on these with the objective of bringing robots closer to real applications. For this, the algorithms have to be robust and reliable in situations whose modeling is quite complex.

This Master's Thesis collects many of the problems of high interest for the community. For instance, multi-robot systems are studied, programming by demonstration is explored and the complex case of the remote handling operations of the ITER project (International Thermonuclear Experimental Reactor) are analyzed. In all these, the Fast Marching Square planning method is used. This, as detailed in this document, provides a path quality and robustness that only a few algorithms are able to reach.

Keywords: Path planning, Fast Marching, Fast Marching Square, Learning, Robot Formation Planning, ITER Project.

Chapter 1

Introduction

The objective of this work is to detail novel approaches to complex problems related with path planning, creating a general path planning framework.

The path planning problem is considered solved by some researchers. They argue that the current existing solutions are good enough to solve most of the problems that require robot motion. However, it is still one of the most active fields in robotics research. In fact, in the last IROS conference (October, 2012) path and motion planning was the topic which occupied more sessions. Therefore, many researchers are still focusing their efforts on the path planning problem, trying to come up with better, faster and more general planning frameworks or modelling the real world in a more complex way.

Hence, it turns out that the path planning problem is not as solved as many people think. It is true that there exist very good solutions but none of them can be considered as a *general* framework. In fact some approaches only perform well in specific cases or are notable to adapt to new models of robots or environments.

1.1 Context and motivation

As metioned before, we consider that a good path planning approach has to perform well in many different cases, because it is not optimal to have different path planning algorithms for every different situation. This would require to distinguish those different situations and implement many different algorithms, which is complex and very time consuming.

In this context, we focus this thesis in solving many different problems with the same path planning algorithm: Fast Marching Square (FM^2). The characteristics of this planning method allow it to be adapted to several problems with a very different point of view. Here, the high level problems are translated into path planning ones instead using the path planning as a tool or intermediary step.

Facing the problem this way, we show in the next chapters that the FM^2 is a very good candidate to be considered as a *general* path planning framework.

1.2 Document structure

In the next chapter, a detailed state of the art of the different problems tackled in this thesis is given. In chapter 3 we describe in detail the FM^2 algorithm, which is going to be our base for the next chapters. Chapter 4 focuses on the robot formation path planning problem and how we faced it with FM^2. Following, chapter 5 creates a variation of the FM^2 which allow to reduce the time computation of the paths, focusing on the application of FM^2 for high dimensional spaces. Then, chapter 6 shows how motion learning can be implemented within the FM^2 framework.

In chapters 7 and 8 the FM^2 is applied to a demanding problem: the International Thermonuclear Experimental Reactor and its Remote Handling operations.

Lastly, the conclusions of the thesis are included in chapter 9

Chapter 2

The State of the Art in Robot Path Planning

Path planning has been a very active field from the beginning of the artificial intelligence. Although currently there are very good approaches, there is not any algorithm able to satisfy all the requirements that a path planning algorithm needs to satisfy: low computational complexity, reliability, completeness, robustness, smooth paths, optimal solutions, safety, etc.

The algorithms which are very fast usually provide non-smooth paths, so a smoothing step is required aftewards. On the other hand, the algorithms whose solutions are smooth are usually fast in 2 dimensions (or even three) but their usefulness decrease as the dimensionality is increased.

2.1 Classification of the path planning algorithms

There is a huge literature in path planning. However, most of the path planning algorithms can be classified as follows:

- *Geometric methods*: The environment is described as a set of polygons and, from their properties, the paths are computed as a sequence of primitives

such as lines, arcs, splines, etc. The most common approach of this class are those based on *visibility graphs* (Asano, Asano, Guibas, Hershberger, & Imai, 1986; Ghosh & Mount, 1991). The visibility graph of a set of nonintersecting polygonal obstacles in the plane is an undirected graph whose vertices are the vertices of the obstacles and whose edges are pairs of vertices such that the open line segment between each two vertices does not intersect any of the obstacles. Although this approach is very intuitive, it is not possible to extend it to more than two dimensions. Also, to determine the visible portions of the map is a very complex problem.

Other common geometric approach is to compute the path using the Delaunay triangulation of the environment (Delaunay, 1934). This is one of the most widely-employed triangulation algorithms because it minimizes the sum of all the angles of the triangles in the triangulation. Although its complexity is $O(n\ log(n))$ in 3D spaces for n points in smooth surfaces (Attali, Boissonnat, & Lieutier, 2003), its main drawback is that the triangulation is not unique and it can lead to very weird paths, as we detail in chapter 8.

- *Graph-based methods*: This is the class in which most of the algorithms are. In this class, the environment is modeled by the state of the robot with respect itself and also with the environment. A graph is built in which every node is one state of the robot and its environment. The transitions between states are modeled as costs and the path chosen is the one that minimizes the total cost of reaching a goal state from a current state.

  There exist several subclasses within the graph search method, depending on how the graph is constructed and also how the costs are assignated. One of the possible classifications is:

  - Grid-based methods: characterized by discretization of the space in grid cells. The most common grid representations are rectangular or

triangular. This discretization can lead to a loss of accuracy, but this issue is overcome by chosing an appropiate cell size. Every cell of the space is a node of a graph, and it is connected with its neighbours (4 or 8-connectivity in 2D, depending on the algorithm) and the cost of travelling from one node to other can be set on many different ways. Once the costs are set, graph search algorithms can be applied in order to choose the path which allows to reach the goal point with the minimum possible cost.

Within this group, we can find the typical graph search algorithms, such Dijkstra (Dijkstra, 1959) or A* (Hart, Nilsson, & Raphael, 1968). Artificial potential fields can be included in this group. Although their mathematical model is continuous (Barranquand, Langlois, & Latombe, 1992) they use to represent the space by means of grid cells. In these algorithms, the robot is treated as an electric charge moving under a potential field in which the obstacles have the same charge, so they repeal the robot from them. The goal point has the opposite charge, attracting the robot towards it. The main drawback of this approach is that it is prone to local minima and oscillations.

Finally, the Fast Marching Method (Sethian, 1999) in which we focus in this thesis (see chapter 3), lies also in this group. In this case, the cost for each node is related with the time a propagating wave takes to reach that node.

– Combinatorial methods: LaValle (in a recent classification (LaValle, 2011)) defines this group as those set of algorithms which constructs structures which capture all the information needed in path planning (Berg, Kreveld, Overmars, & Schwarzkopf, 2008),(LaValle, 2006). The most widespread methods within this group are those based on road maps, which mainly consist in obtaining precalculated short paths from the map (road map) and create the path by taking the needed

sections of the road map.

– Sampling-based methods: this type of algorithms incrementally searches in the space for a solution using a collision detection algorithm (LaValle, 2006). The most extended algorithms of this group are those based on rapidly exploring random trees (RRTs) (Kuffner & LaValle, 2000). The branches of these trees are randomly created from the initial point of the trajectory. RRT is a very fast planning method and one of the most widely used nowadays.

  The main problem of these two last goups is their stochasticity. Most of the times the computed paths are far from the optimal one, are not safe or not smooth. However, there are many different approaches which modify these methods in order to make them more useful.

Figure 2.1 summarizes this classification. It is important to remark that this classification is not unique. Moreover, the classification aforementioned is not a common one. However, it has been built from an implementation point of view. For instance, potential fields or Fast Marching-based methods are not usually included in graph-based methods. Their mathematical models are different but they rely on the same (or very similar) environment representation.

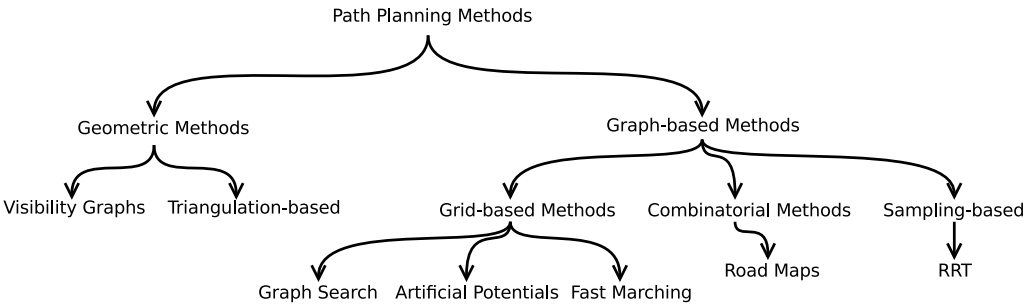


Figure 2.1: *Classification of the current path planning approaches.*

In the next sections the state of the art of path planning research is summarized.

2.2 Latest trends in path planning research

Many researchers consider that the path planning problem as a solved one. However, most (if not all) current approaches perform poorly when they are applied to more complex path planning problems. Current research efforts focus on more realistic environment representations and reformulating the path planning problem in order to get it closer to the real problem robots have to face when working among humans.

Concretely, during the recent years researchers have focused on path planning in dynamic environments with high uncertainty, due sensor noise or also due a movement of other agents in the surroundings of the robot or crowdy places. Also, navigation in large environments, in which all the obstacles are not modelled.

For instance, in dynamic, uncertain environments (DUEs), robots must work in close proximity with many other moving agents, whose future actions and reactions are difficult to predict accurately. Robot motion planning in dynamic environments has recently received substantial attention because of the Defense Advanced Research Project Agency (DARPA) Urban Challenge (DARPA, 2011) and growing interest in service and assistive robots (see, e.g., (Tadokoro, Hayashi, Manabe, Nakami, & Takamori, 1995) and (Roy, Gordon, & Thrun, 2003)). In urban environments, traffic rules define the expected behaviors of the dynamic agents and constrain expected future locations of moving objects. In other applications, agent behaviors are less well defined, and the prediction of their future trajectories is more uncertain.

When the future locations of moving agents are known, the two common approaches are to add a time dimension to the configuration space, or to separate the spatial and temporal planning problems (LaValle, 2006). When the future locations are unknown, the planning problem is solved locally (via reactive planners) (Fiorini & Shiller, 1998), (Choset et al., 2007) or in conjunction with a global planner that guides the robot toward a goal (Xu, Stilwell, & Kurdila, 2010). The

work (Du Toit & Burdick, 2012) presents an initial approach to a general framework that integrates planning, prediction an estimation, incorporating as well the effect of anticipated future measurements in the motion planning process.

Another important problem which still remains open are those systems with high dimensional configuration spaces. These approaches are usually based on RRT, with smoothing steps (Ettlin & Bleuler, 2006), (Karaman & Frazzoli, 2011). Most of these approaches are just modifications of the RRT basic algorithm to apply it to specific cases: Transition-based RRT (T-RRT) (Jaillet, Cortés, & Siméon, 2010) which includes a state transitions cost, Manhattan-like RRT (ML-RRT) (Cortes, Jaillet, & Siméon, 2008) which incorporates a mix of active and passive parameters, or MLT-RRT (Iehl, Cortés, & Siméon, 2012) which is a combination of the aforementioned. Other approaches simply focus on dealing with much higher dimensionality, but under very specific contexts (Shkolnik & Tedrake, 2009).

Finally, other interesting problems have to be faced, such path planning for multisection continuum arms (Godage, Branson, Guglielmino, & Caldwell, 2012) or maximum planning coverage minimizing energy (Michel & McIsaac, 2012).

In spite of the huge literature, it turns out that most of the path planning efforts over the last years focused on solving very specific tasks or creating new robot configurations and solving them as well. We, as path planning researchers, consider that there is a lack of a general planning framework, being this the main objective of this thesis and our work. The Fast Marching Square algorithm described in chapter 3 is a good approach to a general framework, because it can be applied easily to several complex problems wihtout including deep modifications.

In the following sections we include a brief state of the art of the different problems that are faced in this thesis.

2.2.1 Multi-robot systems: state of the art

The case of the multi-robot systems is very similar to the path planning: it has been investigated during many years and researchers are moving to more complex problems. Motion planning for multi-robot systems is considered well solved, so the research efforts are being focused in aerial formations (A. Yang, Naeem, Irwin, & Li, 2012), (Turpin, Michael, & Kumar, 2012), autonomous underwater vehicles (Zhou, Jian, Wen-Xia, & Jin-Ping, 2012), formations with changeable shape (Sanhoury, Amin, & Husain, 2012) or for specific robots or navigation conditions (Sadowska, Kostic, Wouw, Huijberts, & Nijmeijer, 2012), (Tian & Sarkar, 2012).

Among all the possible multi-robot systems (coordination and cooperation, multi-sensor fusion, task allocation, etc (Cai & Yang, 2012)) we focus on the formation control problem, which can be considered as a coordination scheme for the positions and orientations of the member of the formation.

So far, different approaches have been proposed to solve the robot formation control problem. Beard et al (Beard, Lawton, & Hadaegh, 2001) classify the different approaches in three main groups: *leader-followers*, where one robot is the leader and the rest are followers. The leader motion can be determined by a calculated trajectory or by teleoperation; and the followers' motion is determined by tracking the leader with some geometrical restrictions. This motion can change dynamically over time, if necessary (Tanner, 2004). These leaders can be real robots or, as proposed in (Leonard & Fiorelli, 2001), virtual leaders. The second proposed group is *behavioral*, where several behaviors are weighted in order to give a motion plan to each vehicle (Balch & Arkin, 1998). The third group is *virtual structure*, where the entire formation is treated as a single structure, and its desired motion is translated into the desired motion of each vehicle (Egerstedt & Hu, 2001).

Many other works have been carried out, such as using virtual potential

fields to influence the location of each robot during movement in simple formations (Garrido, Moreno, & Lima, 2011) or in very populated groups (Leonard & Fiorelli, 2001). Other techniques are based on those virtual potentials, such as the inclusion of springs and dampers (Urcola & Montano, 2009), (MacArthur & Crane, 2007) to create virtual forces that are transformed into velocity commands.

Another criterion for classification is the rigidity of the formation geometry. Two large groups can be distinguished: *rigid formations*, where the geometry is fully specified and the motion control of each robot ensures that this geometry is accurately achieved (Das et al., 2002). These approaches require a method to switch between geometries when the environment demands it (Fierro, Song, Das, & Kumar, 2002). In *dynamic formations*, the geometric structure is can be distorted in the presence of obstacles and environmental conditions (Ogren, Fiorelli, & Leonard, 2003).

The chapter 4 is an important improvement over the work done in (Garrido et al., 2011), reducing the computational time of the approach and also dealing with uncertainty in the robots and obstacles positions.

2.2.2 Motion learning algorithms: state of the art

Many different approaches have been proposed to implement the robot learning. One approach has been to use the concept of motion primitives (Schaal, Peters, Nakanishi, & Ijspeert, 2003). The Dynamic Movement Primitives (DMP) are a set of nonlinear differential equations which creates smooth control policies. These primitives are learned by means of imitation learning and reinforcement learning. A more recent approach based on the motion primitives idea is proposed in (Lee & Ott, 2010), where the primitives learning is carried out using incremental kinesthetic teaching by means of Hidden Markov Models (HMM).

Other approach is to show the robot how to perform a discrete motion (i.e. point-to-point trajectories) (Khansari-Zadeh & Billard, 2011), where the robot

performs a movement keeping the motion as similar as possible to the demonstrations. Calinon goes a step further and it is proposed a control strategy for a robotic manipulator operating in unstructured environments while interacting with human operators (Calinon, Sardellitti, & Caldwell, 2010). Such situations are starting to be common in manufacturing.

Lastly, a different, very interesting approach is given in (Abbeel & Ng, 2004). In this case apprenticeship learning is carried out by supposing a Markov decision process where the reward function is not explicitly given. By observing an expert developing a task, this algorithm guesses that the expert is maximizing an unknown reward function, expressible as a linear combination of known features.

All the mentioned methods are a few examples of the wide literature about robot learning. All of them have proved a good performance within their objectives, but their underlying mathematical model is usually based on probabilistic terms, causing the learning to be stochastic and even unstable under certain conditions. Besides, these approaches are not able to take into account environment conditions, since their are based on modifying motion control parameters.

However, in (Berenson, Abbeel, & Goldberg, 2012) a different approach is proposed. Here, two modules run in parallel: one planning from scratch and other retrieving and repairing paths stored in a path library with the aim of reducing computation time when planning in high-dimensional spaces. In (Melchior & Simmons, 2012) another approach to learning motion trajectories for robotic manipulator tasks is introduced. A graph is constructed to determine correspondences between multiple imperfect demonstrations. Then the robot learner plans novel trajectories that safely and smoothly generalize the teacher's behavior, while attenuating those imperfections. These approaches are close in concept to the ones proposed in chapters 5 and 6, but the underlying mathematical model and solution are very different.

2.3 Fast Marching Methods in path planning: previous work

The application of the Fast Marching Method in path planning is not novel but recent. At the beginning, the Fast Marching Method was used to find paths within the Voronoi diagram (Garrido, Moreno, Abderrahim, & Martin, 2006). Later, Fast Marching was combined with the Extended Voronoi Transform (EVT) in what they called Voronoi Fast Marching (VFM) (Garrido, Moreno, Blanco, & Muñoz, 2007). This approach has been applied to exploration of cluttered environments (Garrido, Moreno, & Blanco, 2008) and to robot formations (Garrido et al., 2011). An improvement of the VFM was the Fast Marching Square (FM^2) planning method (Garrido, Moreno, Abderrahim, & Blanco, 2009).

The VFM and FM^2 have been applied to other planning problems such as: smooth planning for non-holonomoic robots (Garrido, Moreno, Blanco, & Martin, 2009), simultaneous robot localization and mapping (Garrido, Moreno, & Blanco, 2009), planning in outdoor environments (Malfaz, Garrido, & Blanco, 2012), (Sanctis, 2010), RRT path smoothing (Jurewicz, 2010), or tube skeletons (Garrido, 2008) among others.

Chapter 3

Fast Marching Square Planning Method

Among all the different path planning algorithms, this document focuses on the Fast Marching Square method, FM^2 (Garrido, Moreno, Abderrahim, & Blanco, 2009).

This method is based on creating artificial potential fields from the sampled information through sensors and obtaining the path from these fields. The Fast Marching Method can be applied to create the potential fields and to obtain artificial *local minima free fields*, thereby solving one of the most important drawbacks of these path planning methods.

Although FM^2 is based on the creation of artificial potential fields, we consider that this method should not be labeled as a potential-based method. These algorithms have a mathematical formulation very different to FM^2: they mainly consist on putting together two different potentials: an attractive one and a repulsive one. The robot then is considered as a particle moving under the influence of these potentials. However, this potentials mixture usually lead to local minima.

On the other hand, FM^2 includes the concept of *time* in the potential creation

process. The robot in this case is considered as a particle moving under the influence of time, choosing those paths which can be undertaken in the shortest possible time.

Along this chapter the FM^2 method is detailed. Next section first explains the Fast Marching Method since it is the base for FM^2. Then, in section 3.2 the FM^2 is presented.

3.1 Fast Marching Method (FMM)

The Fast Marching Method (FMM) is a particular case of Level Set Methods, initially developed by Osher and Sethian (Osher & Sethian, 1988). It is an efficient computational numerical algorithm for tracking and modeling the motion of a physical wave interface (front), denoted Γ. This method has been applied to different research fields including computer graphics, medical imaging,computational fluid dynamics, image processing, computation of trajectories, etc.(Jbabdi et al., 2008; Li, Xue, Cui, & Wong, 2011; K. Yang, Li, Liu, & Jiang, 2010).

3.1.1 Intuitive introduction to the Fast Marching Method

The FMM can be understood intuitively considering the expansion of a wave. If a stone is thrown into a pond, a wavefront is originated, and this wave expands with a circle shape around the point where the stone fell. In this example the fluid is always water, thus the wave expansion velocity is always the same, and that is why the wavefront is circular. Instead, if we repeat this experiment mixing water and oil, we would observe that the wave expands at different speeds in each medium. As a consequence, the wavefront will not be circular any more. If we consider another point on the fluid (a target point), the wavefront will arrive to that point after a certain time. The path that the wavefront has followed from the origin to the target point will be the shortest path (in terms of time),

considering that the traveling speed along the path is the expansion velocity of the wavefront (which differs depending on the fluid).

In the FMM, the wavefront is called the interface. The interface can be a flat curve in 2D, or a surface in 3D (but the mathematical model can be generalized to n dimensions). The FMM calculates the time T that a wave needs to reach every point of the space. The wave can be originated from more than one point, each source point originates one wave. Source points have an associated time $T = 0$.

In the context of the FMM we assume that the front Γ evolves by motion in the normal direction. The speed, denoted F, does not have to be the same everywhere, but it is always non-negative. At a given point, the motion of the front is described by the equation known as the Eikonal equation (as given by Osher and Sethian (Osher & Sethian, 1988)):

$$1 = F(x)|\nabla T(x)|$$

where x is the position, $F(x)$ the expansion speed of the wave at that position, and $T(x)$ the time that the wave interface requires to reach x.

The magnitude of the gradient of the arrival function $T(x)$ is inversely proportional to the velocity:

$$\frac{1}{F(x)} = |\nabla T(x)|$$

The $T(x)$ function originated by a wave that grows from one single point presents only a global minima at the source and no local minima. As $F(x) \geq 0 \, \forall x$ the wave only grows (expansion), and hence, points farther from the source have greater T. A local minima would imply that a point has a T value lesser than a neighbour point which is nearer to the source. This is impossible as this neighbour must have been reached by the wave sooner.

In the next subsection the full mathematical model of the FMM is given.

3.1.2 Mathematical formulation of the Fast Marching Method

The starting point is the Eikonal equation:

$$\frac{1}{F(x)} = |\nabla T(x)|$$

Because the front can only expand ($F(x) \geq 0 \forall x$), the arrival time $T(x)$ is single valued. Sethian proposed a discrete solution for the Eikonal equation (Sethian, 1996). In 2D the area is discretized using a grid map. We denote i, j the row i and column j of the grid map, that corresponds to a point $p(x_i, y_j)$ in the real world. The discretization of the gradient $\nabla T(x)$ according to (Osher & Sethian, 1988)) drives to the following equation:

$$max(D_{ij}^{-x}T, 0)^2 + min(D_{ij}^{+x}T, 0)^2 + max(D_{ij}^{-y}T, 0)^2 + min(D_{ij}^{+y}T, 0)^2 = \frac{1}{F_{ij}^2} \quad (3.1)$$

or to the one proposed by Sethian (Sethian, 1996), simpler but less accurate:

$$max(D_{ij}^{-x}T, -D_{ij}^{+x}, 0)^2 + max(D_{ij}^{-y}T, -D_{ij}^{+y}, 0)^2 = \frac{1}{F_{ij}^2} \quad (3.2)$$

where:

$$\begin{aligned} D_{ij}^{-x} &= \tfrac{T_{i,j} - T_{i-1,j}}{\triangle x} \\ D_{ij}^{+x} &= \tfrac{T_{i+1,j} - T_{i,j}}{\triangle x} \\ D_{ij}^{-y} &= \tfrac{T_{i,j} - T_{i,j-1}}{\triangle y} \\ D_{ij}^{+1} &= \tfrac{T_{i,j+1} - T_{i,j}}{\triangle y} \end{aligned} \quad (3.3)$$

and $\triangle x$ and $\triangle y$ are the grid spacing in the x and y directions. Substituting Eq. 3.3 in Eq. 3.2 and letting

$$\begin{aligned} T &= T_{i,j} \\ T_1 &= min(T_{i-1,j}, T_{i+1,j}) \\ T_2 &= min(T_{i,j-1}, T_{i,j+1}) \end{aligned} \quad (3.4)$$

we can rewrite the Eikonal Equation, for a discrete 2D space as:

$$max\left(\frac{T-T_1}{\triangle x},0\right)^2 + max\left(\frac{T-T_2}{\triangle y},0\right)^2 = \frac{1}{F_{i,j}^2} \tag{3.5}$$

As we are assuming that the speed of the front is positive ($F > 0$) T must be greater than T_1 and T_2 whenever the front wave has not already passed over the coordinates i, j. Consequently Eq. 3.5 can be solved as the following quadratic:

$$\left(\frac{T-T_1}{\triangle x}\right)^2 + \left(\frac{T-T_2}{\triangle y}\right)^2 = \frac{1}{F_{i,j}^2} \tag{3.6}$$

Whenever $T > T_1$ and $T > T_2$ (we always take the greater value of T when solving Eq. 3.6). If $T < T_1$ or $T < T_2$, from Eq. 3.5 the corresponding member is 0 ($max\left(\frac{T-T_1}{\triangle x},0\right)$), and hence Eq. 3.5 is reduced to:

$$\left(\frac{T-T_1}{\triangle x}\right) = \frac{1}{F_{i,j}} \tag{3.7}$$

if T resulted to be smaller than T_1 when solving 3.6, or

$$\left(\frac{T-T_2}{\triangle y}\right) = \frac{1}{F_{i,j}} \tag{3.8}$$

if T resulted to be smaller than T_2 when solving 3.6.

3.1.3 Implementation details of the Fast Marching Method

Eq. 3.5 can be solved iteratively over a grid map. For doing so, the cells of the grid map must be labeled of one of the following types:

- *Unknown*: Cells whose T value is not known yet (the wave front has not reached the cell).
- *Narrow Band*: Candidate cells to be part of the front wave in the next iteration. They are assigned a T value that can still change in future iterations of the algorithm.
- *Frozen*: Cells that have already been passed over by the wave and hence their T value is fixed.

The algorithm has three stages: initialization, main loop, and finalization.

Initialization The algorithm starts by setting $T = 0$ in the cell or cells that originate the wave. These cells are labeled as *frozen*. Afterwards it labels all their Manhattan neighbors as *narrow band*, computing T (Eq. 3.5) for each of them.

Main loop In each iteration the algorithm will solve the Eikonal Equation (Eq. 3.5) for the Manhattan neighbors (that are not yet frozen) of the narrow band cell with the lesser T value. This cell is then labeled as *frozen*. The narrow band maintains an ordered list of its cells. Cells are ordered by increasing T value (first cells have lesser T values).

Finalization When all the cells are *frozen* (the *narrow band* is empty) the algorithm finishes.

We can see the process in Figures 3.1 and 3.2. In Figure 3.1 the wave is originated from one point. In 3.2 there are two wave sources. Black points are *frozen*, and their T value will not change. Gray points are the *narrow band*, while white ones are *unknown*. As it can be appreciated the waves grow concentric to the source. In Figure 3.2 they join, and the waves develop themselves growing together. The iterative process expands cells in the same order that the physical wave grows, as cells with less T are expanded first, that is, if two cells have a different arrival time, the cell that is reached before by the front wave is expanded first.

If we consider T as the third dimension over the z axis, the result of completing the wave expansion of Figures 3.1 and 3.2 results in Figure 3.3a and 3.3b respectively. As it was supposed to happen, as $F > 0$, when we get far from the sources the time T required to reach the point is greater (higher on the z axis). It can be appreciated that with one single source, there is only one minima at the source. With more than one source we have a minima at each source with $T = 0$.

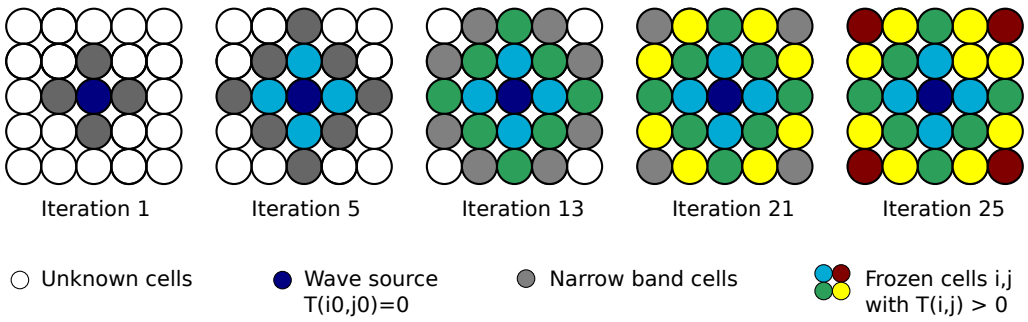


Figure 3.1: *Iterations of the FMM in a 5x5 grid map with one wave source.*

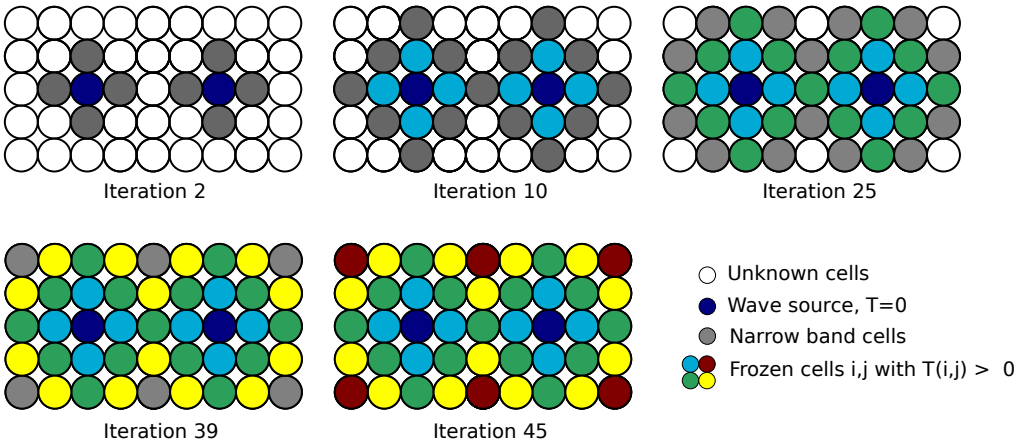


Figure 3.2: *Iterations of the FMM in a 5x9 grid map with two wave sources.*

Finally, the pseudo-code of the FMM is shown in the Algorithm 1.

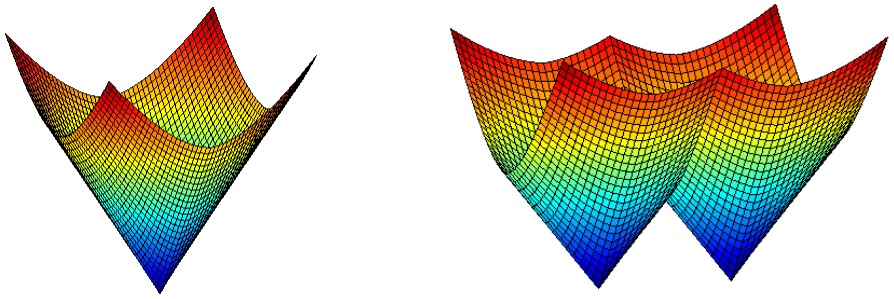
3.1.4 Application of Fast Marching Method to path planning

Let us consider a binary grid map, in which obstacles are valued as 0, and free space as 1. These values can be taken as the wave expansion speed F over the grid map. At obstacles, wave expansion speed is 0, as the wave cannot go through obstacles, and on free space, wave expansion speed is constant and equals to 1. If we want to compute the path between two points p_0 and p_1 we could expand a wave from p_1 until it reaches p_0. Due to the wave expansion properties, the path that has followed the wave interface from the target to the

input : A grid map G of size $m \times n$
input : The set of cells Ori where the wave is originated
output: The grid map G with the T value set for all cells

Initialization;
foreach $g_{ij} \in Ori$ **do**
  $g_{ij}.T \leftarrow 0$;
  $g_{ij}.state \leftarrow$ FROZEN;
  foreach $g_{kl} \in g_{ij}.neighbours$ **do**
    if $g_{kl} =$ *FROZEN* **then** skip; **else**
      $g_{kl}.T \leftarrow solveEikonal(g_{kl})$;
      if $g_{kl}.state =$ *NARROW BAND* **then**
      $narrow\_band.update\_position(g\_kl)$;
      if $g_{kl}.state =$ *UNKNOWN* **then**
        $g_{kl}.state \leftarrow$ NARROW BAND;
        $narrow\_band.insert\_in\_position(g_{kl})$;
      end
    end
  end
  Iterations;
  while $narrow\_band$ *NOT EMPTY* **do**
    $g_{ij} \leftarrow narrow\_band.pop\_first()$;
    foreach $g_{kl} \in g_{ij}.neighbours$ **do**
      if $g_{kl} =$ *FROZEN* **then** skip; **else**
        $g_{kl}.T \leftarrow solveEikonal(g_{kl})$;
        if $g_{kl}.state =$ *NARROW BAND* **then**
        $narrow\_band.update\_position(g\_kl)$;
        if $g_{kl}.state =$ *UNKNOWN* **then**
          $g_{kl}.state \leftarrow$ NARROW BAND;
          $narrow\_band.insert\_in\_position(g_{kl})$;
        end
      end
    end
  end
end

Algorithm 1: FMM algorithm



(a) *One wave source* **(b)** *Two wave sources*

Figure 3.3: *FMM applied over a grid map. The arrival time is shown in the z axis.*

source point will be always the shortest trajectory in time. As the wave expansion speed is constant, this trajectory is also the shortest solution in distance. The wave is originated from the target point, hence, the computed $T(x)$ field will have only one minima at the target point. Hence, following the maximum gradient direction from the initial point we will reach the target point, obtaining the trajectory. This solution is unique and complete.

Figure 3.4 shows an example. We want to trace the shortest path from a start point to a goal point. First, we have the binary map (obtained through a 3D laser range sensor). For security reasons, the obstacles of this map (labeled as 0, black points) are dilated by the maximum radius of the robot. Then, the FMM is applied using the goal point as a wave source. Once the interface Γ has reached the start point the algorithm stops expanding.

The resulting grid map stores at any pixel the time T required by the front wave to reach that pixel. The isocurves join together all the points that have been passed through at the same instant of time. These curves are the trace of the front wave. If we compute the maximum gradient direction at any point of the grid map we obtain the normal direction to the isocurve, that is, the direction

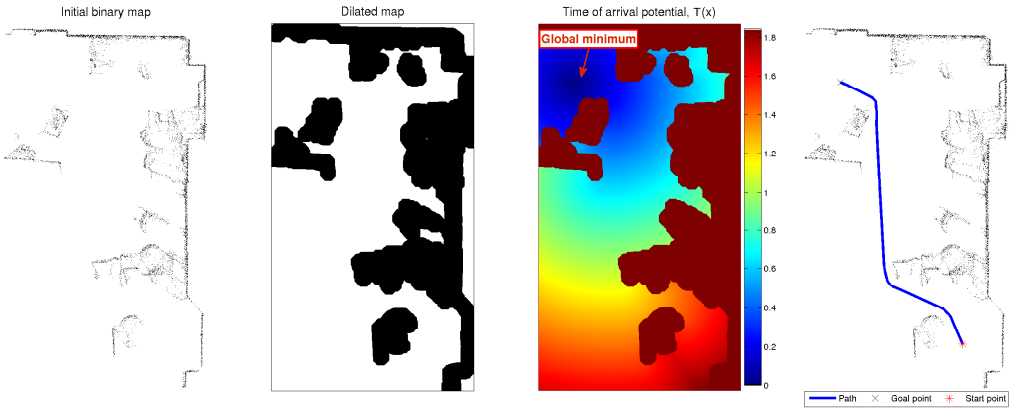


Figure 3.4: *Steps of the FMM applied to path planning.*

the curve has followed when expanding. The maximum gradient direction is computed applying the Sobel operator over the grid map.

$$grad_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \star L \qquad grad_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \star L \tag{3.9}$$

For tracing the path between the initial and the goal points we just need to follow the maximum gradient direction starting at the initial point. The path is computed iteratively. $grad_{ix}$ and $grad_{iy}$ are computed at every point p_i. From p_i is computed p_{i+1} (equation 3.10) and successively until arriving to the goal point. As this goal point is located at the global minima it is always reached (whenever there is path).

$$\begin{aligned} mod_i &= \sqrt{grad_{ix}^2 + grad_{iy}^2} \\ alpha_i &= \arctan(\tfrac{grad_{iy}}{grad_{ix}}) \\ p_{(i+1)x} &= p_{ix} + mod_i \cdot \cos(alpha_i) \\ p_{(i+1)y} &= p_{iy} + mod_i \cdot \sin(alpha_i) \end{aligned} \tag{3.10}$$

3.2 Fast Marching Square planning method (FM^2)

The trajectories generated in the original work by Sethian (Sethian, 1996) (see Figure3.4) on the FMM are optimal according to the minimal Euclidean distance criterion, but it creates paths which run too close to obstacles and are not smooth. These facts turn FMM into an unreliable path planner for most robotic applications. However, the FM^2 algorithm solves these problems by obtaining a velocities map which modifies the wave expansion according to the distance of the closest obstacle.

Let us take an evidence grid map in which obstacles are labeled as 0 and free space as 1. We can apply the FMM to this map being all the obstacles a wave source. In the previous section, there was just one wave source, at the target point. Here all the obstacles are a source of the wave, and hence, several waves are being expanded at the same time. The map resulting of applying this wave expansion to the binary map depicted in Figure 3.4 can be seen in the first subfigure of Figure 3.5. It represents a potential field of the original map. As pixels get far from the obstacles, the computed T value is greater. This map can be seen as a *velocities map*. If we consider the T value as a measure proportional to the maximum allowed speed of the robot at each point, we can appreciate that speeds are lower when the pixel is close to the obstacles, and greater far from them. In fact, a robot whose speed at each point is given by the T value will never collide, as $T \rightarrow 0$ when approaching the obstacles. Making an appropriate, relative scaling (between 0 and 1) of the values of this potential to the robot allowed speeds, we have then the velocities map, that provides a safe speed for the robot at any point of the environment. In Figure 3.6 we can appreciate the speeds profile. In the image is clear that speeds become greater far from the obstacles.

We could calculate now the path as we did in Section 3.1.4 but instead of taking a constant value for the expansion speed F, we use the speed given by the velocities map. Now, if we expand a wave from one point of the grid map,

considering that the expansion speed $F(x, y) = T(x, y)$, being $F(x, y)$ the speed at point x, y and $T(x, y)$ the value of the velocities map at x, y, we will have that the expansion speed depends on the position, and it is precisely the safe speed given by the velocities map. As this map provides the maximum safe speed of the robot, the obtained trajectory is the fastest path (in time) assuming the robot moves at the maximum allowed speed at every point.

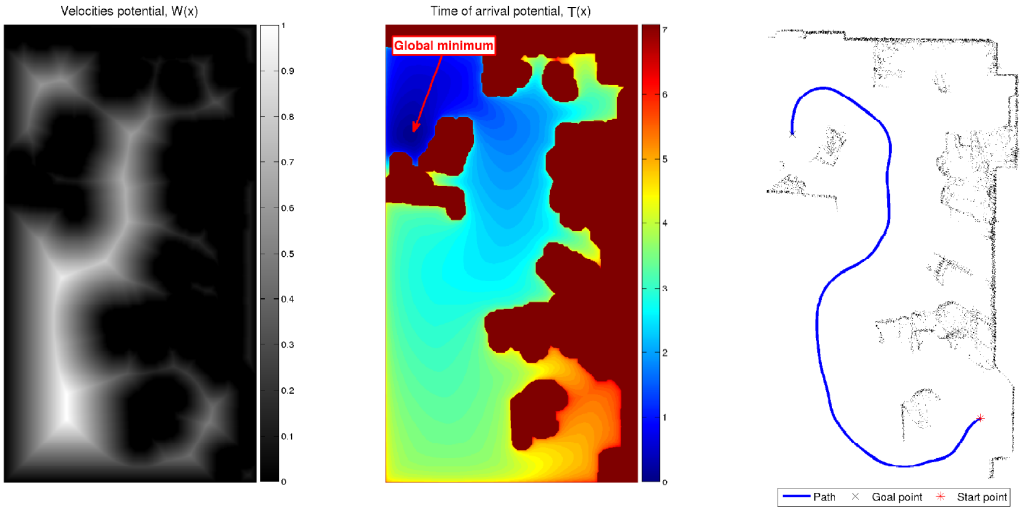


Figure 3.5: *Steps of the FM^2 applied to path planning.*

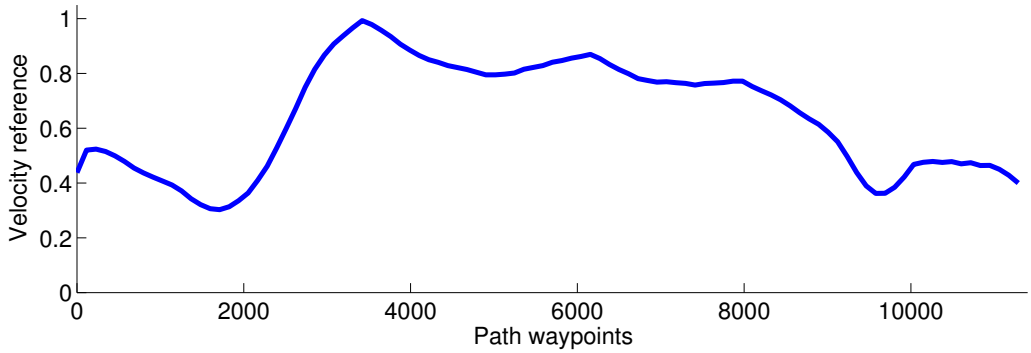


Figure 3.6: *Velocity profile of the path shown in Figure 3.5*

The FM^2 method is analogous to the Geometric Optics where light rays (trajectory in FM^2) travel in curved trajectories in media with changing refraction index (velocities map). Therefore, time optimality is justified by the principle of Fermat: *Light travels the path which takes least time*.

As a final remark, the FM^2 does not require to dilate the obstacles, since it is going to compute a very safe path for the robot. However, the binary map could be dilated before computing the velocities map.

3.2.1 Saturated variation of the Fast Marching Square Method

In most scenarios the trajectories provided by FM^2 are not logical nor optimal, even though the quality of the trajectory in terms of smoothness and safety is always good. The FM^2 computed trajectory, as it has been presented, tries to keep the trajectory as far as possible from obstacles. This computed trajectory is similar to the path computed with the Voronoi diagram (Siegwart & Nourbakhsh, 2004). But there are environments in which there is no need to follow a trajectory so far away from obstacles, as distance may be safe enough to navigate. To solve this a saturated variation of the velocities map can be used. When the first FMM has been computed, the velocities map is first scaled, to have continuous values between 0 and 1, and then saturated.

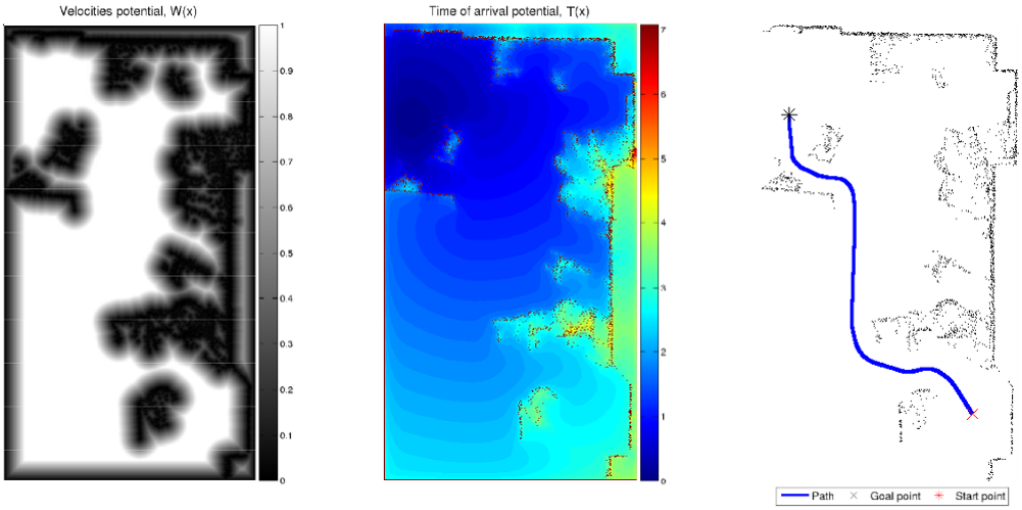
The scaling of the map is made according to two configuration parameters:

- Maximum allowed speed, which is the maximum control speed the robot may receive.
- Safe distance, which is the distance from the closest obstacle at which the maximum speed can be reached.

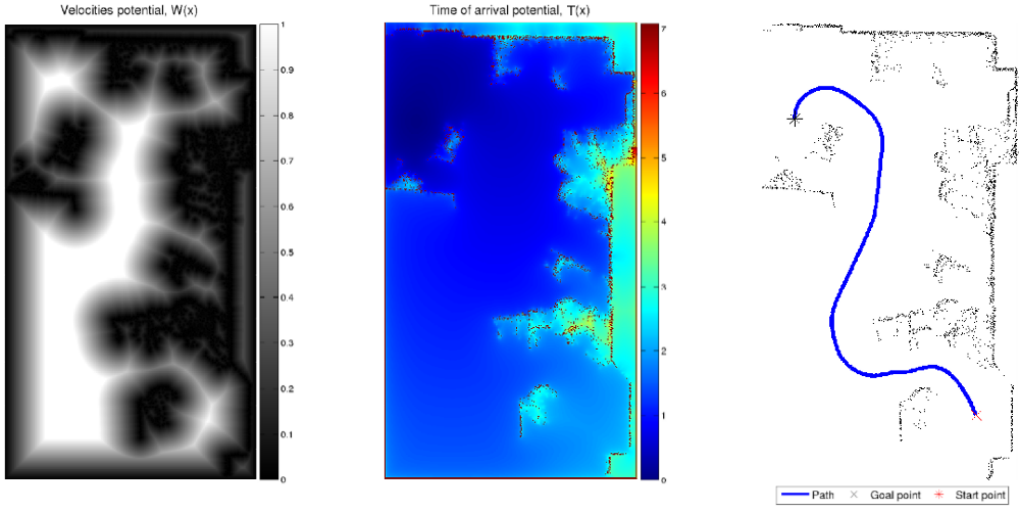
At the end the map is saturated to the maximum allowed speed. After this scaling and saturation process the velocities map provides the maximum speed for all the points that are farther than the safe distance from the obstacles and

the control speed varying form 0 (at obstacles) to the maximum speed (at safe distance) for the rest of points.

Figure 3.7 shows the saturated variation of Figure 3.5 with the new computed trajectory. We can appreciate that now the path is closer than a human could expect.



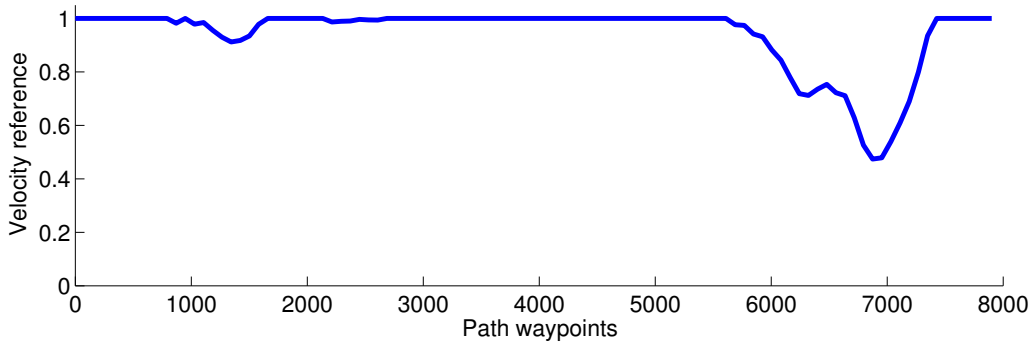
(a) *Saturation level: 0.3*



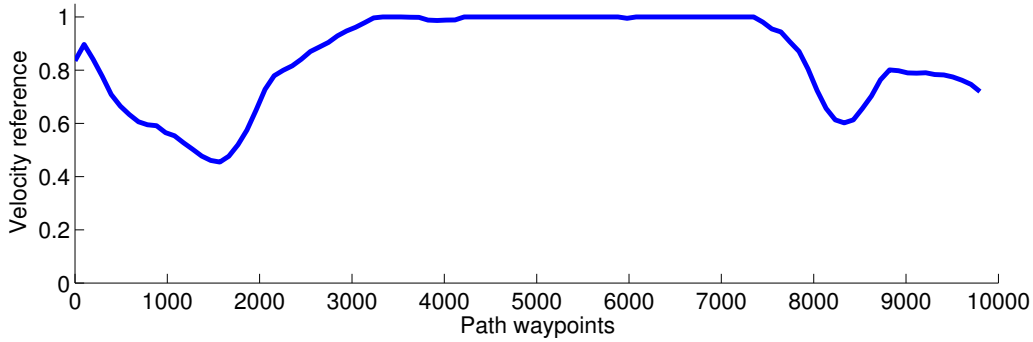
(b) *Saturation level: 0.6*

Figure 3.7: *Results of the saturated version of FM^2 for different saturation levels.*

In Figure 3.8 the speed profile with the saturated area is shown.



(a) *Saturation level: 0.3*



(b) *Saturation level: 0.6*

Figure 3.8: *Velocity profiles of the paths shown in Figure 3.7*

3.3 Conclusions

In this chapter we have presented the mathematical foundations of the FMM. We have presented the algorithm that we have implemented to apply the FMM over a grid map. Section 3.1.4 presents how the FMM method can be applied to compute the trajectory among two points in a grid map.

The limitations of this methodology lay in the fact that it provides the shortest path in distance, which leads to risk due to its closeness to the obstacles. In Section 3.2 the FM^2 has been explained in detail. The FM^2 computes two wave

expansions over the grid map. The first expansion computes the velocities map that provides the maximum allowed speed of the robot at each point of the map. This velocities map is used to compute the second expansion, form the target point to the initial point. As a result of the second expansion, the trajectory is computed (using the maximum gradient direction). This solution provides both a path (way point) and the control speed at each point. As a result, this trajectory is safe and optimal in time. The FM^2 computes paths that tend to navigate far from obstacles, but this situation is not always necessary.

Initially, the FM^2 method can be understood as just another path planning method. However, as shown in the next chapters, this method is highly versatile, being possible to integrate it into more complex problems based on path planning, such as robot formations path planning, trajectory learning and so on.

Chapter 4

Robot Formation Path Planning based on the FM^2 Method

4.1 Introduction to robot formation motion planning problem

Due to their wide range of applications (surveillance, cooperative mapping, etc), robot formations have become one of the most insteresting topics in robotics research. Although a single robot is currently able to perform very complex tasks on its own, some of these tasks can be performed in a more efficient way using a group of robots.

The final objective in robot formation path planning is to find the paths and poses (positions and orientations) for each robot of the formation, taking into account the characteristics of the environment, the others robots in the formation, and the final objective. Therefore, the robot formation should be able to move throughout the scenario adapting the shape of the formation to their needs.

One of the main difficulties is that the position of the robots or obstacles is not totally accurate. This uncertainty becomes dangerous when the formation

must navigate through narrow passages, sharp curves or in harsh environmental conditions. In these situations, robots could crash into each other.

This chapter is organized as follows. Section 4.2 details the basic algorithm to solve the robot formation path planning based on FM^2. In section 4.3 the basic algorithm is improved, decreasing computation time and including uncertainty. Finally, in section 4.4 an outline of the application to 3 dimensions of the proposed method is included.

4.2 Basic robot formation planning algorithm using FM^2

In this chapter, the leader-followers scheme is used for robot formation path planning. The pose reference for the follower robots are defined by geometric equations, placing the goal point of each follower as a function of the leader's pose. The leader can be a robot, another vehicle, a person or even a *virtual leader*, which is a hand-defined point, usually by geometric relations.

The algorithm described next is an adaptation of the one proposed in (Garrido et al., 2011) to the FM^2 path planner method. This change is motivated by the fact that FM^2 is an improvement of the VFM planning method. Hence, the robot formation path planning is based on a state-of-the-art algorithm. There also exist two other advantages to using FM^2: it is easier to implement than VFM and it provides a continuous velocities map, whilst the VFM provides a velocities map with discrete gray level.

The FM^2 method provides a two-level artificial potential which repels the robot from the walls and obstacles. On the other hand, robot formation motion control requires additional repulsive forces between robots. Working only with the artificial repulsive potential given by the FM^2, the robots of the formation could crash into each other. Thus, integrating the potential given by FM^2 and the repulsive force between robots, each robot has at each moment one single potential attracting it into the objective but repelling it from obstacles, walls and other robots. The main requirement when integrating all the potentials is to do

it in a way that does not create local minima.

The FM^2 uses a two-step potential to compute the path: the first step creates a potential which can interpreted be as a velocities potential, which we denoted as $W(\mathbf{x})$; and the second step creates a funnel shaped potential, which represents the distance to the goal in the metrics $W(\mathbf{x})$ and is denoted as $D(\mathbf{x})$.

The robot formation path planning algorithm using FM^2 is the following:

- The environment map $\mathbf{W_0}$ is read as a binary map, where 0 (black) means obstacles or walls and 1 (white) means free space. This map is common for all the robots in the formation (both leaders and followers).
- The first potential $\mathbf{W}$ is calculated applying the FMM to the binary map $\mathbf{W_0}$, according to the FMM 1st step of the FM^2 method (section 3.2).
- The second potential $\mathbf{D}$ is calculated applying the FMM on the potential $\mathbf{W}$.
- An initial path for the leader is calculated applying gradient descent on the potential $\mathbf{D}$, according to the FM^2 method.
- So far the algorithm described is the application of FM^2 to the leader of the formation. Then a loop begins in which each cycle represents a step of the robots' movement. This loop consists of:

    1. For each cycle t, each robot i (both leader and followers) includes in its binary map $\mathbf{W_{0i}^t}$ the other robots in the positions $(x_j, y_j) \forall j \neq i$ (in 2D case) as black points, representing obstacles.
    2. For each cycle t, each robot i generates a new first potential $\mathbf{W}_i^t$ from $\mathbf{W_{0i}^t}$.
    3. From the leader's pose and the desired formation geometry, the partial goal (x_{gk}, y_{gk}) is calculated for each follower k (where k represents all the followers of the formation). The shape of the formation

is deformed proportionally to the grey level of the partial goal's position. Thus, the formation is adapted to the environment moving farther from obstacles and walls and also avoiding collisions with other robots (which are treated as obstacles). This way, the repulsive force between robots and walls and also the repulsive force between robots are implemented. The initial geometry of the formation and how it is affected by the environment is shown in Figure 4.1.

4. The potentials $\mathbf{D}_i^t$ are calculated applying the FMM to the metrics matrices $\mathbf{W}_i^t$. For the leader the goal point is the end point of the path. The goals of the followers are the partial goals computed on the previous step. The low computational cost of FM^2 allows us to do this without compromising the refresh rate.
5. The path is calculated for each robot i. This path is the one with the minimum distance with the metrics $\mathbf{W}_i^t$ and it is obtained applying gradient descent on the potential $\mathbf{D}_i^t$.
6. All the robots move forward following their paths until a new iteration is completed.

The aforementioned algorithm is summarized in the flowchart of Figure 4.2. It is a base which assures the correct navigation of a robot formation through different environments, avoiding obstacles and adapting to narrow passages. In (Garrido et al., 2011) many additional techniques are proposed to improve the time or behaviour performance of the algorithm. These techniques such as maximum energy configuration, using a tube around the path to decrease the computational cost, or adding springs can be applied to this algorithm with very similar results. In addition, this algorithm can be applied to any kind of robot formation, with real or virtual leaders. Figure 4.3 shows the steps of the algorithm on a triangle-shaped robot formation. This shape has been chosen because it is easier to analyse the behaviour of the followers. In the Figure 4.4 the complete sequence of movement is shown.

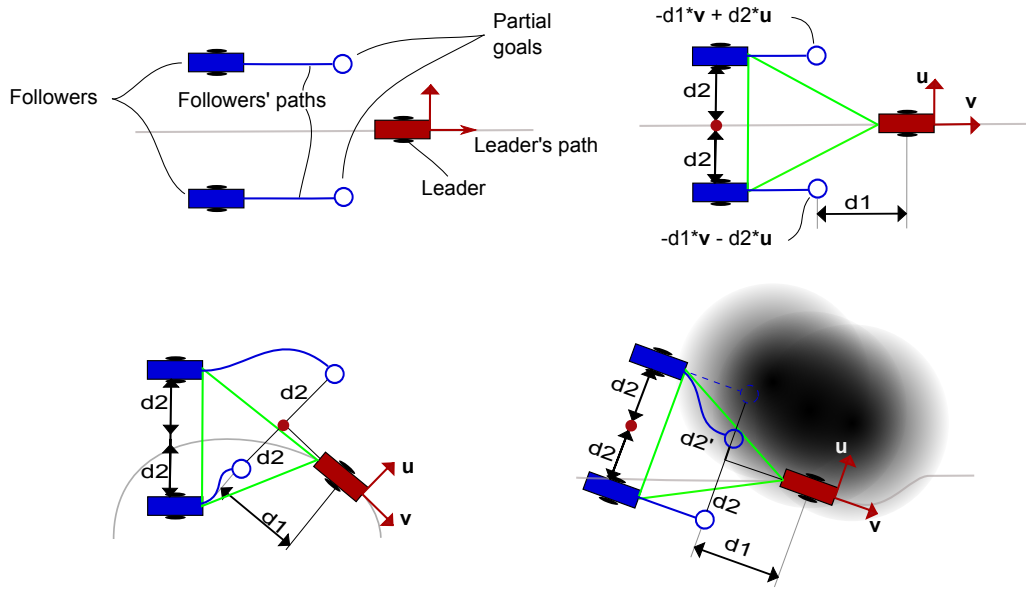


Figure 4.1: *Top left - Main components of the robot formation algorithm. Top right - Reference geometric definition of a simple, triangle-shaped robot formation. Bottom left - Behaviour of the partial goals depending on the leader's pose. Bottom right - Behaviour of the partial goals depending on the obstacles of the environment.*

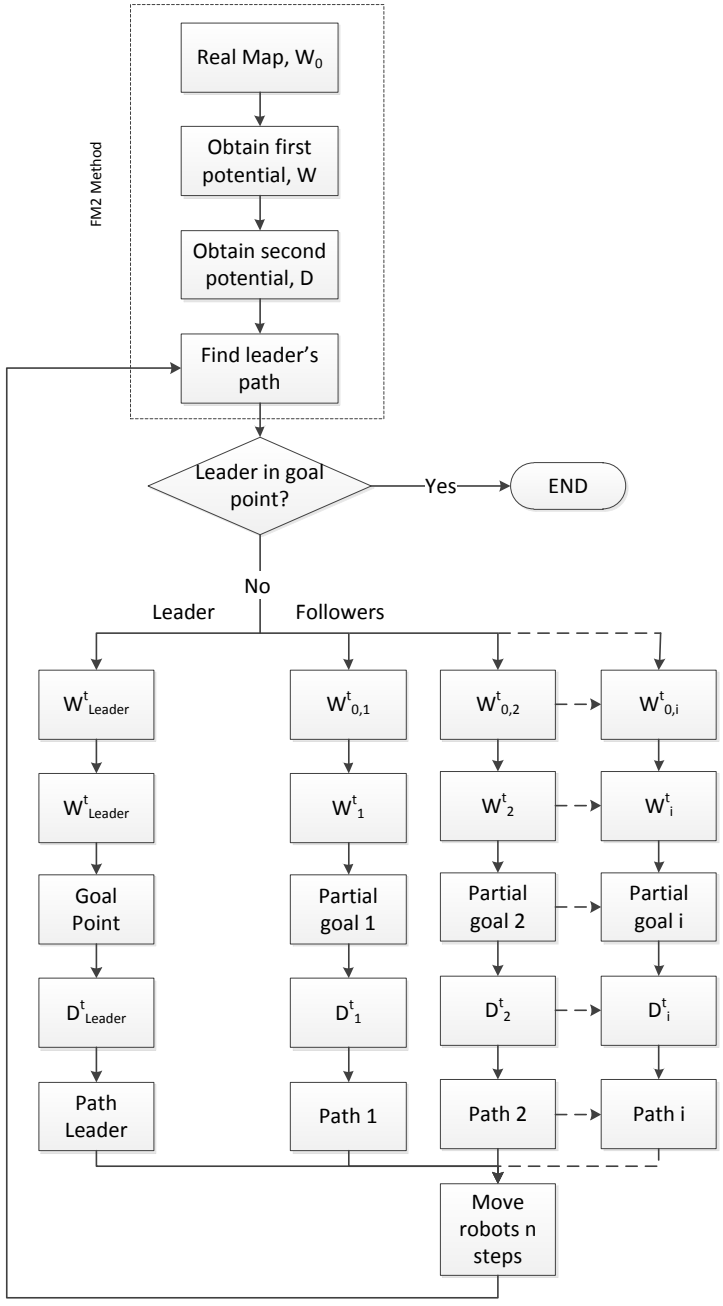
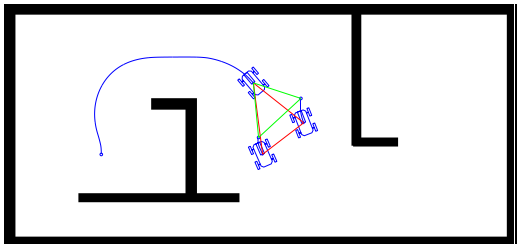
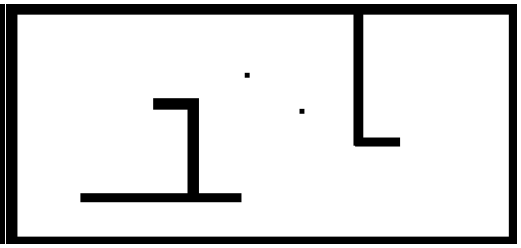


Figure 4.2: *Flowchart of the basic robot formation planning algorithm using FM^2.*

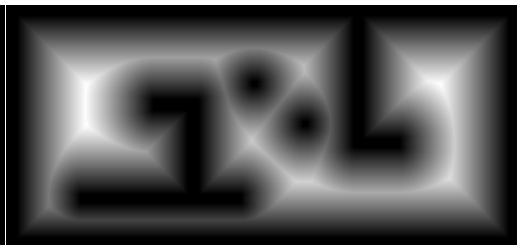


(a)



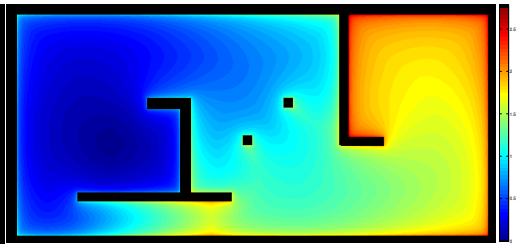
(b) *Follower 2 considering the other robots as obstacles.*

(c) *First potential,* $\mathbf{W}_1^t$*, for the follower 1.*

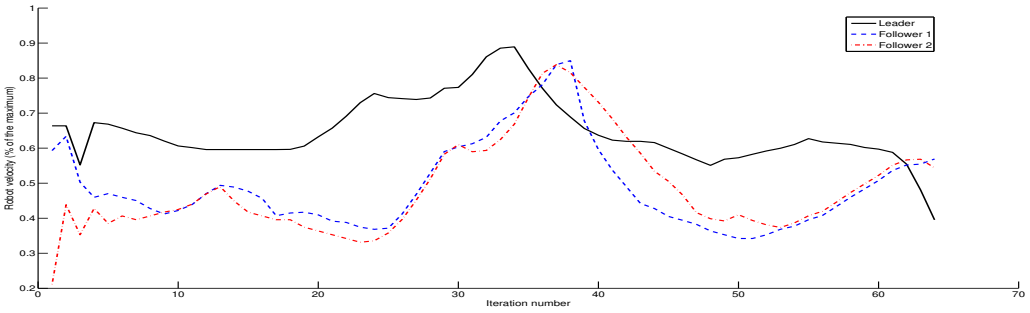


(d) $\mathbf{W}_2^t$.

(e) $\mathbf{W}_{leader}^t$.



(f) *Second potential,* $\mathbf{D}_{1eader}^t$*, for the leader*



(g) *Reference velocities for each robot of the formation.*

Figure 4.3: *a) Snapshot of the formation moving. The leader follows the blue path. The green triangle (leader-partial goals) is the desired formation and the red triangle (leader-followers) is the current formation.*

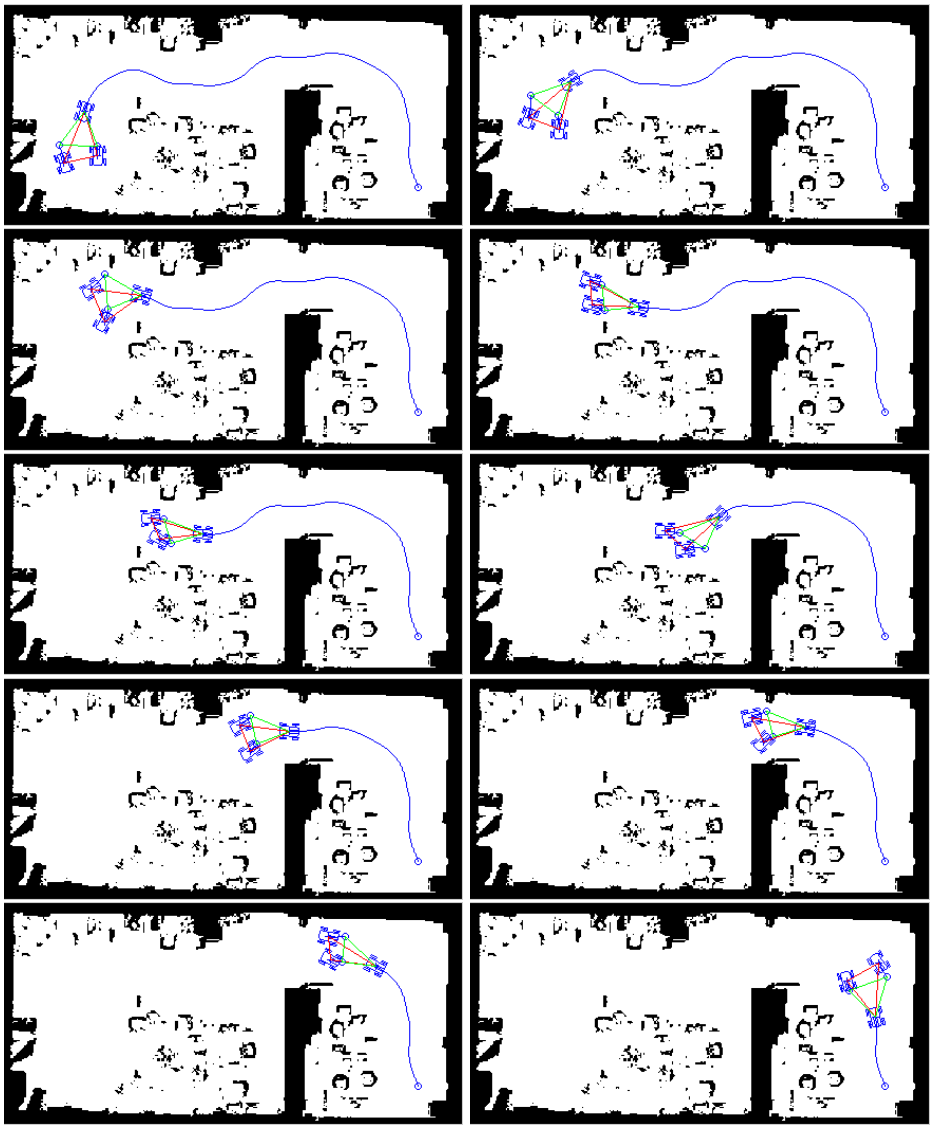


Figure 4.4: *Sequence of movement of a robot formation with the basic path planning algorithm, simulated in a map of our laboratory obtained with SLAM techniques.*

4.3 Inclusion of uncertainty on the basic algorithm

In the previous work the obstacles were included in the initial binary map. Here the obstacles and the other robots of the formation are included in the velocities map, allowing to easily include a degree of uncertainty in the position of the obstacles and robots. This modification also improves enormously the computational cost of the algorithm, since the velocities map is not calculated once for every robot and every iteration.

In the implementation phase, there are two approaches that can be used. The first one is decentralized control. This is based on using completely autonomous robots, which detect the environment and obstacles with their sensors, compute their localization and communicate their positions to all the other members of the formation. This requires a complex communication protocol and the uncertainties are high. On the other hand is centralized control, where one main computer receives all the information through sensors and communicates the decisions directly to the robots. In this case, the sensors could be a camera above the robots or a motion capture system. The uncertainties of this approach are usually lower and its implementation is easier. Although both approaches are suitable for our proposed algorithm, we think it is easier to demonstrate by means of the second approach. An example of a low-cost, easy implementation is shown in Figure 4.5. Of course, these strategies are not error-free and have an uncertainty associated due to sensor noise and measurement errors.

In the proposed method, each robot i of the formation has its own first potential $\mathbf{W}_i^t$ depending on time. This potential is defined by the global first potential $\mathbf{W}$ (defined by the map) in which an uncertainty function is included for each robot of the formation.

Let us suppose that robot i of the formation is in the position (x_i, y_i). This position has an error, since it has been calculated using sensor information. With the dimensions of the robots known, namely $l_i \times w_i$, the robot j takes into account the position of robot i and its uncertainty as follows:

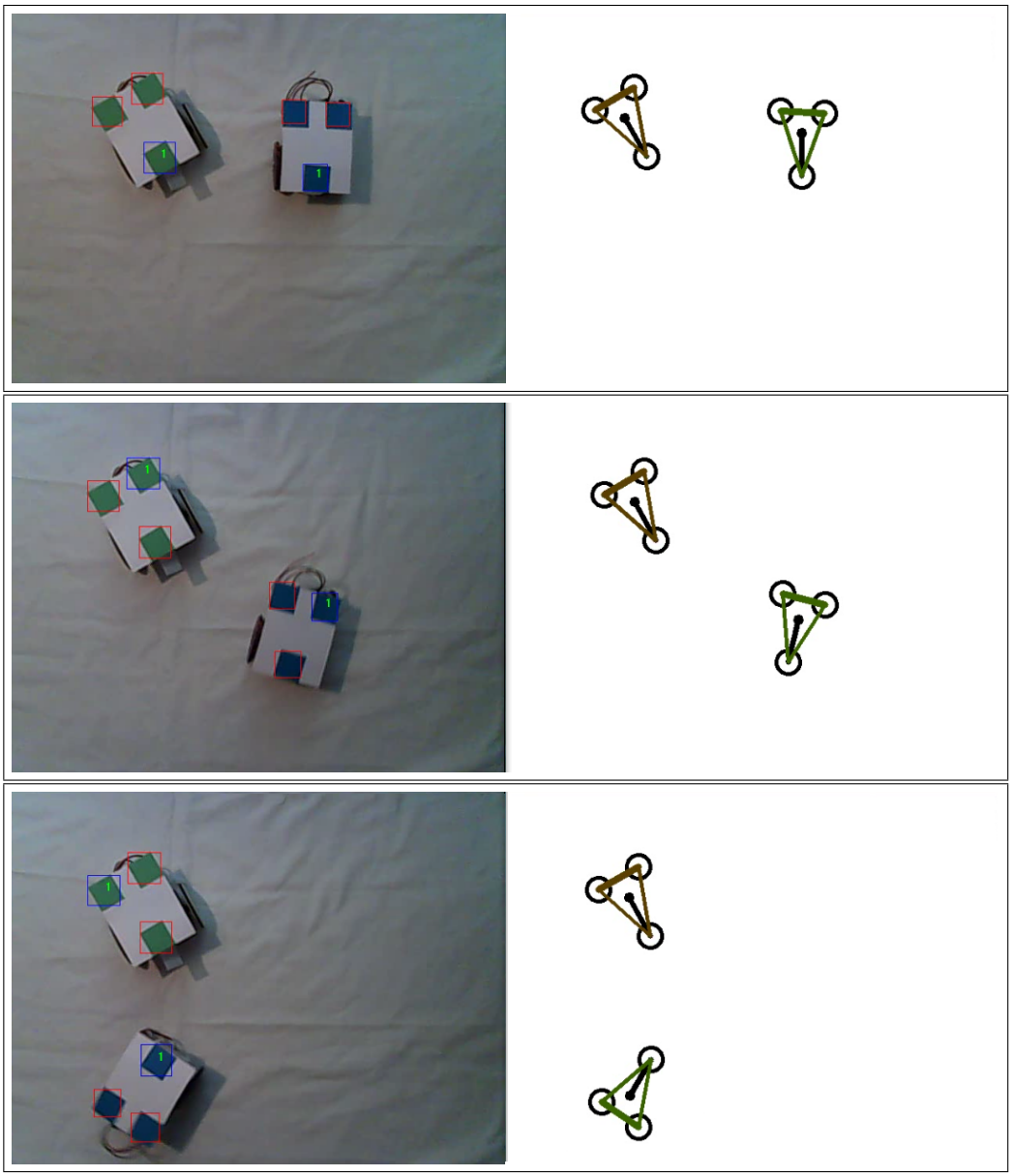


Figure 4.5: *Example of a system which is able to capture the position and orientation of the robots by using a camera and colour labels on the robots.*

- A map is created in which all is a gray space with uniform value $0 \leq \alpha \leq 1$, where α means the uncertainty level (1: totally uncertain, 0: no uncertainty).

- In the middle of this map a zone with value 0 is included. The size of this zone is equal to the dimensions $l_i \times w_i$, representing the robot on the measured position.

- The FMM is applied to this map using the position of robot i as the origin. Thus, a gray scale map is generated where the highest values depends on the size of the map and the uncertainty. The minimum between the gray scale map and 1 is calculated in order to set the maximum value (white). Then, this map can be interpreted as a *uncertainty function* $\mathbf{Wr}_i$ where white (1) means that it is quite certain that robot i is not in those points, and black (0) means that robot i is certain to be in those points. The uncertainty function should not depend on the time, since this uncertainty appears because of the sensor noise and it is supposed to maintain itself in the same range of values.

- Calculate the minimum between the first potential $\mathbf{W}_j^t$ of robot j and the uncertainty function $\mathbf{Wr}_i$. Thus, $\mathbf{W}_j^t$ is updated with the position of robot i with the uncertainty included.

$$\mathbf{W}_j^t = min(\mathbf{W}_j^t, \mathbf{Wr}_i) \tag{4.1}$$

  In the initialization, the first potential for the robot is equal to the global first potential, $\mathbf{W}_j^t = \mathbf{W}$.

- For the robot j, this process is repeated for all the other robots i in the formation. At the end, the first potential $\mathbf{W}_j^t$ will include an uncertainty function for every other robot in the formation.

This algorithm can be integrated into the one described in section 4.2 by including it in place of steps 1 and 2 of the loop.

With this method, summarized in Figure 4.6, one robot in the formation (leader or follower) is able to calculate the path to its objective taking into account the global map and the other robots' position with its uncertainty included. This way, the robot will navigate far from places that are obstacle-free but the velocity is slow, and it will also avoid places were the velocity could be high but it is not possible to assure safety.

The steps of the algorithm and its details are shown in Figure 4.7. Full sequences of movements are shown in Figures 4.8, 4.9, 4.10, testing different robot formation shapes.

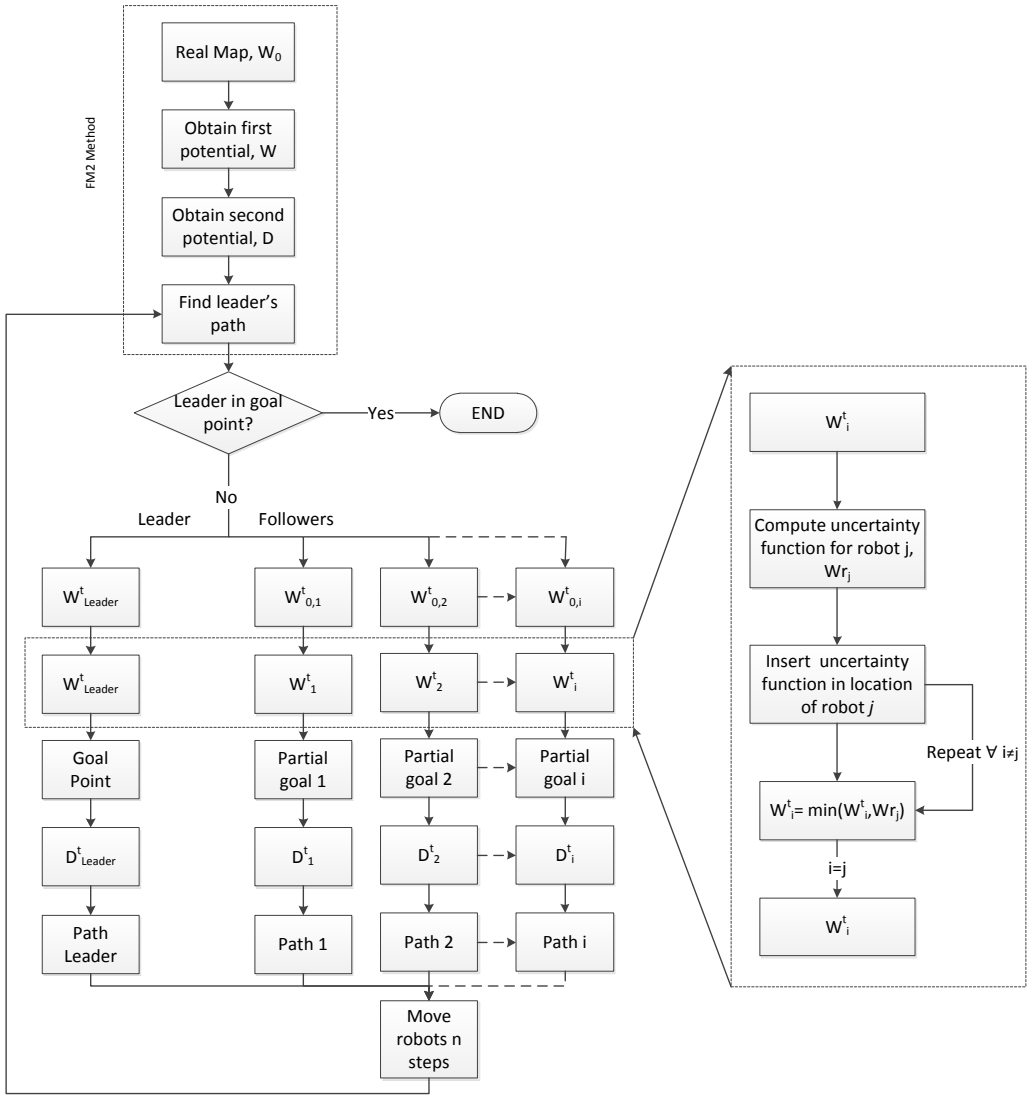
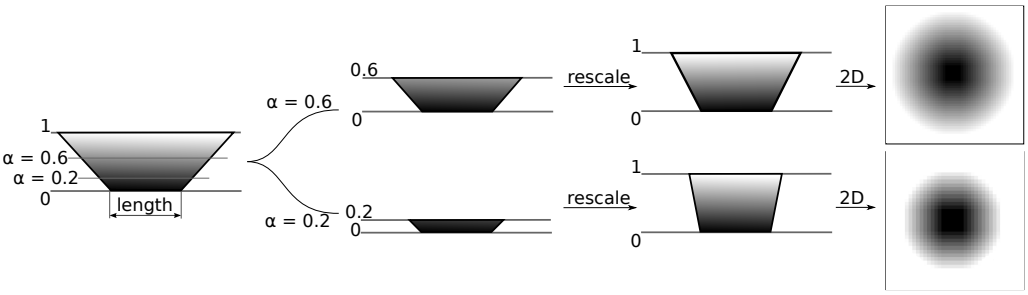
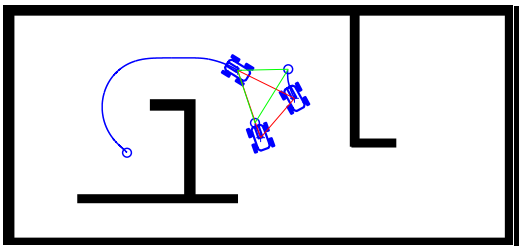


Figure 4.6: *Flowchart of how the velocities map is modified for every robot in the formation.*

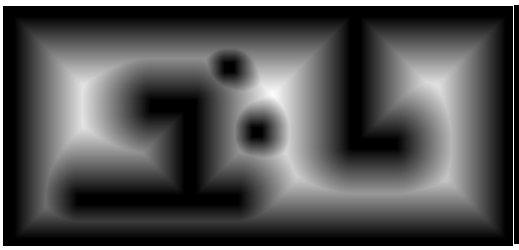


(a) *Intuitive generation of the uncertainty function* $\mathbf{Wr}_i$ *depending on* α *for one dimension and its extension to 2 dimensions.*

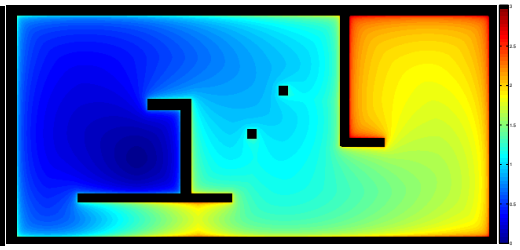


(b) *Current position of the robot formation.*

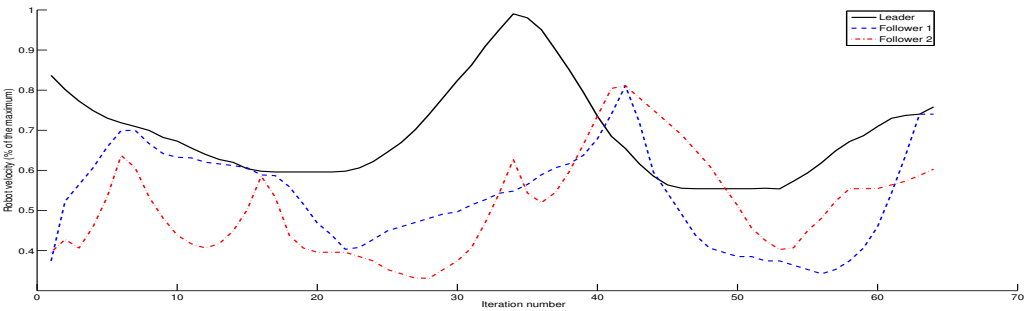
(c) $\mathbf{W}^T_{leader}$



(d) $\mathbf{W}^T_1$



(e) $\mathbf{D}^t_{1eader}$



(f) *Reference velocities for each robot of the formation.*

Figure 4.7: *Steps of the formation planning method including uncertainty.*

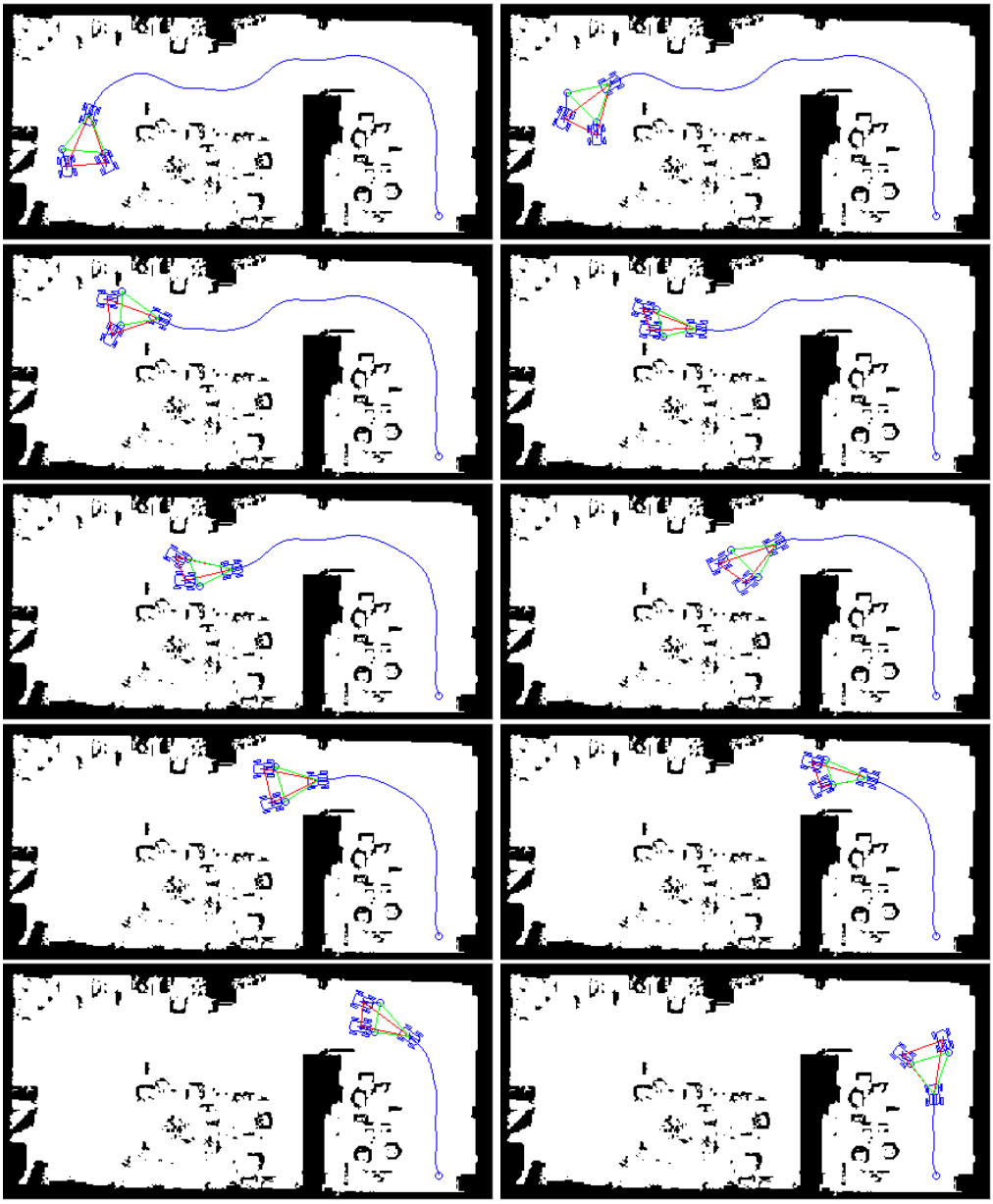


Figure 4.8: *Sequence of movement of a robot formation with the proposed path planning algorithm.*

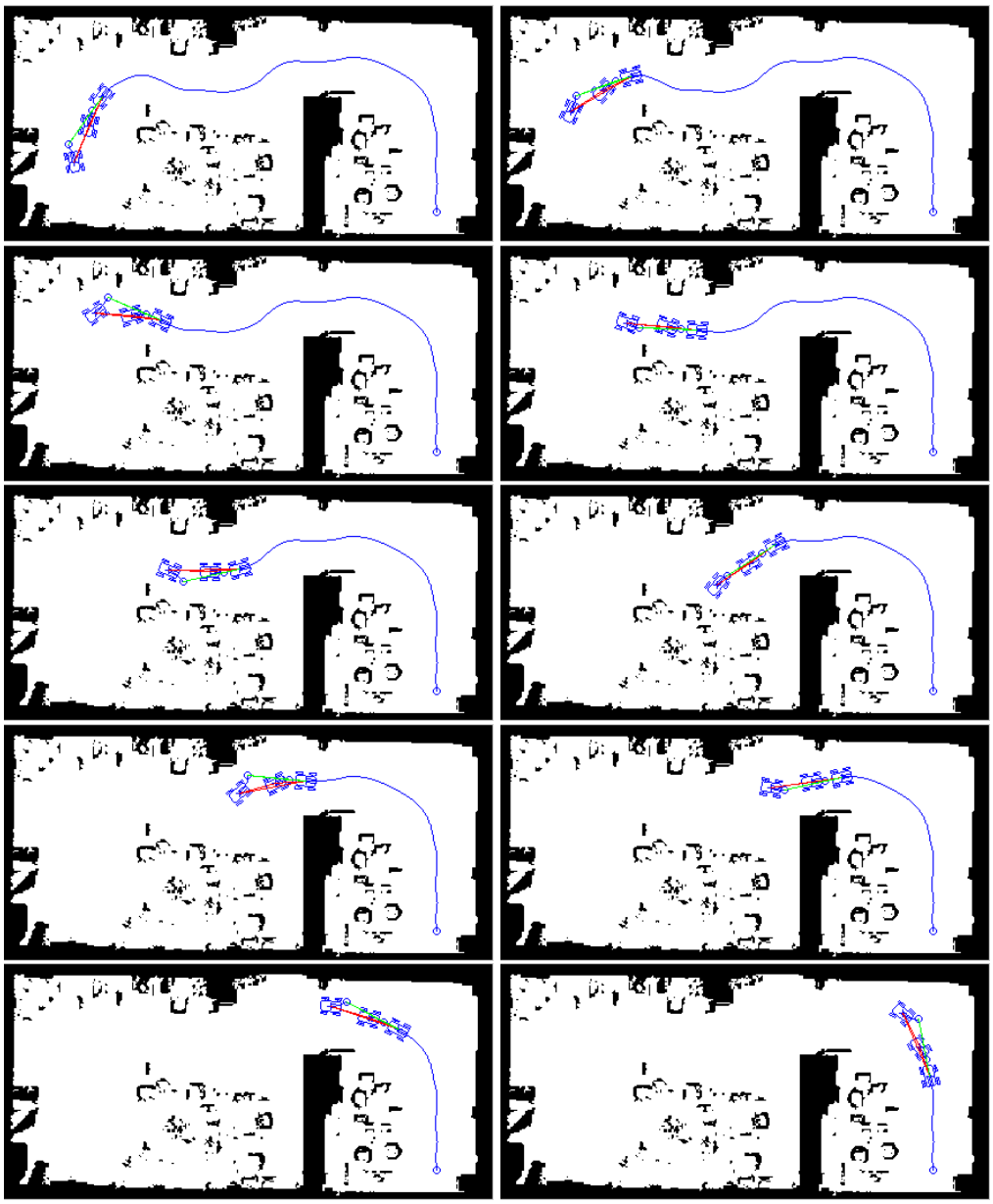


Figure 4.9: *Sequence with a different formation. This time, 3 robots travels in a line.*

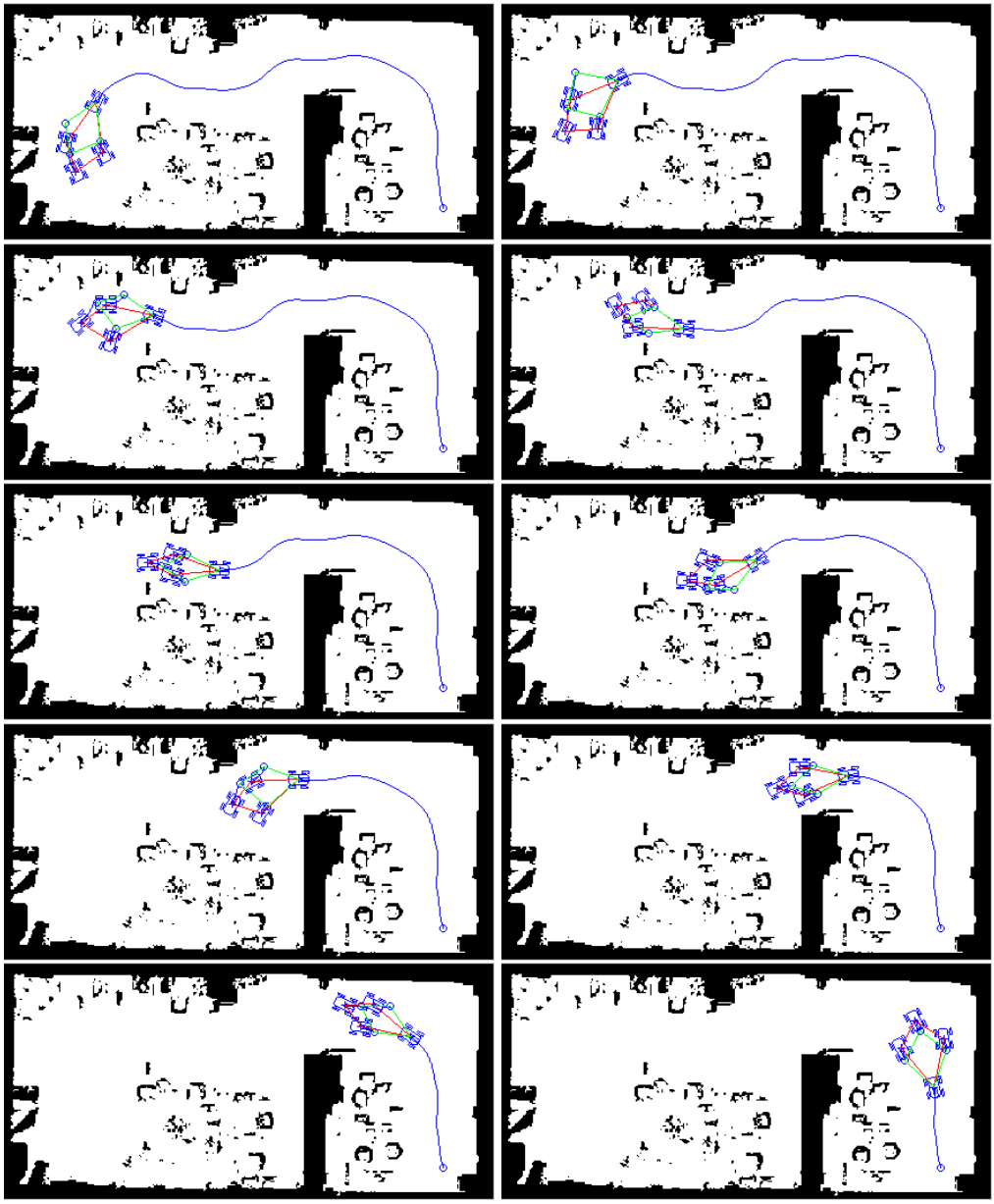


Figure 4.10: *Sequence of movements of a robot formation composed of 4 robots.*

For n robots, the FMM must be applied n times (one per robot), which increases the computational cost. To reduce this cost, the uncertainty function is computed on a smaller map and is later added to a bigger map. In our simulations, the map on which the uncertainty function is applied has a size of 10 times the dimensions of the robots. Moreover, if all the robots of the formation are of the same size, it is only necessary to compute the uncertainty function once and later include it in all the positions needed, avoiding unnecessary computational cost.

Comparing Figures 4.4 and 4.8, it is possible to see that the motion of the formation is not highly modified. However, the inclusion of the other robots as uncertainty functions in the velocities map has many advantages: in the basic algorithm there were places in the velocities map which were far from the robots but still influenced their movement. With the approach shown herein robots are only taken into account within their uncertainty area, see Figure 4.11. Therefore, the robots behave normally until they are in places were other robots could be. The proposed approach allows dealing with uncertainty in a very intuitive way, avoiding complex probabilistic modelling. Furthermore, in the basic algorithm the velocities map had to be calculated in every loop cycle. This supposes an average computation time of 1.5 ± 0.1 seconds in a 625×293 pixel map. With the new approach the computation time of each iteration is 0.82 ± 0.03 seconds for the same map.

4.3.1 Velocity saturation

In environments with large, open areas the FM^2 can provide good trajectories but they can be improved since in most situations it is not necessary to move through the safest path but through one that is safe enough. For instance, in an open room it may not be necessary to go through the middle of the room because it is enough to keep a minimum distance from the walls. To solve this a saturated variation of the velocities map $W(\mathbf{x})$ has been implemented. This results

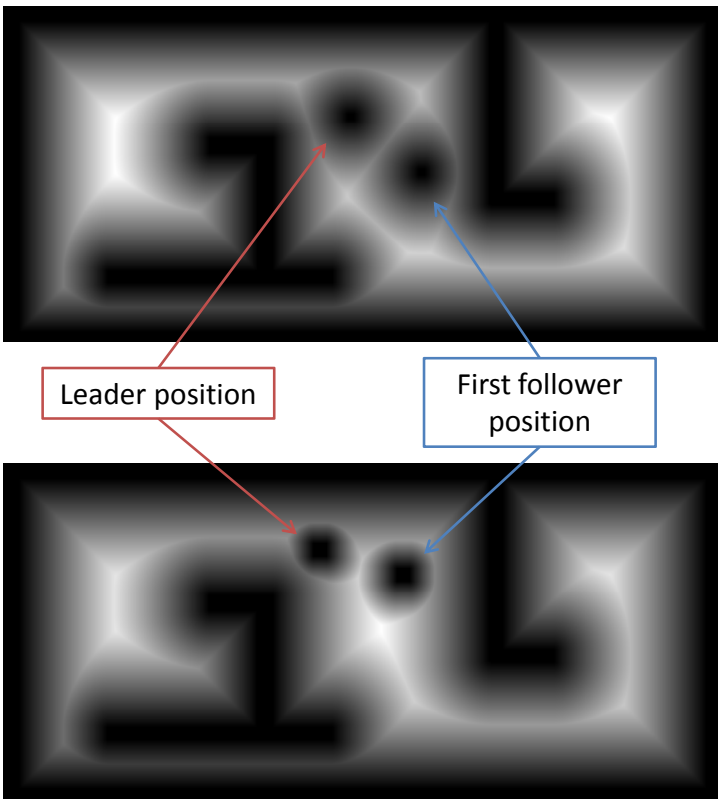
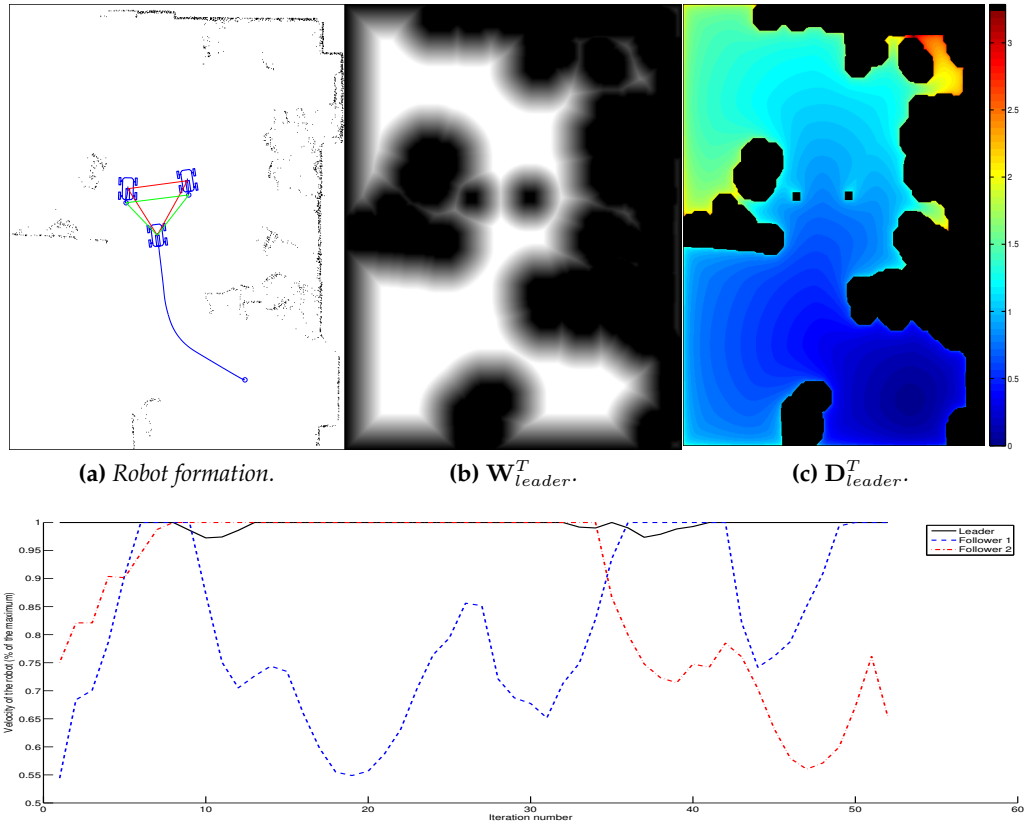


Figure 4.11: *Comparison of the velocities map created for the second follower with the basic algorithm (top) and using uncertainty functions (bottom).*

in a maximum velocity in open areas which decreases when the robot is close enough to the walls or obstacles. This has already been proposed in (Garrido, Moreno, Abderrahim, & Blanco, 2009) for single robot motion, improving the trajectories, which are closer to the optimal path in distance and making it more human-like. Here, velocity saturation is applied to a robot formation, which allows the geometry to have less deformation since the velocity is constant in most points.

Figure 4.12 shows the underlying characteristics of this variation.



(a) *Robot formation.* **(b)** $\mathbf{W}^T_{leader}$. **(c)** $\mathbf{D}^T_{leader}$.

(d) *Reference velocities for the robots*

Figure 4.12: *Steps of the robot formation algorithms with saturated velocities map.*

A full sequence of movement is included in Figure 4.13. It should be noted

that this modification will require faster and more agile robots, since the shape is not deformed until any robot of the formation is close enough to an obstacle. Thus, while the advantage of this variation is that the formation maintains its predefined shape for a longer period of time. However, the drawback is that it usually generates sharper curves. This version of the proposed algorithm does not include any modification in the computation time for every iteration.

4.3.2 Mobile obstacles

The 50% reduction in computation time encourages a deeper study in dynamic environments. In most robotic applications, there will be two types of obstacles: static, such as walls; and dynamic, like people walking around, doors, etc. In a real application, a robot formation must be able to change its path according to the dynamic obstacles in the scenario.

Since the leader of the formation is recalculating its complete path in each iteration, the path will always be collision-free for the leader. The followers compute their path to the partial goals, so mobile obstacles do not represent a problem for the followers until they are close to them. The obstacles can be detected in many different ways: cameras, robot sensors, motion capture systems, etc. As for the robots, when an obstacle is detected, its position (and also velocity) will be measured with an associated error due to sensors noise. It is possible to deal with this uncertainty in the same way as done in section 4.3:

- The leader obtains a safe, collision-free path, avoiding obstacles which could become a problem in the following steps.

- The position of the obstacle and its size have some degree of uncertainty. Then, the algorithm in section 4.3 is used in order to take that uncertainty into account: an uncertainty function is calculated for each mobile obstacle, depending on its size and velocity. All the generated functions are included in the velocities map of the robots.

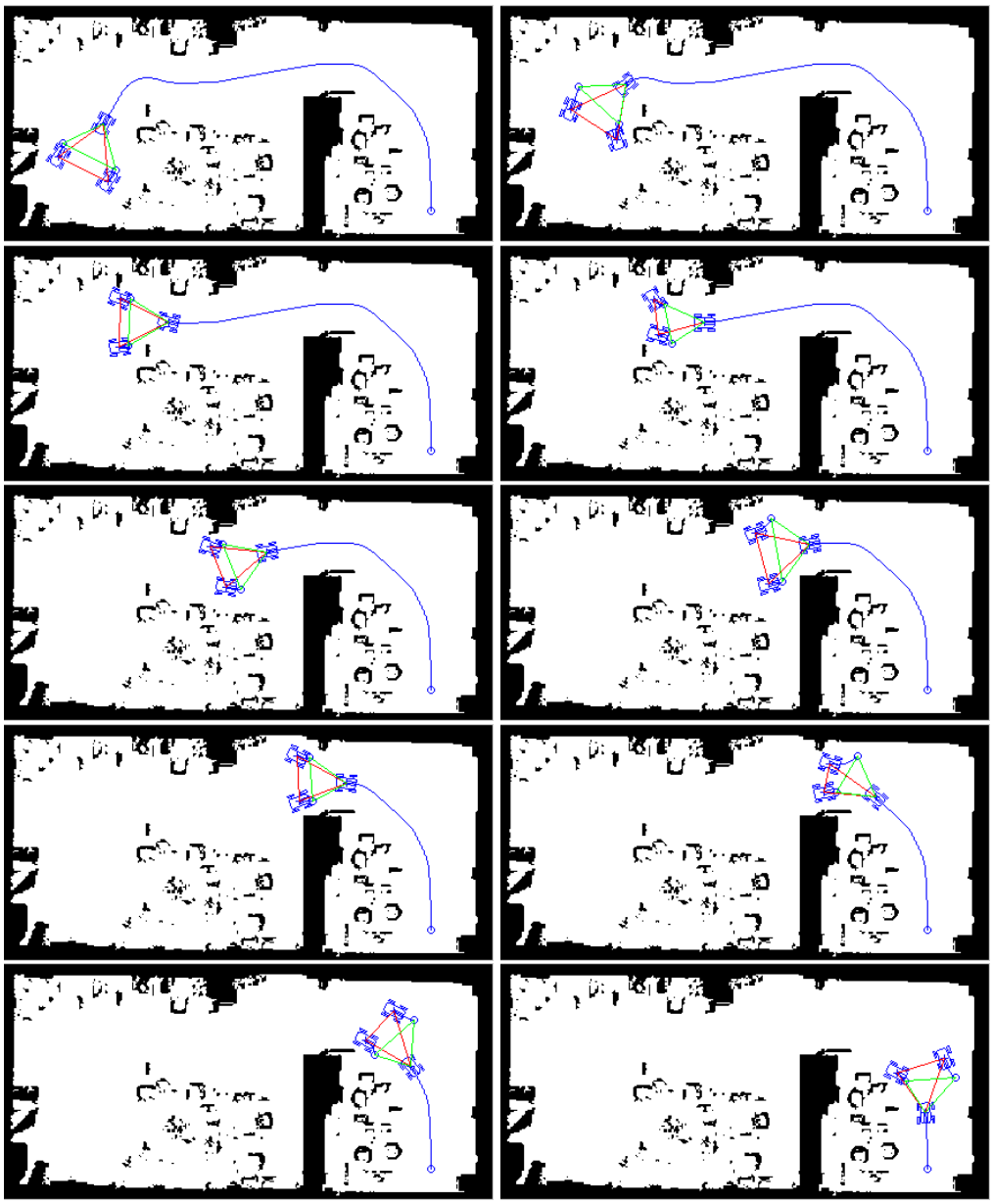


Figure 4.13: *Sequence of movements of a robot formation using velocities map saturation.*

With this method a low computational cost is achieved when dealing with dynamic obstacles, since the underlying algorithm is the same proposed in section 4.3. The only modification is that the obstacles detected are included in the first potential of all the formation robots. The inclusion of mobile obstacles is detailed in Figure 4.14. A complete sequence of movement in a real laboratory, obtained through a 360° field-of-view range scanner, is shown in Figure 4.15, where velocity saturation was also applied. Predictive algorithms such as (Tao, Faloutsos, Papadias., & Liu, 2004; Shen, 2009), can be used to predict those movements and set the partial objectives accordingly.

Other methods to include uncertainty in mobile obstacles, based on multidimensional Gaussian functions have already been proposed (Wang, 2009). The main advantage of the proposed method is that using FM^2 has a similar result and it is easier to implement. Also, the way Gaussian functions can be merged with the FM^2 requires a deep study while the advantages are not remarkable in comparison with the proposed method.

4.4 Formation planning algorithm in 3D

Since FMM and FM^2 can be expanded to more than 2 dimensions, the algorithm detailed in this chaper is also applicable to 3 dimensions.

However, a problem arises when dealing with the formation geometry in 3 dimensions. In the previous section, one of the references for the formation geometry was the tangent vector of the leader's trajectory. The rest of the references were computed from this one: normal vector and combinations of normal and tangent vectors (see Figure 4.1). If the same concept is applied in 3 dimensions, it turns out that there exist a infinite number of perpendicular vectors to the tangent to the trajectory, as can be seen in Figure 4.16. In fact, the perpendicular of a given vector is a plane.

In order to adapt the proposed robot formation planning method to 3 dimensions, a third reference has to be defined so the reference geometry of the

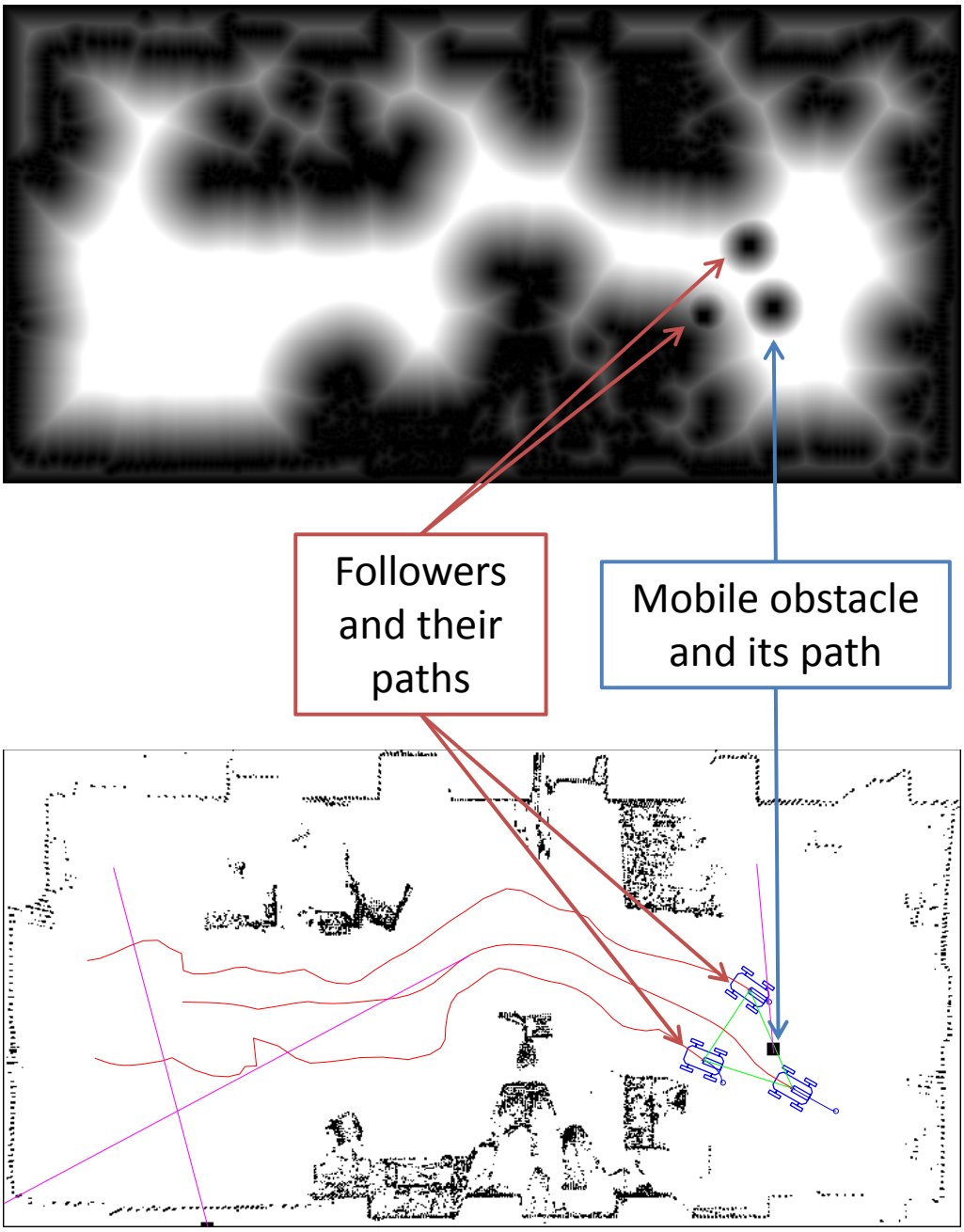


Figure 4.14: *Top - Velocities map of the leader at time T,* $\mathbf{W}^T_{leader}$*, with the followers and obstacles included with their uncertainty function. Bottom - Current position of the formation and the obstacle.*

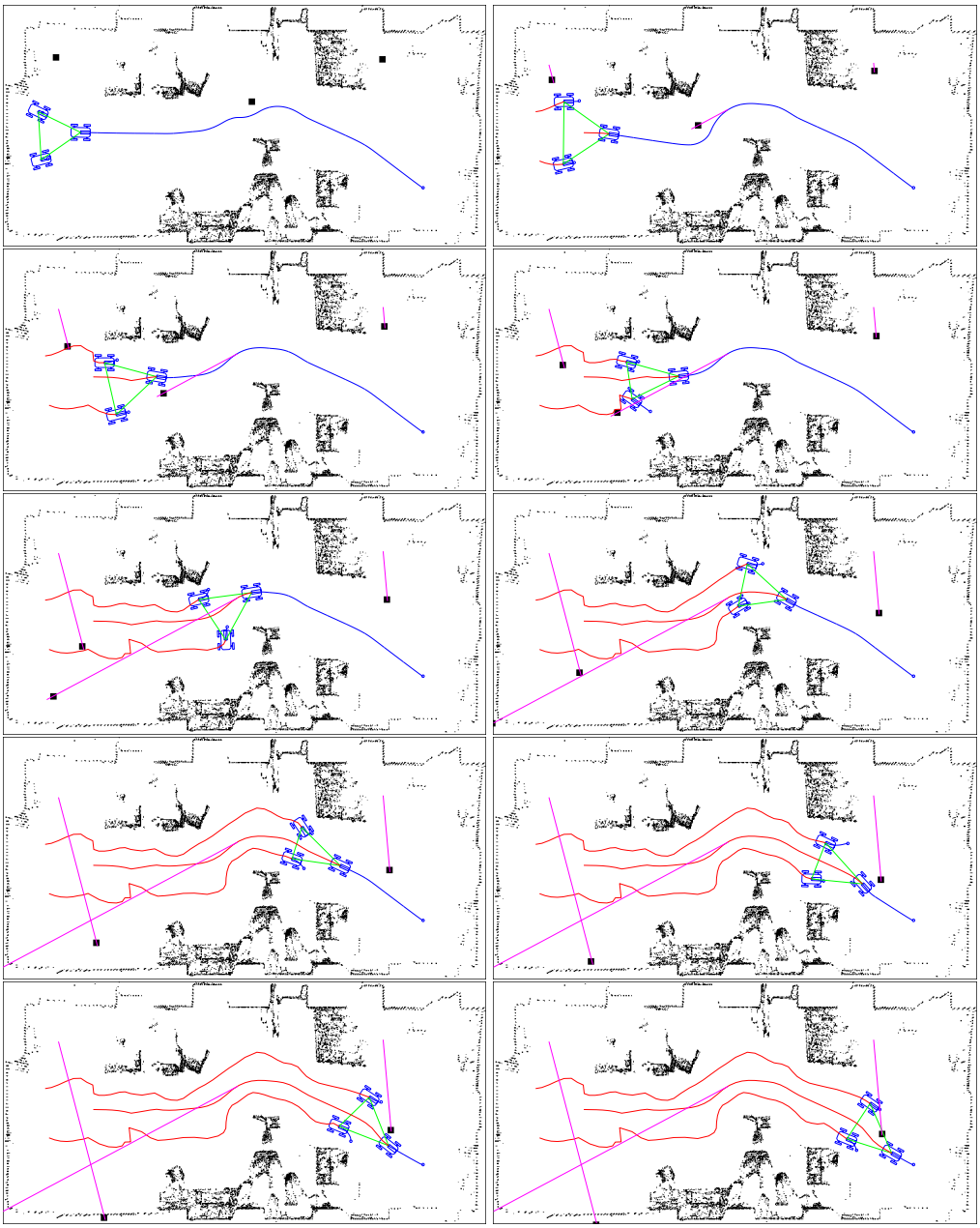


Figure 4.15: *Navigation sequence of a formation in a real environment. The trajectories of the robots (red) and obstacles (pink) are included (the sharp trajectories of the followers are due to simulation discretization).*

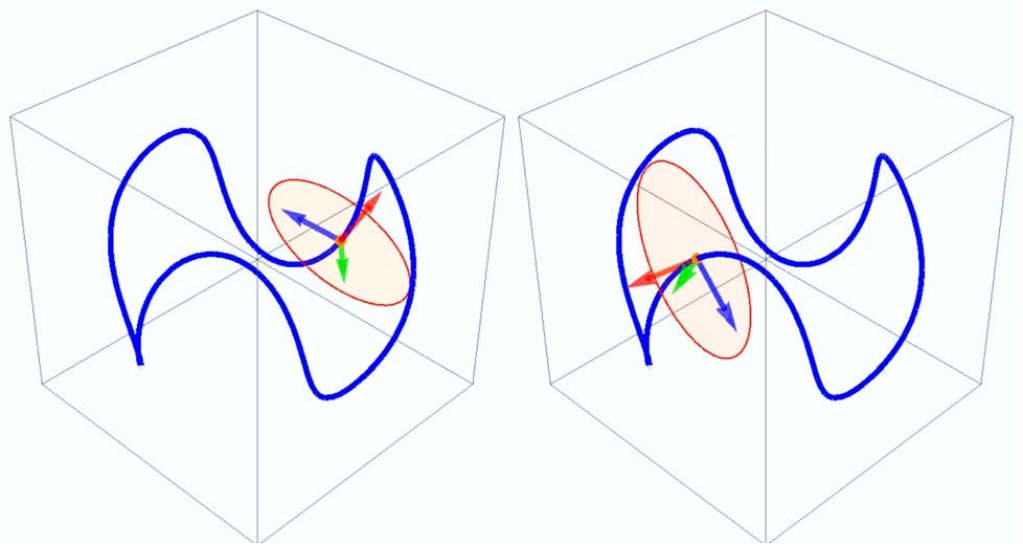


Figure 4.16: *The red vector represents the tangent to the trajectory and the red circle the perpendicular plane to this vector.*

formation can be defined. In our case, two different options are possible:

- Use an *absolute* reference, i.e., defining the normal vector as parallel (or perpendicular) to the ground of the robot formation environment. The main drawback of this approach is that the formation will be very rigid and could not adapt well to the environment, creating strange formation shapes in some situations.

- Use a *relative* reference, using the local characteristics of the path as a third reference. It allows the formation to be more *environment-independent* when defining the reference geometry.

In our case, we use a relative reference, based on the Frenet-Serret formulae (Frenet, 1852; Serret, 1851).

4.4.1 Frenet-Serret formulae

The Frenet-Serret formulae are used to describe the kinematic properties (velocity, curvature and torsion) of a particle which moves in a three-dimensional Euclidean space, $\mathbb{R}^3$. Let $\mathbf{r}(t)$ be a parametrization of a continuous, differentiable curve C in a Euclidean space $\mathbb{R}^3$. Let us denote $\mathbf{T}(t)$, $\mathbf{N}(t)$ and $\mathbf{B}(t)$ the unit tangent vector, unit normal vector and unit binormal vector respectively. Let us denote also the curvature as $\kappa(t)$ and the torsion as $\tau(t)$, the Frenet-Serret formulae are:

$$\begin{cases} \mathbf{T}'(t) = \kappa(t)|\mathbf{r}'(t)|\mathbf{N}(t) \\ \mathbf{N}'(t) = -\kappa(t)|\mathbf{r}'(t)|\mathbf{T}(t) + \tau(t)|\mathbf{r}'(t)|\mathbf{B}(t) \\ \mathbf{B}'(t) = -\tau(t)|\mathbf{r}'(t)|\mathbf{N}(t) \end{cases} \tag{4.2}$$

The Frenet trihedron, Frenet-Serret frame or TNB frame, is defined by the collection of the three vector functions $\mathbf{T}(t)$, $\mathbf{N}(t)$ and $\mathbf{B}(t)$ satisfying the following fundamental relations:

$$\mathbf{T}(t) = \frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|} \quad \mathbf{N}(t) = \frac{\mathbf{T}'(t)}{|\mathbf{T}'(t)|} \quad \mathbf{B}(t) = \mathbf{T}(t) \times \mathbf{N}(t) \tag{4.3}$$

The graphical representation of the Frenet trihedron is already shown in Figure 4.16, where the red vector is $\mathbf{T}(t)$, the blue vector is $\mathbf{N}(t)$ and the green vector represents $\mathbf{B}(t)$.

4.4.2 Implementation considerations of the Frenet trihedron

Since the FMM provides continuous trajectories it is possible to use the Frenet trihedron using as a curve C the path provided. However, due to the voxel grid environment representation, the paths in the implemetation are not continuous. If the grid size is small enough (this means that the path behaves as a continuous curve with this grid size) the vector functions $\mathbf{T}(t)$, $\mathbf{N}(t)$ and $\mathbf{B}(t)$ can be approximated by using numerical differentiation for every iteration i. Concretely, we

will use the centered differences of order n, $\delta_h^n[f](x)$. Hence:

$$\mathbf{T}(t) \parallel \mathbf{r}'(t) \tag{4.4}$$

satifying that $|\mathbf{T}(t)| = 1$ (as exposed in equation 4.3). Therefore:

$$\mathbf{r}'(t) \approx \delta_h[r](x) = \mathbf{r}(t + \tfrac{1}{2}h) - \mathbf{r}(t - \tfrac{1}{2}h) = \frac{\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})}{2\Delta t} \tag{4.5}$$

Normalizing to length 1:

$$\frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|} \approx \frac{\frac{\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})}{2\Delta t}}{|\frac{\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})}{2\Delta t}|} \tag{4.6}$$

Assuming that the time intervals Δt are equal among iterations, $t_{i+1} - t_i = t_i - t_{i-1}$, then:

$$\frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|} \approx \frac{\frac{\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})}{t_{i+1} - t_{i-1}}}{|\frac{\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})}{t_{i+1} - t_{i-1}}|} \tag{4.7}$$

As $t_{i+1} > t_i > t_{i-1}$:

$$\frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|} \approx \frac{\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})}{|\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})|} \tag{4.8}$$

So, finally, $\mathbf{T}(t)$ is computed as follows:

$$\mathbf{T}(t) = \frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|} \approx \frac{\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})}{|\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})|} \tag{4.9}$$

Similarly, for $\mathbf{N}(t)$:

$$\mathbf{N}(t) \parallel \mathbf{T}'(t) \tag{4.10}$$

Subject to $|\mathbf{N}(t)| = 1$. Consequently:

$$\mathbf{N}(t) \parallel \mathbf{T}'(t) \parallel \mathbf{r}''(t) \tag{4.11}$$

Approximated by the second order centered difference:

$$\frac{\delta_h^2[r](t)}{h^2} = \frac{\mathbf{r}(t+h) - 2\mathbf{r}(t) + \mathbf{r}(t-h)}{h^2} = \frac{\mathbf{r}(t_{i+1}) - 2\mathbf{r}(t_i) + \mathbf{r}(t_{i-1})}{(\Delta t)^2} \tag{4.12}$$

Following the same assumptions as before, and normalizing the lenght to 1, the result is:

$$\mathbf{N}(t) \approx \frac{\mathbf{r}(t_{i+1}) - 2\mathbf{r}(t_i) + \mathbf{r}(t_{i-1})}{|\mathbf{r}(t_{i+1}) - 2\mathbf{r}(t_i) + \mathbf{r}(t_{i-1})|} \tag{4.13}$$

Summarizing, the formulae to compute the Frenet trihedron are:

$$\begin{cases} \mathbf{T}(t) \approx \frac{\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})}{|\mathbf{r}(t_{i+1}) - \mathbf{r}(t_{i-1})|} \\ \mathbf{N}(t) \approx \frac{\mathbf{r}(t_{i+1}) - 2\mathbf{r}(t_i) + \mathbf{r}(t_{i-1})}{|\mathbf{r}(t_{i+1}) - 2\mathbf{r}(t_i) + \mathbf{r}(t_{i-1})|} \\ \mathbf{B}(t) = \mathbf{T}(t) \times \mathbf{N}(t) \end{cases} \tag{4.14}$$

where the vector $\mathbf{r}(t_i)$ is in our case a point of the path (x_i, y_i, z_i).

4.4.3 Application of the Frenet trihedron to 3D robot formation path planning

The advantage of using the Frenet trihedron is that among the infinite possible vectors perpendicular to the tangent vector, one is chosen in the direction of the curvature $\mathbf{N}$ (or normal acceleration). Furthermore, the direction of this vector changes continuously which is an important property when applying this trihedron as a reference for the geometry formation.

Then, by combining vectors $\mathbf{T}$, $\mathbf{N}$ and $\mathbf{B}$ any shape can be given to the formation, as depicted in Figure 4.17. The deformation of the shape depending on the obstacles of the environment is exactly the same as for the 2D method explained along the chapter.

In Figure 4.18 a motion sequence is shown in order to prove the feasibility of this method.

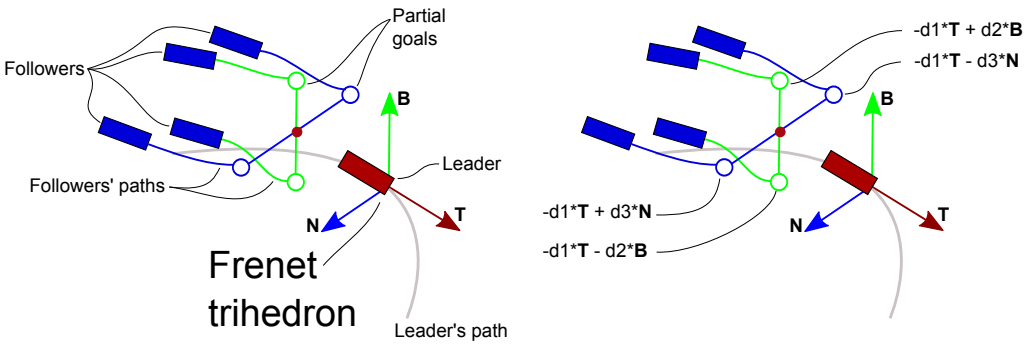


Figure 4.17: *Schema of the definition of the geometry in 3D based on the Frenet trihedron.*

4.5 Conclusions and future work

All the graphs included, except Figure 4.5, correspond to Matlab® implementations of the proposed algorithm, applied to different cases. It is important to note that the absolute times given for comparison are not representative, since the algorithms implemented in real robots would run much faster. However, the 50% time reduction is very remarkable because this would apply also to a real implementation. In the simulations, both the initial and the final points of the trajectory are given, and the paths are calculated with the FM^2 algorithm (both the leader's and followers' paths). To calculate the partial goals of the followers, a shape is previously set (e. g. a triangle, see Figure 4.1) defining the distances from the followers to the leader and modifying those distances as a function of the gray level of the current position.

The sequences shown in Figures 4.4, 4.8, 4.9, 4.10, 4.13 and 4.15 prove that the algorithm behaves well even in complex, non-regular, cluttered environments. It is also shown that many different formation shapes can be implemented, depending on the requirements of the specific application. These simulations were carried out in real scenarios acquired through sensors to show that the algorithm is robust to the environment modelling noise and irregularities. Moreover, Figures 4.18 proves that the algorithm can easily extended to 3D dimensions.

All these tests show that the proposed method, in combination with the FM^2

path planner, is robust enough to manage autonomous movements through an indoor environment, avoiding static and mobile obstacles successfully.

Moreover, the modifications to the algorithm to improve the behaviour of the formation or decrease its computational cost proposed in our previous work (Garrido et al., 2011) can also be applied in the method described here.

Results show that the proposed algorithm is able to manage uncertainties successfully with lower complexity than previous approeaches. In addition, this approach allows us to include any number of robots in the formations, by only setting the desired position with respect to the leader or the other robots. Therefore, this chapter introduces a novel approach to solve robot formation motion planning which is robust and fast enough to work under uncertainty conditions.

Future work in robot formation using FM^2 is related to improving the behaviour of the global formation during its movement, making it more autonomous when deciding how to move through the map in terms of flexibility. One interesting way to do this is to create a difficulty map which expresses, at each point of the initial map, the complexity that point represents to the formation (how much the formation has to change its shape, for example).

More complex fields can be studied, such as cooperative SLAM with formations, where the uncertainty is also present when sensing the environment.

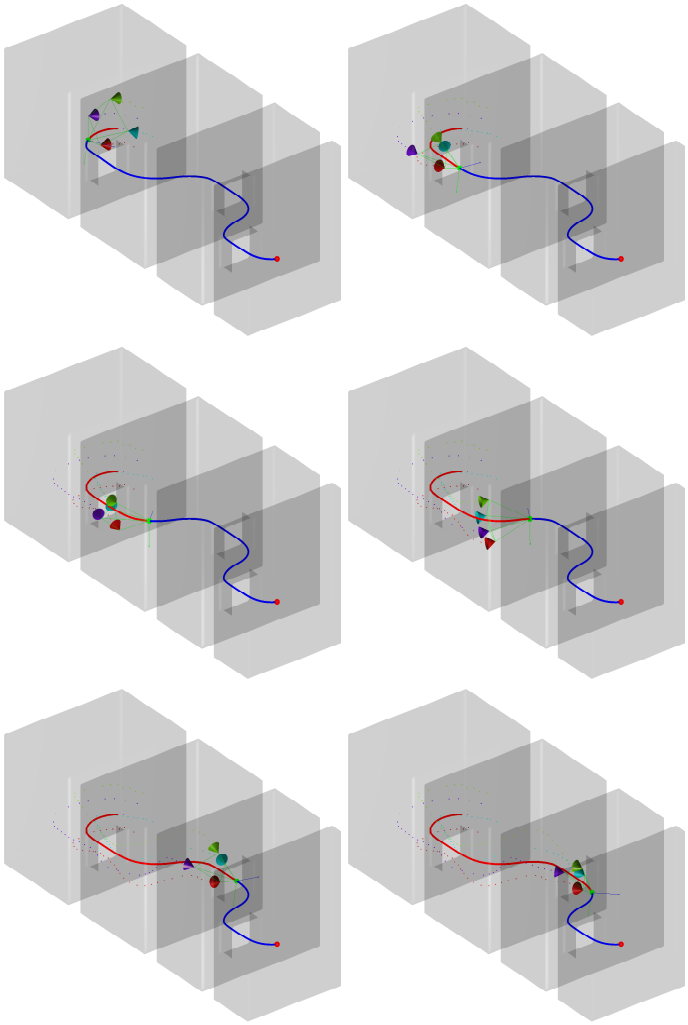


Figure 4.18: *Sequence of a pyramid-shaped robot formation (represented by cones) navigating in a 3D environment.*

Chapter 5

Adapting FM^2 Method to the Environment

5.1 Introduction

In this chapter, we get a fast path planning method based on FM^2. We present an algorithm which creates random paths for a given map using the FM^2 path planning method. The main idea of the proposed algorithm is to preprocess an already known map in order to get the *common* possible paths for that map, combining them when a new path is required. The key idea is that in a real robot application, the environment will be known (or obtained through SLAM). This environment can be dynamic but its main parts will remain, such as floor, doors, walls, and so on. Then the path planning method *adapts* to the environment creating some kind of road map which allows the FM^2 to spend the shortest possible time in path planning.

To achieve this goal a new concept is included, which we have called *FM^2 skeleton*: a set of random paths, obtained by means of FM^2, distributed throughout the map. This set creates a collision-free, smooth road map.

5.2 FM^2 skeleton

The objective is to create a skeleton which have *branches* in every zone of the map. The next points outline an algorithm to obtain a FM^2 skeleton:

- Model the environment as an occupancy grid map where the walls and obstacles are modeled with 0 (black) and the clear space with 1 (white).
- Distribute uniformly an n number of random points throughout the whole map. Erase those points which fall in zones where there are obstacles or walls.
- Include *characteristic* points of the environment. Those points where the robot commonly operates such as doors, light switches, furniture, among others.

Then, a loop begins whose steps are the following:

1. Select a point i.
2. Search a point j which Euclidean distance to the point i is higher than a value d_{min}. If no point is found, then select the farthest one.
3. Compute the path between the two points i,j according to the FM^2 method (specifically, we use the saturated version of the FM^2, at level sat).
4. Store the path with the other paths calculated in a binary map, called $\mathbf{W_p}$.

The loop ends when there are not any more couple points to obtain more paths. The resulting $\mathbf{W_p}$ can be considered as a FM^2 skeleton. Figure 5.1 includes a flowchart of the algorithm detailed in the previous lines.

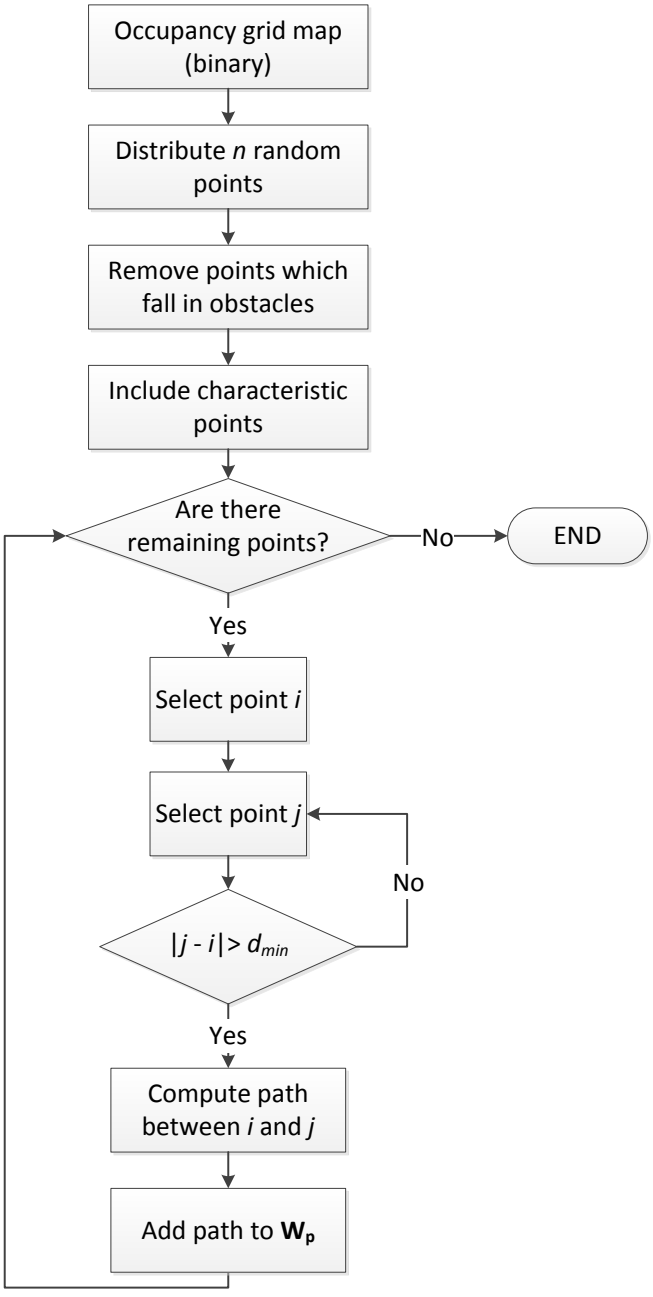
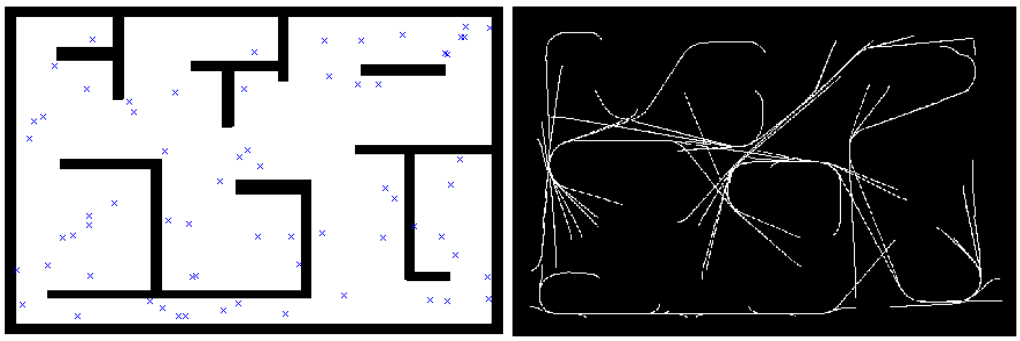


Figure 5.1: *Flowchart of the construction of the FM^2 skeleton.*

The figure 5.2 a) shows the random points distribution and 5.2 a) displays the skeleton obtained for those points. As one can see, all the rooms of the environment are connected with smooth branches.



(a) (b)

Figure 5.2: *a) Initial binary map with the set of n points randomly chosen but uniformly distributed. b)* $\mathbf{W_p}$ *map.*

5.3 Path planning over the FM^2 Skeleton

Once the FM Skeleton has been created, the next step is to compute the path between the desired points. The main function of this skeleton is just to route the wave expansion of the FM^2 algorithm in order to save computational time. Therefore, the algorithm is as follows:

- Dilate (with image processing techniques) the $\mathbf{W_p}$ map to obtain a thickened skeleton, ensuring the continuity of the skeleton.
- Compute the maximum between $\mathbf{W_p}$ and a value g_{min} close to 0 but always positive. This allows to expand the FMM wavefront out of the skeleton if necessary.
- The final map $\mathbf{W_{skeleton}}$ is calculated including the walls and obstacles (as black values) to the skeleton map $\mathbf{W_p}$.

These three steps, show in Figure 5.3 can be summarized in the following formula:

$$\mathbf{W_{skeleton}} = min\left(\mathbf{W_0}, max(g_{min}, \mathbf{W_p} \oplus \mathbf{SE})\right) \tag{5.1}$$

Figure 5.3: *Use of the FM^2 skeleton in path planning.*

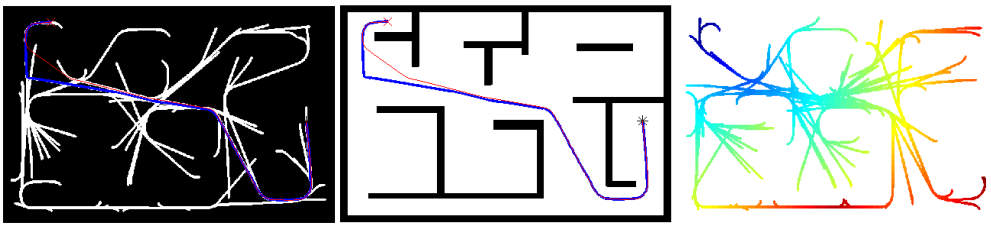
where $\mathbf{W_0}$ is the initial binary map and the symbol $\oplus$ represents the dilation operation with the structuring element $\mathbf{SE}$, which shape can be a disk or square (in 2D) and its size depends on how thick the skeleton is wanted to be (large $\mathbf{SE}$ means more area covered by the skeleton but less time reduction when planning).

Then, the result is a map $\mathbf{W_{skeleton}}$ in which the skeleton has the highest gray level (1) and the rest of the free space have a low gray value (g_{min}). The reason to do this is that the initial and final points are not restricted to fall into the FM^2 skeleton when searching a path. If the points are not in it, the robot will take the shortest path to return into the skeleton where the wave expansion is much faster due to its higher gray level in the $\mathbf{W_{skeleton}}$ map.

Finally, to obtain a path over the FM^2 skeleton it is enough to apply the base FMM, expanding a wave from the goal point until it reaches the initial point. This will provide a funnel-shaped potential which represents the time the wave takes to expand. The last step is then to apply gradient descent over this potential to obtain the path.

This method can be employed as an offline unsupervised learning algorithm, where the robot is able to *predict* which paths are going to be executed more often. Thus precalculating and merging those paths in a skeleton form can be interpreted as a training process where the path planning system evolves to adapt to the environment constrains.

Figure 5.4 shows the final performance of the proposed algorithm. Although the generation of the diagram is not deterministic, the algorithm appears to predict where are the main zones of the map (corridors between rooms) and the branches of the skeleton go into each room. The paths provided by our method could be sometimes a little worse (a bit longer, with not very smooth curves) but the time reduction could be worthy in most of the applications. For this particular case, when simulating in a 628x412 pixels map the time elapsed with the proposed algorithm is 0.16 seconds, while it took 0.40 seconds to the standard FM^2 method. The generation of the skeleton (using the parameters described in the next section) took 13.24 seconds. In the worst case, the proposed method will last at most the same time as the standard FM^2 when the required path is quite new.



(a) (b) (c)

Figure 5.4: *a) Comparison of the paths obtained with the saturated variation of the FM^2 method (in red) and the proposed method (in blue) shown over the thickened skeleton. b) Same comparison but shown over the initial map. c) Potential obtained when propagating the wave through other skeleton of the same map. Blue means getting closer to the global minimum and red means that the time is increasing.*

5.3.1 Setting the parameters

The proposed algorithm has a set of parameters which can change significantly the skeleton obtained. Following, how this parameters influence is described:

- *Number of points, n*: An equation has been experimentally obtained which

assures a balanced distribution of points (in 2D):

$$n = \alpha(\sqrt{x} + \sqrt{y}) \tag{5.2}$$

where x and y are the dimensions of the map (columns and rows respectively) and α is a factor manually chosen.

- *Minimum distance between points, d_{min}*: Our experiments point out that a good equation d_{min} can be found by:

$$d_{min} = \beta\sqrt{(x^2 + y^2)} \tag{5.3}$$

where β is another factor to be set manually.

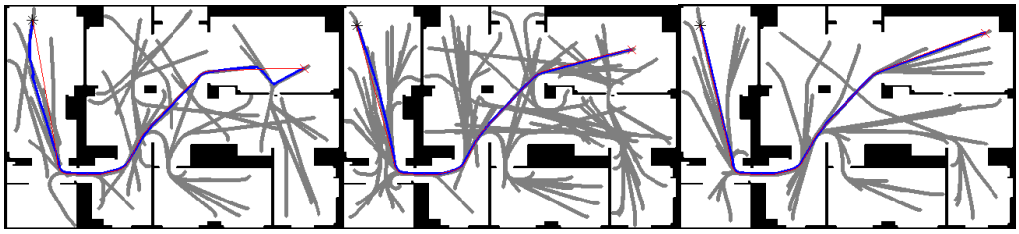
- *Minimum gray level, g_{min}*: Between 0 and 1.

- *Saturation gray level, sat*: Between 0 and 1.

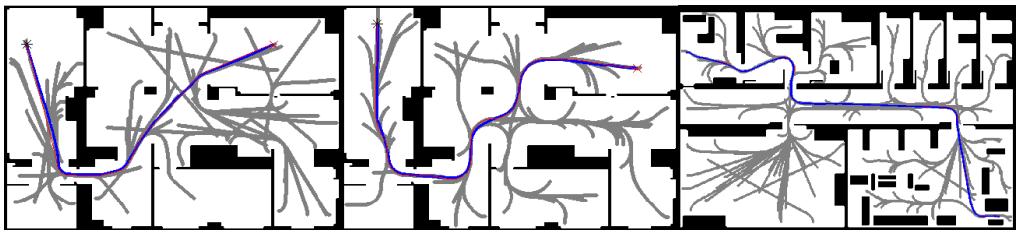
Table 5.1 shows the parameters configuration for many experiments, including a time comparison where t_s, t_{pp} and t_{FM^2} are the times elapsed when creating the initial skeleton, when planning over the skeleton and when planning with the standard FM^2 method, respectively. Also, figure 5.5 plots the results obtained in these experiments. The α and β values have been chosen experimentally with the objective of achieve enough points and with enough connectivity among them. Note that if α is too small the skeleton will have not enough branches to cover all the map. However, if it is set too large the skeleton will cover a wide area and the time reduction will not be that important. In the case of β, is a small value is chosen the skeleton will not have enough connectivity. Note that in the experiment (6) the environment is different. In this case the map is twice the size of the other experiments' map. The α has to change this is new map has more rooms, and with $\alpha = 2$ the probability of having a room without an skeleton branch is high.

Table 5.1: *Time results depending on the algorithm parameters.*

Test	α	n	β	d_{min} (grid cells)	g_{min}	sat	t_s (s)	t_{pp} (s)	t_{FM^2} (s)
1	2	76	0.3	158	0.001	0.3	3.34	0.10	0.20
2	**4**	**152**	0.3	158	0.001	0.3	6.82	0.11	0.20
3	2	76	**0.1**	**53**	0.001	0.3	5.82	0.07	0.20
4	2	76	0.3	158	**0.1**	0.3	3.94	0.21	0.21
5	2	76	0.3	158	0.001	**0.7**	4.21	0.09	0.20
6	4	188	0.3	250	0.001	0.3	23.24	0.17	0.46



(a) *Experiment 1* **(b)** *Experiment 2* **(c)** *Experiment 3*



(d) *Experiment 4* **(e)** *Experiment 5* **(f)** *Experiment 6*

Figure 5.5: *Skeleton depending on the different parameters. In most cases the path obtained using the FM^2 skeleton (blue) is very close to the one obtained with the standard FM^2 method (red).*

The results outline that the time reduction for 2D planning use to be around 50% (depending on the parameters) obtaining a path as smooth and safe as with the base FM^2 method. The time consuming by the skeleton generation depends mostly on the parameters given to the algorithm but this is not critical since this process can be executed offline.

5.4 Extension to d-dimensions

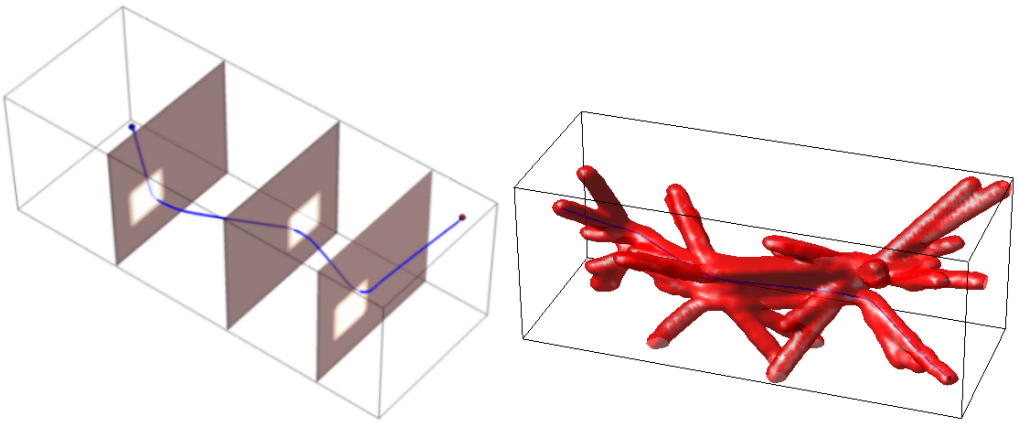
Many mobile robotics applications can be approximated by a 2-dimensional problem. But other ones simply cannot. The complexity of mobile manipulators or UAVs control does not accept bi-dimensional solutions. Therefore, most of the algorithms proposed which work very well in 2 dimensions often fail when increasing the dimensionality (due to the computational complexity).

Although it has been proved that the complexity of the FMM is $O(n)$ (Yatziv, Bartesaghi, & Sapiro, 2005), it suffers the same problem when expanding to more than 2 dimensions, since the computational cost increases exponentially with the dimension of the problem. The proposed method in this chapter assumes that the skeleton generation is done offline, and the path planning has to be done online. The fact of creating a skeleton by using FM^2 allows to search paths within the skeleton in d dimensions. Thus, the problem can be reduced to an *unidimensional* one, since the wavefront propagation is being guided by a *tube*.

To give a better intuition of this fact, we use the simile proposed by Greene to justify the string's theory (Greene, 1999): let us suppose an ant moving in a cable. This cable exists in 3 dimensions but the ant only can move forwards/backwards and clockwise/counterclockwise around the cable. The movement of the ant can be perfectly defined with only two values (dimensions). Moreover, if we put the ant inside the cable, it can only move forwards or backwards (because moving clockwise or counterclockwise changes nothing when it is inside the cable) so the movement in 3 dimensions is restricted to only one degree of freedom. Also, if we look at a cable from far away it appears to be unidimensional (a line).

If we apply this concept to path planning, the wavefront is expanded trough the d-dimensional skeleton the dimensionality can be considered as $d \simeq 1$. It is true that inside the tube the dimension is still d, but the *volume* of the tube is negligible in comparison with the volume of the rest of the space ($d = 3$) or hyperspace ($d > 3$).

The algorithm to employ is exactly the same as proposed in section 5.3. It is only necessary to adapt the parameters described in section 5.3.1. Figure 5.6 shows an example in 3 dimensions, where the proposed method took 0.22 seconds, while the time elapsed with the standard FM^2 method was 0.76 seconds. The skeleton was generated in 15.12 seconds.



(a) *3D initial map with FM^2 path.* **(b)** *The skeleton and also the path obtained.*

Figure 5.6: *Example of application of the algorithm in 3 dimensions.*

5.5 Conclusions and future work

This chapter presents a novel method for fast path planning without losing the safeness or smoothness in the obtained paths in comparison with the standard FM^2 path planning method. Also, the simulations carried out with Matlab ® show an important time reduction when calculating the path: 50% or even more.

It has been shown how the different parameters of the algorithm influence the skeleton and the time taken when calculating new paths.

Lastly, it has been proved that it is possible to extend the algorithm to more than 2 dimensions. Thus, the proposed algorithms is applicable not only in mobile robot applications but manipulation and UAVs among others.

The future work is focused on removing the stochasticity of the algorithm. One of the main ideas is to automatically obtain the points to generate the FM^2 skeleton from the map characteristics such, for example, the center of the different rooms. Another interesting work is to convert the skeleton creation to an online algorithm, allowing it to evolve and adapt to environment changes.

Chapter 6

Kinesthetic Teaching and Learning based on FM^2

6.1 Introduction

During the last years, robot configurations are becoming more and more complex, with a larger number of degrees of freedom (DOF) involved in order to get better and more natural movements. Thus, the control of the robot becomes challenging and the commonly used techniques are unuseful. This problem has been faced by means of learning techniques. The movements the robot should execute are *shown* to the robot in many different ways and the robot learns how it has to behave.

It is important to remark that in learning by imitation (also referred to as programming by demonstration) (Calinon, 2009), the demonstrations can be provided either by observing a demonstrator doing a task or by physical guiding of the robot during the task (kinesthetic teaching). While the first method requires the system to handle the re-targeting problem, the kinesthetic teaching method simplifies the problem using the same embodiment for both demonstration and reproduction.

In this chapter we propose a novel kinesthetic teaching method based on the FMM. The method assumes that the task taught to the robot can be *codified* into a path planning problem, either in joint coordinates or Cartesian coordinates. One of the main advantages of the proposed method is that it is very easy to implement and very intuitive, leaving aside complex theoretical formulation. The proposed method takes into account the environment, since it modifies the path planning algorithm of the system instead of modifying the motion control.

In the next section the learning algorithm is explained and its stability is analyzed. In section 6.3 the effect of the two parameters of the proposed approach is studied. Following, section 6.4 shows a experiment carried out in a real robot. Finally, section 6.5 outlines the application of the proposed method to more than 2 dimensions.

6.2 Kinesthetic Teaching and Learning Algorithm Based on FM^2

The FM^2 has proved to work efficiently in path planning tasks. The results of this method only depend on the environment conditions, such as obstacles, walls or other robots. This means that the method will always give the same results when working under the same environment, without taking into account previous experience.

The objective of the learning is to introduce new data to the FM^2 algorithm to improve the motion planning modifying the velocities potential W depending on what an expert shows to the robot. This will cause that the paths given by a modified W potential will present different characteristics from those given by the standard FM^2 method. Since only W is being modified, the good characteristics of the FM^2 method remain, such as smoothness and local-minima-free.

This chapter is focused on kinesthetic teaching, more precisely, guiding the robot (usually robotics manipulators or humanoid robot's arms) through the

desired trajectory. During the guidance, the robot records the data and later it adapts the parameters of the underlying mathematical model, commonly based on probabilistic formulas.

When being taught, the main objective of the robot is to be able to reproduce by itself the motion learned and adapt it to new motion requirements. Also, it is expected to improve the motion taught making it smoother, more efficient, faster, etc. Hence, the objective can be translated into learning a path and adapt it when necessary. Therefore the algorithm works over the path taught and over the path planning algorithm implemented. It does not mind if the teaching is being carried out in end-effector coordinates or joint coordinates.

6.2.1 Learning Algorithm

The proposed algorithm uses data gathered during a kinesthetic teaching process. During this learning, the end-effector's positions are stored with a time cycle T. The proposed algorithm can work as well with joint-coordinates, but to make it easier to understand the algorithm we use the end-effector's Cartesian coordinates.

The proposed algorithm starts with the velocities map of the environment $\mathbf{W}$ saturated at level sat, and the set of n points obtained during the kinesthetic guiding. The algorithm works as follows:

1. Connect all the points in the same order they were stored. This connection can be done using straight lines but we recommend to use FM^2 to take advantage of this method. Since this is considered to be done offline the computational cost is not very important. These connections are stored in a binary map $\mathbf{W_p}$.

2. Dilate $\mathbf{W_p}$ using a structuring element SE, whose size aoi defines the *area of influence* of the learned data.

3. FMM is applied to the $\mathbf{W_p}$, in order to convert it to a gray scale map. This map has to be rescaled to a maximum value of $(1 - sat)$.
4. Add the rescaled map $\mathbf{W_p}$ to the initial map $\mathbf{W}$.
5. Restore the obstacles and walls to value 0 because the previous steps could delete this information.
6. (Optional) Apply a smoothing filter in order to do not have harsh changes in the final velocities map $\mathbf{W}$.

By following these simple steps, shown in Figure 6.1, the velocities map is modified. The new paths provided by the path planning system will be very different depending on the experience of the robot. Figure 6.2 shows the different steps of the algorithm. In this case the path obtained is very similar to the one taught but much is smoother.

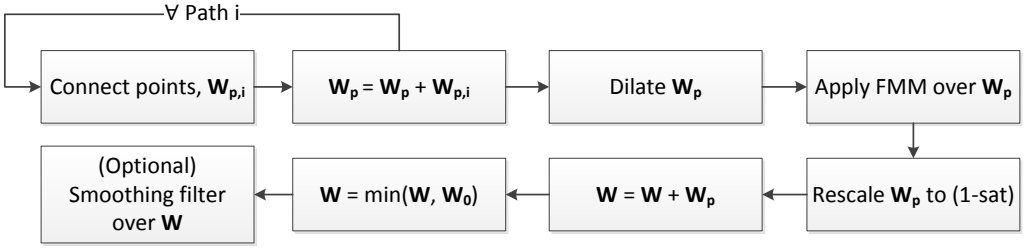
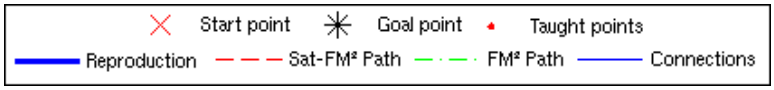
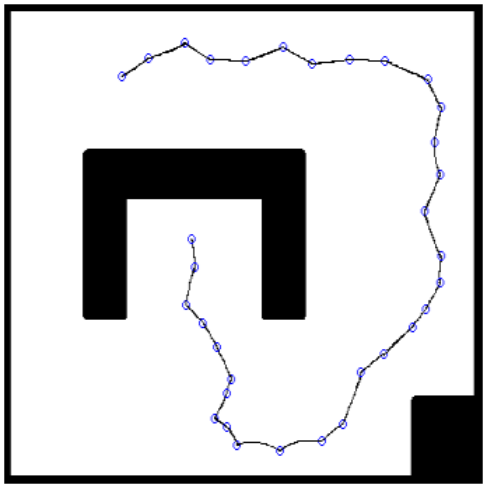


Figure 6.1: *Flowchart of the learning method based on the FMM.*

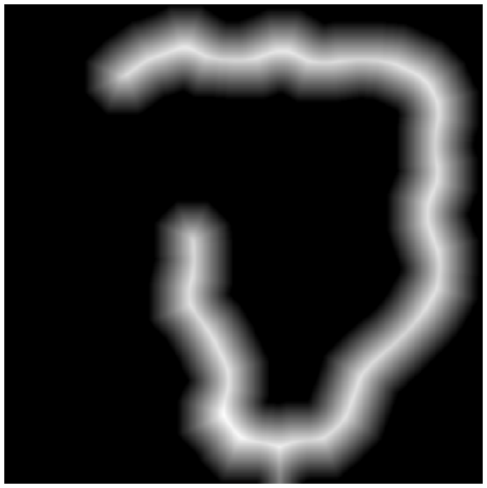
6.2.2 Stability Analysis

The motion of robots can be considered as a nonlinear autonomous dynamical system, where autonomous refers time-invariant. In this case, the proposed learning algorithm is asymptotically stable according to the Lyapunov Stability theorem (Slotine & Li, 1991). This theorem expresses that a function $\dot{x} = f(x)$ is asymptotically stable at the point p_g if a continuous and continuously differentiable Lyapunov function $V(x)$ can be found such that it is always positive, its derivative is always negative and $V(x_g) = \dot{V}(x_g) = 0$.



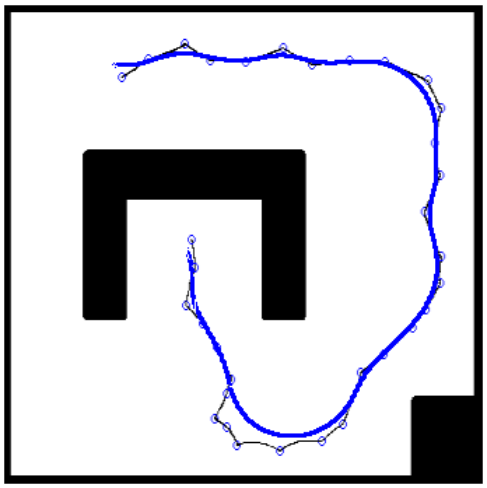


(a) $\mathbf{W_P}$ *first step.*



(b) *Dilated* $\mathbf{W_P}$.

(c) *Final* $\mathbf{W}$ *and different paths.*



(d) *Comparison between demonstration and reproduction.*

Figure 6.2: *Different steps of the learning algorithm.*

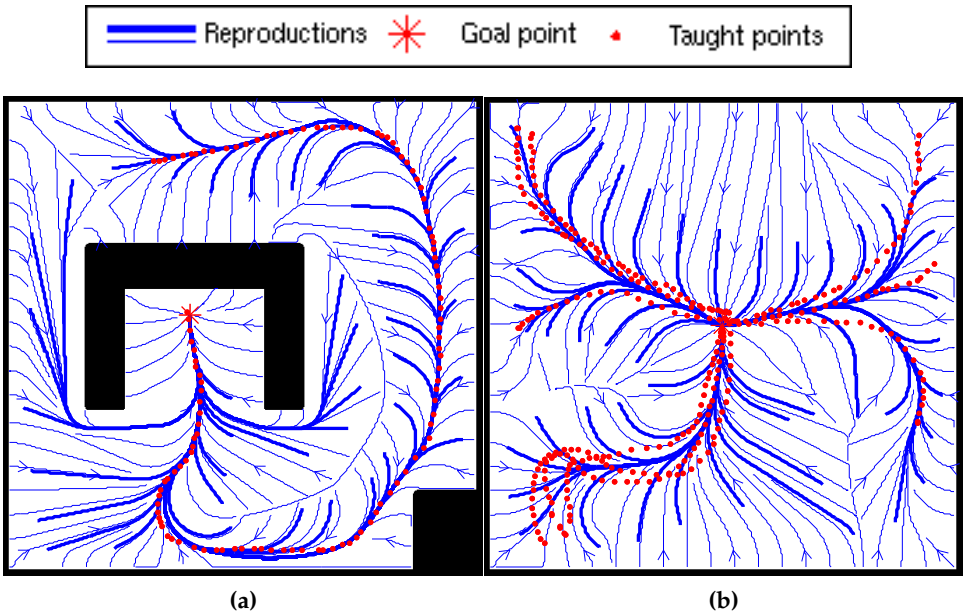
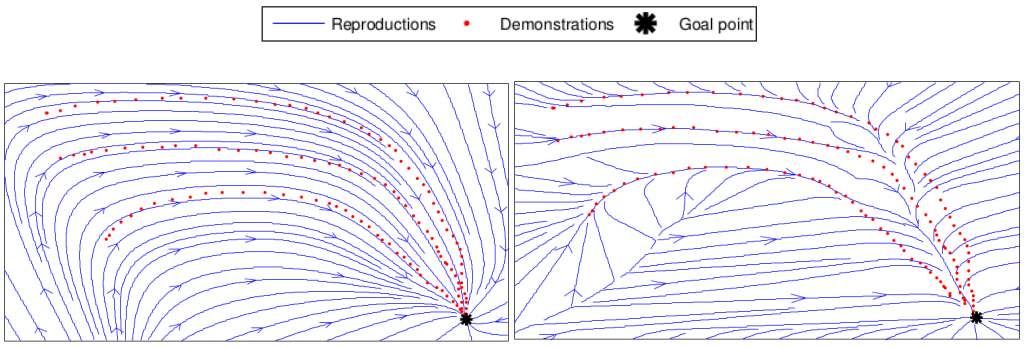


Figure 6.3: *Two examples of $D(\mathbf{x})$. a) $D(\mathbf{x})$ depends on the environment and also on the path taught. b) $D(x)$ depends only on the experience.*

Let us consider as Lyapunov function the one generated when expanding the second wave of FM^2, which we have called $D(\mathbf{x})$. This function starts at the goal point of the robot $\mathbf{p_g}$, where the $D(\mathbf{p_g})$ value is 0. Given the fact that this wave expands always with non-negative velocities, the value of $D(\mathbf{x})$ will be higher (positive) as the wave gets farther from $\mathbf{p_g}$. Finally, the derivative of the function is always negative since $D(\mathbf{x})$ is free of local minima.

These conditions will be always satisfied, regardless the environment or even the number and shape of the given paths during the learning process. Figure 6.3 serves as an illustration of the aforementioned. In 6.3 a) it can be seen how all the possible points will converge to the destination but with the tendency of following the learned data. In 6.3 b) all the paths will also converge to the destination regardless the initial point. However, since the experience is very different, the robot will follow different behaviors depending on the starting point.

This algorithm has been compared against a method called *Stable Estimator of*



(a) *SEDS* **(b)** *FML*

Figure 6.4: *Comparison between SEDS and FM Learning.*

Dynamical Systems (SEDS) (Khansari-Zadeh & Billard, 2011). In figure 6.4 a) the reproductions with SEDS are always similar to the taught data, this means that the streamlines are uniform. With the proposed algorithm (Figure 6.4 b)) these streamlines converge to the goal point as with SEDS. However, if the starting point is out of the area of influence of the taught trajectories, the experience will not be taken into account. Therefore, it is possible to set this behaviour tuning the aoi and sat parameters. This could be considered as a drawback of the algorithm depending on the application A further comparison among FM Learning and other methods and also a detailed study of the influence of the parameters is matter of future research.

6.3 Analysis of the parameters

The proposed learning method has two parameters: saturation level, sat, and area of influence, aoi. In this section we discuss how they affect to the results.

The fact of having two parameters to be set can be considered both an advantage and a drawback. The drawback is that tuning those two parameters can be a slow process and it could be difficult to find the correct combination to obtain the desired results. However, those two parameters allow us to set different behaviours during the learning process, obtaining different results for the same

taught trajectories.

The size of the aoi becomes very important in the performance of the motion. If this size is too small, the robot will just repeat the movement taught. On the other side, if the size is too big, the motion will differ a lot from the demonstrations. Figure 6.5 depicts this fact. Here, the robot has been taught 5 similar trajectories in a 500x500 pixels workspace with obstacles included. In the first case (figure 6.5 a) and b)) the aoi size is 10 pixels. The path obtained after learning is just the shortest path of those taught. However, in the second case (figure 6.5 c) and d)) the aoi size is 30 pixels. All the paths have turned into a unique learned, wide white zone in the velocities map. The path obtained in this case can be considered as a generalization of the taught paths. The aoi size also depends on whether we are carrying out one-shot learning or with multiple demonstrations.

From the reproductions field point of view, the aoi parameter influences on how the reproduced trajectories will follow the taught pattern. In Figure 6.6 a) the reproductions go directly to the taught trajectories and connect with them in a sharp way. As we increase the aoi value (Figures 6.6 b) and c)) this connections get smoother. Intuitively, the aoi represents the *liberty* of the reproductions against the taught trajectories. In other words, how similar we want the reproductions to be with respect to demonstrations. Higher aoi values mean smoother paths, so the similarity will be lower.

By modifying the sat level, the effect on the reproductions field is very different. Figure 6.7 show that in those places in which there are no experience (no trajectories taught), a low sat value (Figures 6.7 a) and c)) make the reproductions to tend to the closest place with experience. Therefore, the reproductions far from the demonstrations go perpendicular to the taught trajectories to reach them as soon as possible. However, once the sat value is increased (Figures 6.7 b) and d)), the reproductions tend to go to the goal point. Intuitively, the sat value means the importance of the taught trajectories against the rest of the

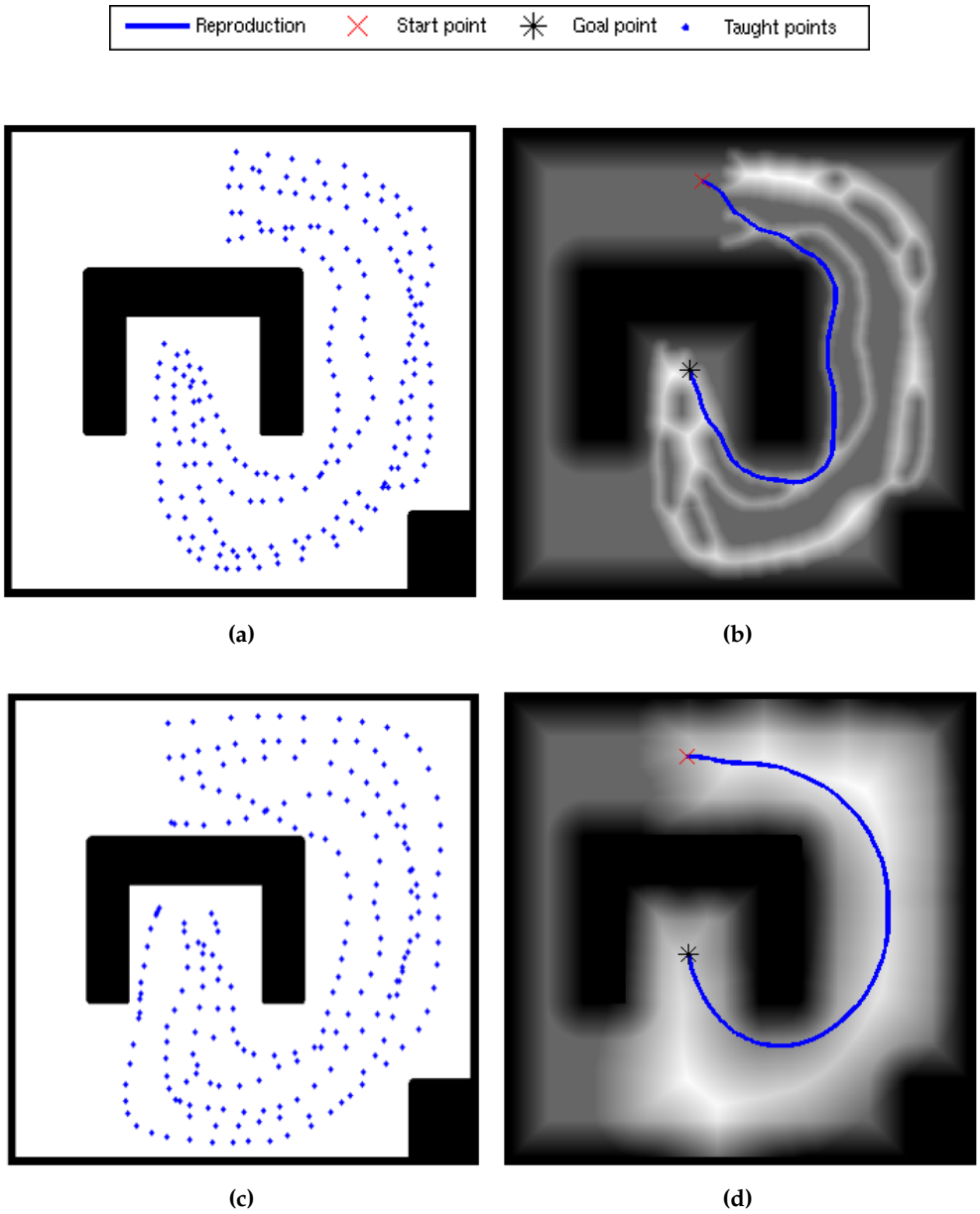
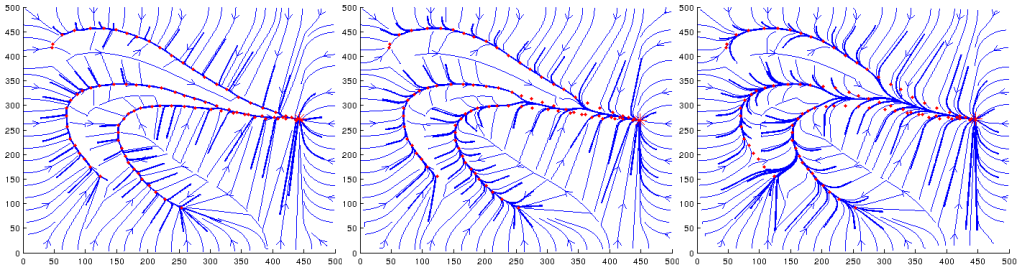


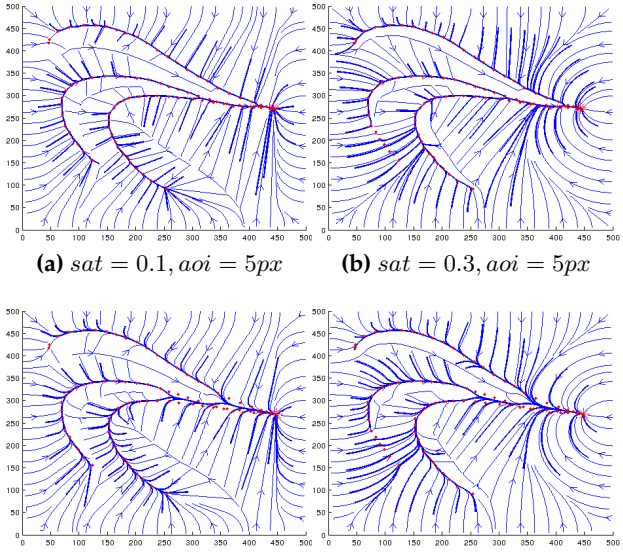
Figure 6.5: *Different paths using learned data depending on the aoi parameter in a 500x500 pixels map. a),b) $aoi = 10$ pixels. c),d) $aoi = 30$ pixels.*



(a) $sat = 0.1, aoi = 5px$ **(b)** $sat = 0.1, aoi = 15px$ **(c)** $sat = 0.1, aoi = 25px$

Figure 6.6: *Comparison among different aoi values for a given sat level in a 500x500 workspace.*

workspace. Higher sat values mean that the gray level of the zones without experience will be higher, so the trajectories will tend to go to the goal point more directly.



(a) $sat = 0.1, aoi = 5px$ **(b)** $sat = 0.3, aoi = 5px$

(c) $sat = 0.1, aoi = 15px$ **(d)** $sat = 0.3, aoi = 15px$

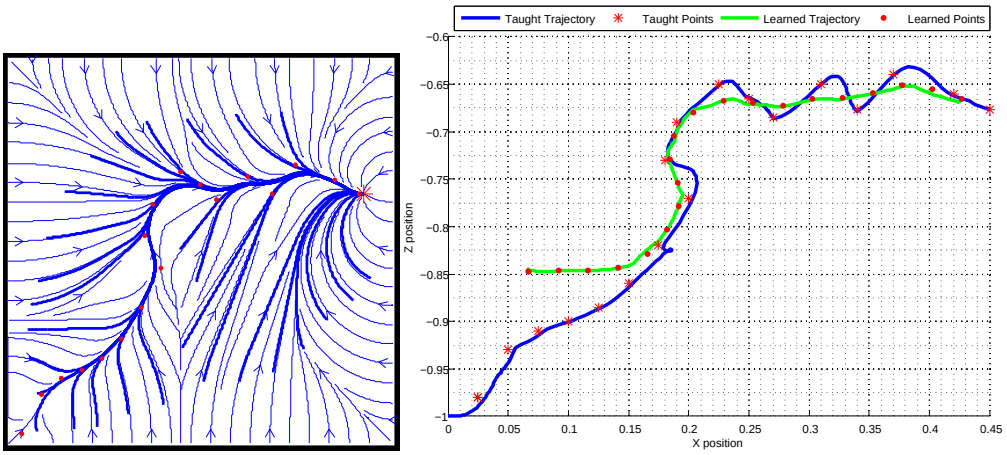
Figure 6.7: *Comparison among different sat level value for given aoi sizes in a 500x500 workspace.*

6.4 Experiments

To prove the feasibility of the proposed learning method, it has been implemented in the mobile manipulator Manfred V2. To gather the data, the arm is placed in different positions and the Cartesian coordinates of the end-effector are stored. After, the algorithm is run and the robot performs the learned trajectory.

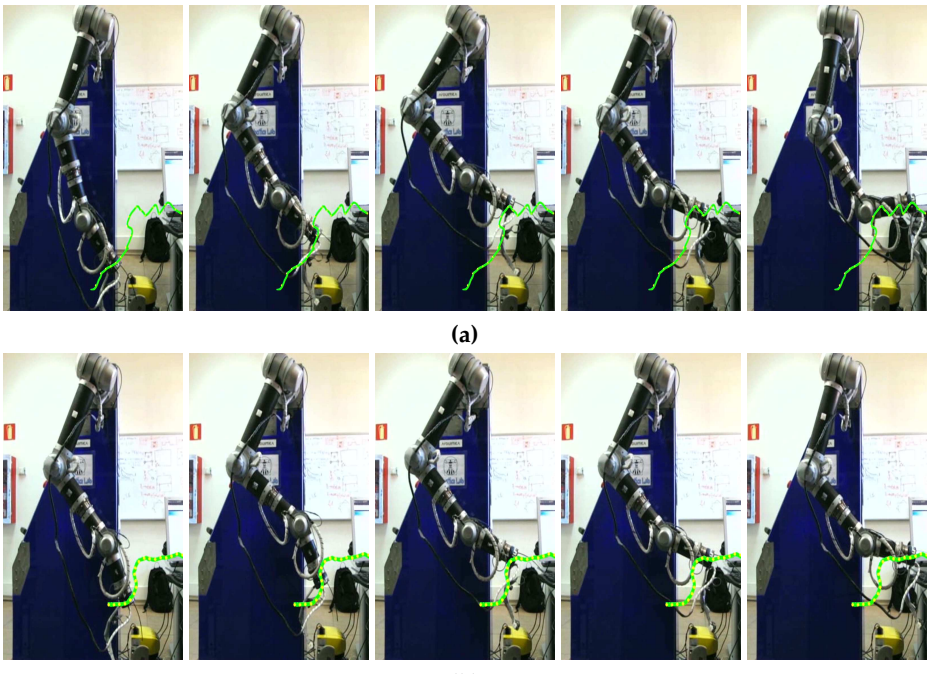
For better understanding, the arm is moved in a 2-dimensional plane and only the XZ coordinates of the end-effector are stored. One-shoot learning is carried about. This is, the robot is taught only once since we assume that this is a desirable point by robot's end users.

With the learned data, the $D(\mathbf{x})$ map (figure 6.8 a)) converges always to the goal point, independently of the starting point of the trajectory and resembling as much as possible to the learned trajectory. Figure 6.8 b) compares the two trajectories carried out by the robot (with the initial taught data and with the learned data), using $sat = 0.5$ and $aoi = 35$ pixels in a 500x500 pixels region (each pixel corresponds to 1 millimeter). It is possible to see how the second trajectory adapts to the learned one and improves it, developing an smoother trajectory. Finally, figure 6.9 shows the robot Manfred V2 developing both motion sequences.



(a) **(b)**

Figure 6.8: *a) $D(x)$ map obtained from Manfred. b) Comparison between the taught trajectory and the one reproduced once the robot has learned.*



(a)

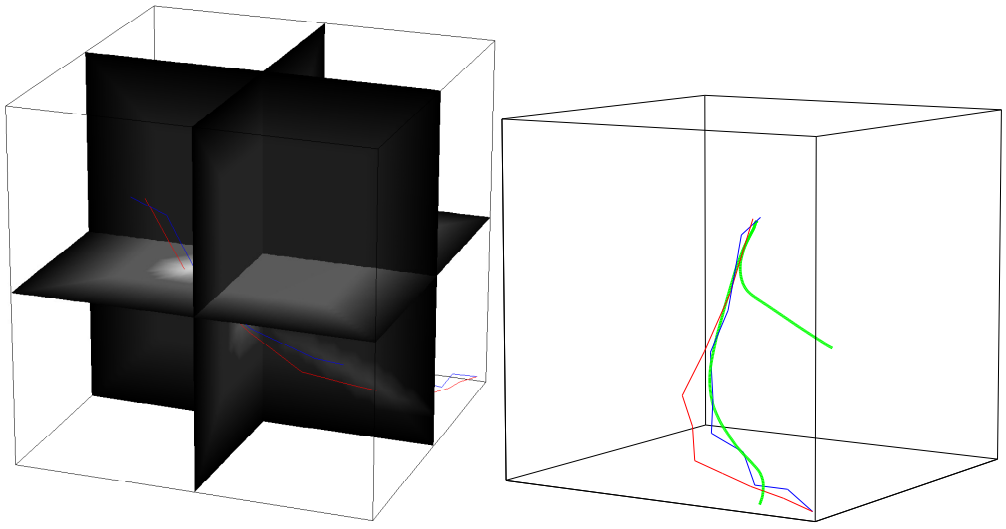
(b)

Figure 6.9: *The robot Manfred V2 developing the trajectories. a) Taught trajectory. b) Reproduction with FM Learning starting from a different point and with almost the same goal point.*

6.5 Extension to 3D

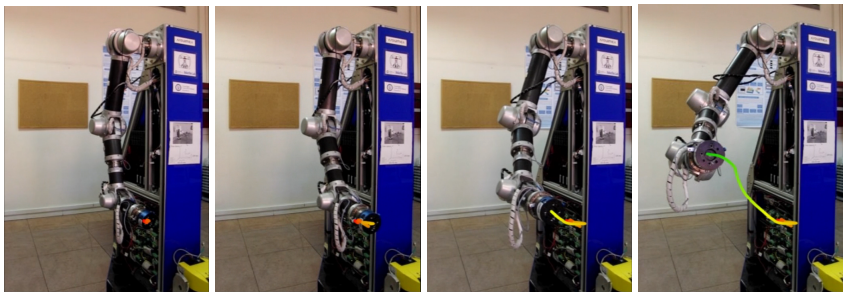
As FM, FM^2 and all the other steps of the proposed learning algorithm are defined for n dimensions, this section proves that the method also works for 3 dimensions without any modification. In our case, the third dimension is z, but it could be another one, such as roll, pitch, yaw if working on end-effector coordinates or joint angles if working in joint coordinates.

In Figure 6.10 a) two taught trajectories are shown together with the final **W** map they generated, using a 50x50x50 centimeters workspace, with a resolution of 1 cm per voxel, with parameters: $sat = 0.4$ level of and $aoi = 5$ centimeters. Figure 6.11 shows the robot Manfred V2 executing these trajectories. Additionally, Figure 6.10 b) includes two reproduced trajectories from different points of the work space. Both reproductions end in the same point (defined as the mean point of the taught trajectories) following the pattern shown to the system during the training process. In figure 6.12 Manfred V2 carry out these two reproductions.

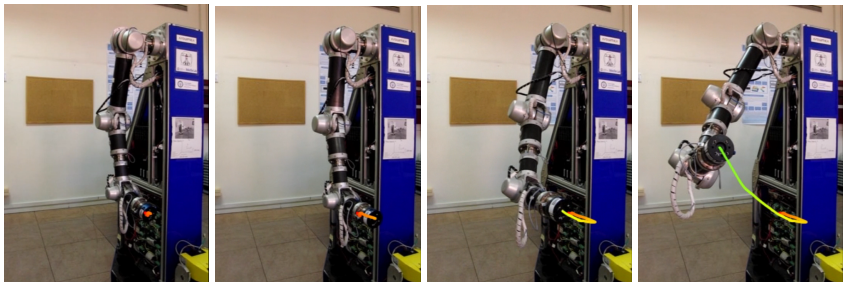


(a) **W** *map* **(b)** *Reproductions*

Figure 6.10: *Taught trajectories (blue and red) together with the map* **W** *and two reproductions.*



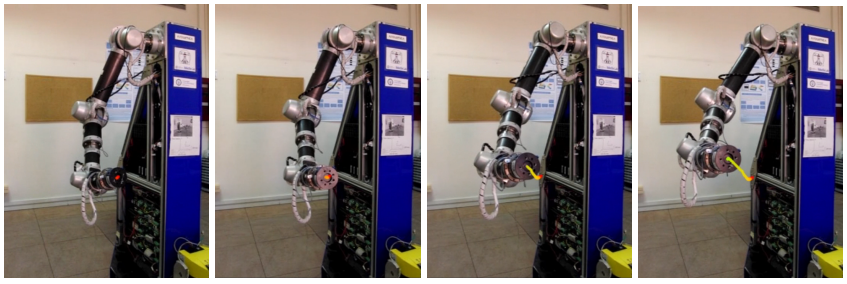
(a) *Taught trajectory 1, blue in Figure 6.10.*



(b) *Taught trajectory 2, red in Figure 6.10.*

Figure 6.11: *The robot Manfred V2 developing the taught trajectories in 3D.*

(a) *Reproduced trajectoy 1 of Figure 6.10.*



(b) *Reproduced trajectoy 2 of Figure 6.10.*

Figure 6.12: *The robot Manfred V2 developing the learned trajectories in 3D.*

It is worthy to mention that the aoi parameter has to be carefully chosen according to the dimensions employed. Since we used the spatial coordinates, we are assuming that the robot can perform in the same way in the three axes. However, if the workspace axes are multi-domain, i.e., spatial and joint coordinates, the aoi parameter has to be set accordingly to every dimension and the scale employed.

6.6 Conclusions and future work

This chapter presents a novel point of view for robot learning based on fast marching techniques. The proposed algorithm is not based on probabilistic approaches which can derive in local instabilities and non-convergences in case of an incomplete data set during the learning process.

The presented method erases the stochasticity of most of the learning methods, whose results vary depending on the sequence followed when learning. The kinesthetic teaching using FM^2 ensures global stability. Another important advantage is that it is very easy to include obstacles in the workspace.

In section 6.2.1 it has been discussed how the different parameters of the method change the behavior of the learning, being possible to set whether the robot will learn and follow very well the shown trajectories or if it is going to extract the main information of a set of paths, generalizing the information provided. The learning algorithm has been implemented in Manfred V2 to prove the feasibility of the system. The results are shown in section 6.4.

The chapter has focused on considering end-effector coordinates. To simplify, it has been supposed that such end-effector is moving in two dimensions. Nevertheless, the proposed algorithm, since it is based on FMM, can be applied to 3 or more dimensions, being possible to apply the algorithm to joint coordinates if desired.

This chapter proves that the FMM can be applied to approach learning solutions and hence it opens a new point of view in learning techniques. The FMM

is a very easy to implement and to understand technique. Therefore, the future work will focus on maturating FM Learning at the level of the current learning techniques, including among others: different velocities to the motions taught to the robot, a forgetting factor, deeper comparisons, and so on. Another very interesting case to study is the behaviour of the proposed method under perturbations or dynamic obstacles.

Chapter 7

Performance Study of FM^2 for Remote Handling Operations in the ITER Project

7.1 Introduction

The world's rising demand for energy is a key issue in the near future, since current energy sources are either finite (fossil fuels), or their output is insufficient to meet demand. It is in this context that the International Thermonuclear Experimental Reactor (ITER) project was born, which will act as an experimental facility to prove the feasibility of fusion power as an alternative, safe and reliable source of energy.

The reactor will be installed inside the Tokamak Building (TB) of ITER, as illustrated in Figure 7.1. The remote handling (RH) systems are required during maintenance operations that will play an important role in the ITER project, (Ribeiro et al., 2011). One of such systems is the Cask and Plug Remote Handling System (CPRHS), a mobile vehicle responsible for RH operations of transportation of contaminated components and equipment from the TB to the Hot

Cell Building (HCB). The largest CPRHS has dimensions 8.5m x 2.62m x 3.7m (length, width, height) and when fully loaded weights approximately 100 tons. The CPRHS is divided into three main components: the Cask, that contains the load; the Pallet, that supports the Cask; and the Cask Transfer System (CTS). The CTS acts as a mobile robot, by driving the entire vehicle, or by moving independently from the other components. The CTS has a rhombic kinematic configuration, as depicted in Figure 7.2. This configuration allows to control the velocity, V_i, and orientation, θ_i, of each wheel $i \in \{R, F\}$. Additionally the rhombic configuration provides the ability for both wheels to follow the same path, in this chapter referred as line guidance, or for each wheel to follow a different path, referred as free roaming providing flexibility when moving in the cluttered environments of ITER. In (Ribeiro, Lima, Aparício, & Ferreira, 1997) a comparison about different kinematic configuration is carried out. As shown in Figure 7.3, the rhombic configuration is the one which offers highest mobility and maneuver capability, minimizing the spanned area with the main drawback that the control is more complex.

In Figure 7.1 a snapshot of the Trajectory Evaluator and Simulator program is shown. This will be detailed in the following chapter.

The nature of the ITER project requires robust path planning and motion algorithms. Beyond the obstacle avoidance, the cluttered environment and the large dimensions of the CPRHS convert this typical navigation problem in which the typical solutions are not enough and new, improved path planning algorithms are required.

In this context, this chapter studies the application of the FMM to the path planning problem in the ITER scenarios. We will focus on the line guidance approach, since it is enough to solve most of the situations that could be given during ITER operation.

Although the ITER scenarios are 2D, the structure of the building and the dimensions of the CPRHS turn this problem into a very hard one. Also, the ITER

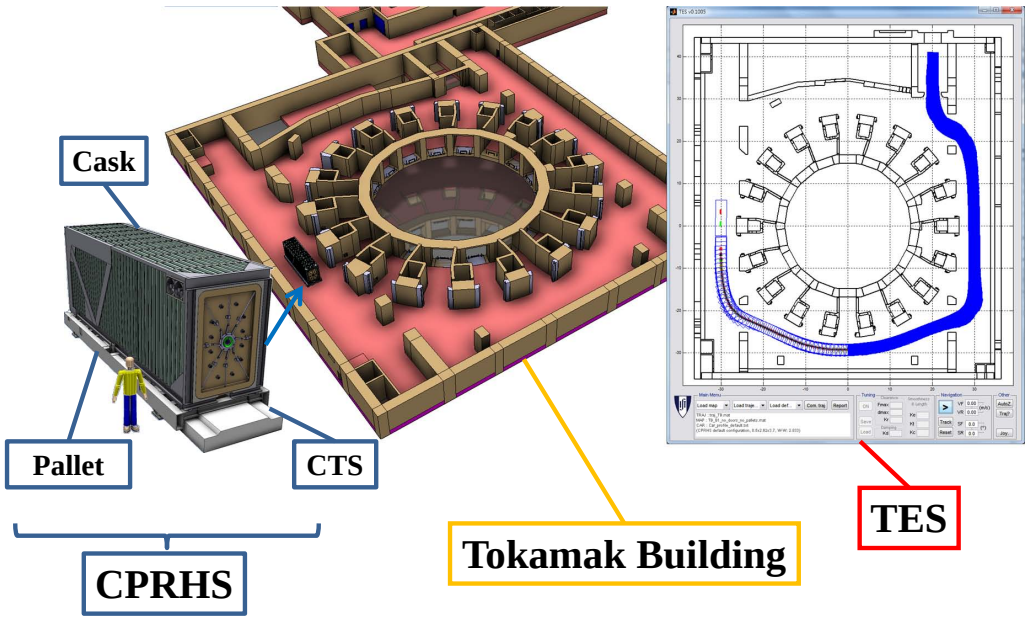


Figure 7.1: *Left - CAD model of the CPRHS. Middle - Level B1 of the TB with the CPRHS in operation. Right - Snapshot of the software application TES to evaluate the trajectory optimization.*

safety and operating requirements are quite strong. Hence, classical path planning methods need to be improved in order to accomplish all the restrictions.

Concretely, the ITER requirements focus on safety and smoothness. The large dimensions of the CPRHS, the cluttered environment and the contaminated nature of its load turn the motion planning problem into a very complex one, demanding a robust planner. The trajectories must be also the shortest in time while satisfying energy optimization for the CPRHS. Another very important requirement is that the minimum safety distance to obstacles to be guaranteed is 300 mm (Fonte, Valente, Vale, & Ribeiro, 2010; Valente, Vale, Fonte, & Ribeiro, 2011). Since FM^2 considers that the robot is a point, without kinematic constrains, it is mandatory to use a collision detector algorithm to study the reliability of the trajectories given.

The FM^2 method could be directly applied to the several scenarios of ITER. However, the study of FM^2 on ITER will focus on the level B1 of the TB, Figure

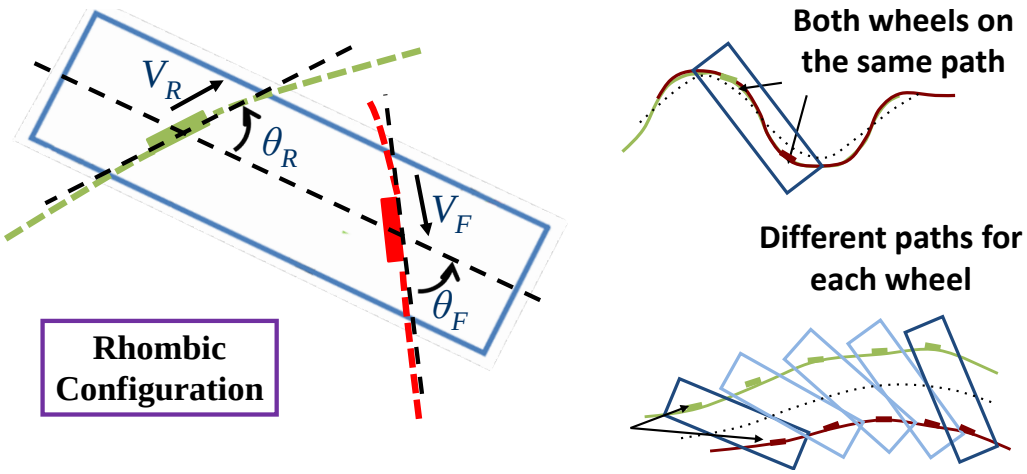
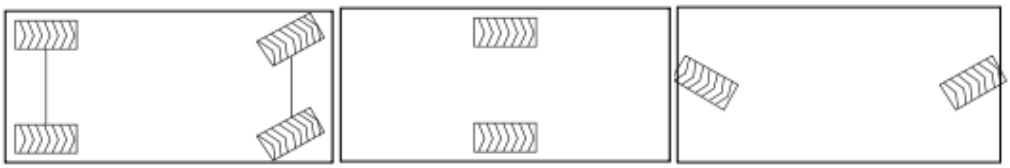


Figure 7.2: *Left - Model of the rhombic vehicle and the Cask Transfer System control variables. Right - Possible motion options for a rhombic like vehicle.*

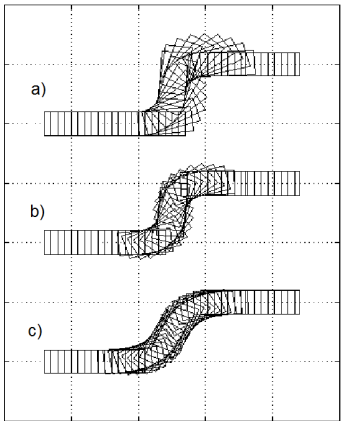
7.4 because this is the most building part of the ITER installations. Due to the radiation, this is the places where errors are most difficult to solve so the deepest analysis is required here.

In Figure 7.5 it is shown that the FM^2 provides smooth and reliable paths as usual for the level B1 of the TB. However, the dimensions of the vehicle and its complex kinematics lead to a deeper study of the reliability of the FM^2 in such environment.

Next section will focus on how this collision detector is implemented leveraging the characteristics of the FM^2 method. The reliability of FM^2 on ITER environments is analyzed in section 7.3. Lastly, in section 7.4 a 3-dimensional path planning approach is studied.



(a) *Cart* **(b)** *Differential* **(c)** *Rhombic*



(d) *Area spanned*

Figure 7.3: *Comparison of the different kinematic configurations.*

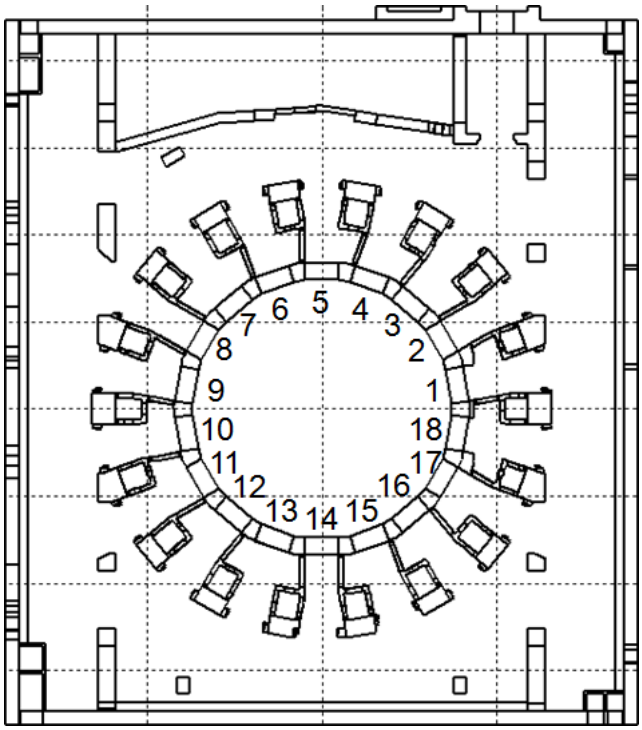


Figure 7.4: *Level B1 of the TB and its corresponding port numbers.*

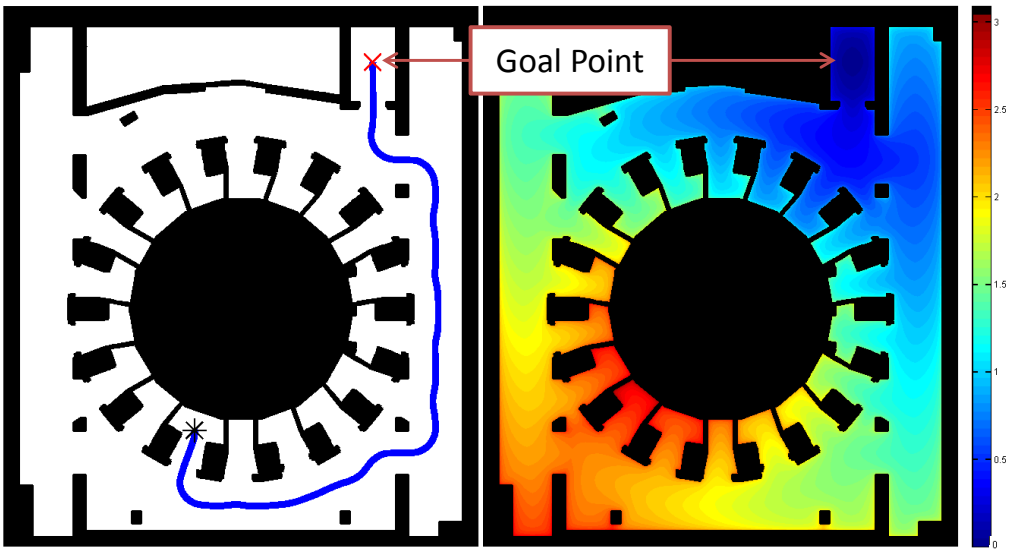
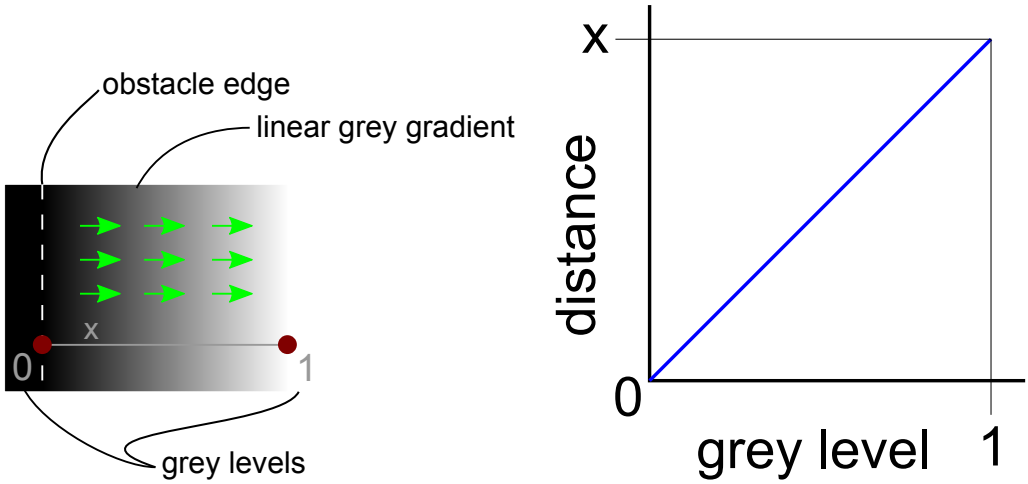


Figure 7.5: *Level B1 of the TB and its corresponding port numbers.*

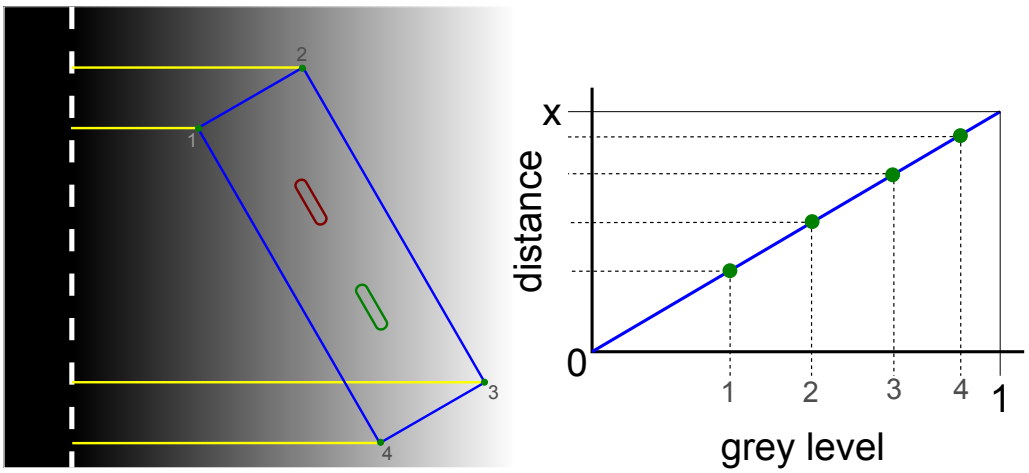
7.2 Collision detector based on FM^2

Thanks to the fact that the velocities map has a value for every point directly proportional to the distance to the closest obstacle, it can be easily translated into a distances map. The most important point is that, calculating the gradient in each point of this map, the direction to the closest obstacle is also obtained. Since these operations are computationally expensive, it is important to stress that these maps should be calculated offline and once per map (a *gradients map* can be previously computed). These concepts are illustrated in Figures 7.6 and 7.7. In Figure 7.8 it is detailed how these gradients are in the level B1 of the TB.



(a) *Grey gradients* **(b)** *Conversion grey-distance*

Figure 7.6: *Use of the velocities map as a collision checker.*



(a) *Distances on the CPRHS* **(b)** *Conversion grey-distance in the corners*

Figure 7.7: *Application of the collision checker to the CPRHS.*

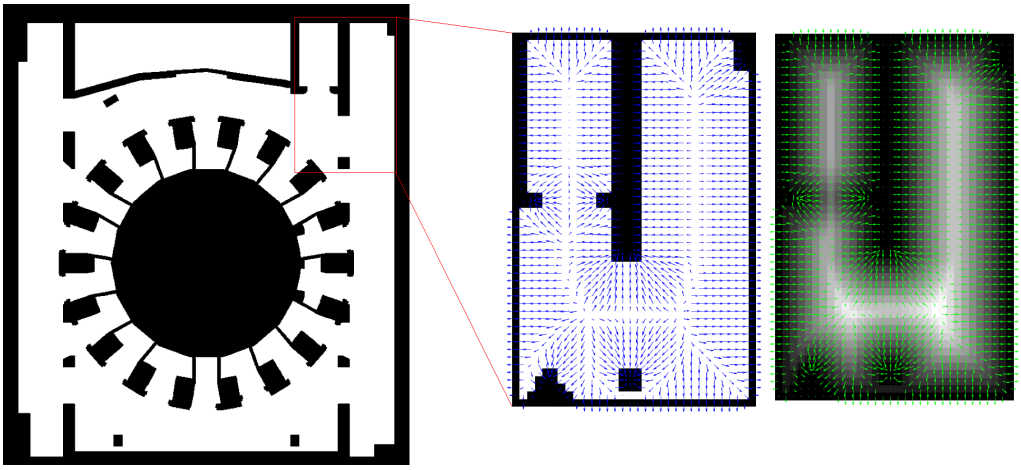
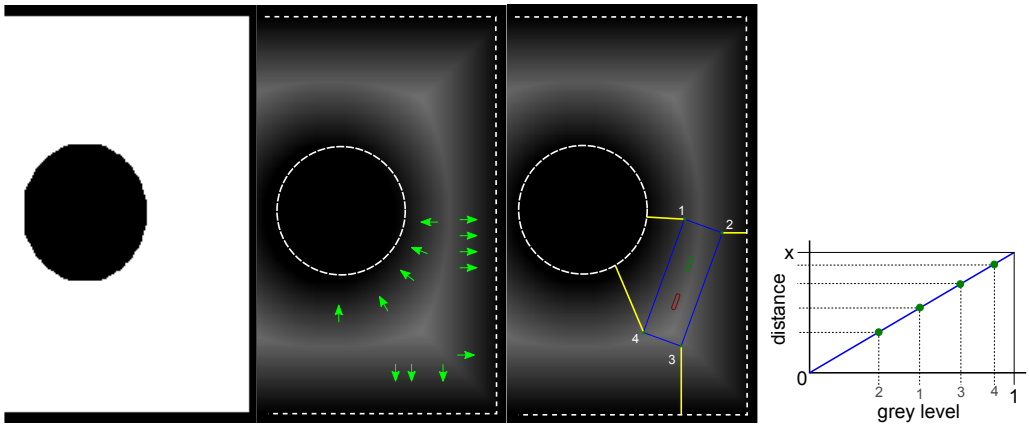


Figure 7.8: *Gradients in the level B1 of the TB.*

Therefore, to compute the distance to the closets obstacle (and also where it is), a set of as many as desired points are distributed along the perimeter of the CPRHS. The gray level of all these points is checked and the lowest one is chosen. Thus, it is already possible to obtain the distance to the closest obstacle and also its direction, according to Figure 7.9.



(a) *Binary map* **(b)** *Gradients of the velocities map* **(c)** *Distances to CPRHS in the corners* **(d)** *Conversion grey-distance*

Figure 7.9: *Application of the collision checker to the CPRHS.*

The results of the application of this algorithm with real ITER building map and CPRHS's dimensions are shown in the next section.

7.2.1 Collision detector validation and simulation

In order to validate the proposed collision detection algorithm, simulation tests are carried out. In these, both wheels of the CPRHS have to follow the computed path. Also, the velocity command applied to the CPRHS is according to the position of the front wheel of the CPRHS (green) within the velocities map (thanks to the FM^2 characteristics).

For the collision detection, one point every centimeter is places around the CPRHS's perimeter (a total of 2224 points). Also, the map dimensions are 1648x1450

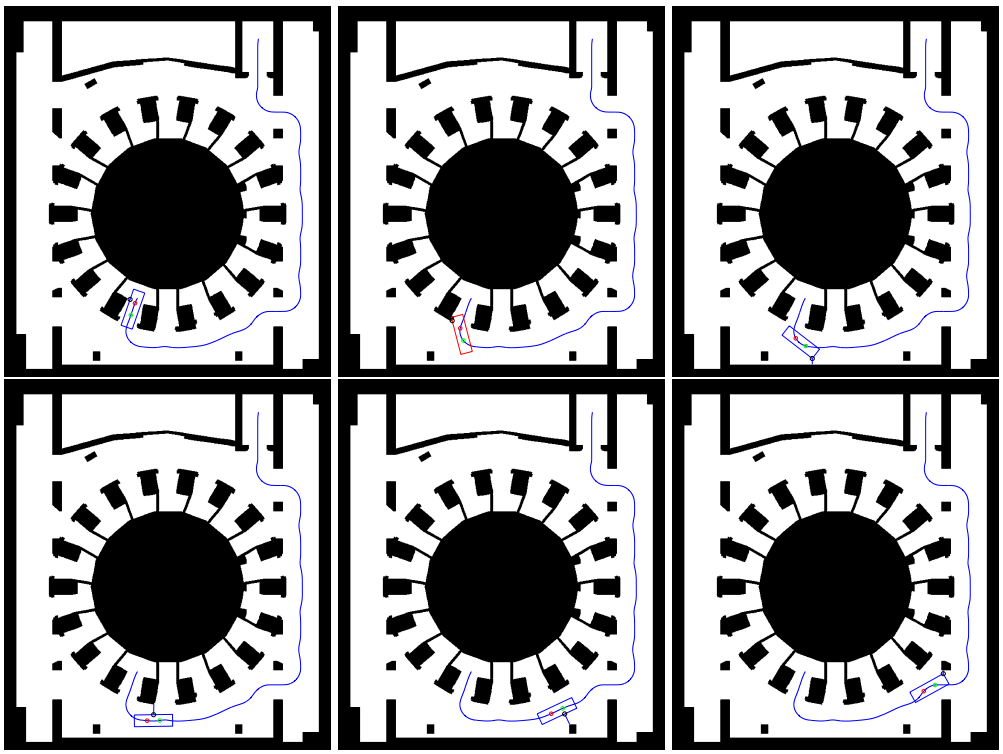
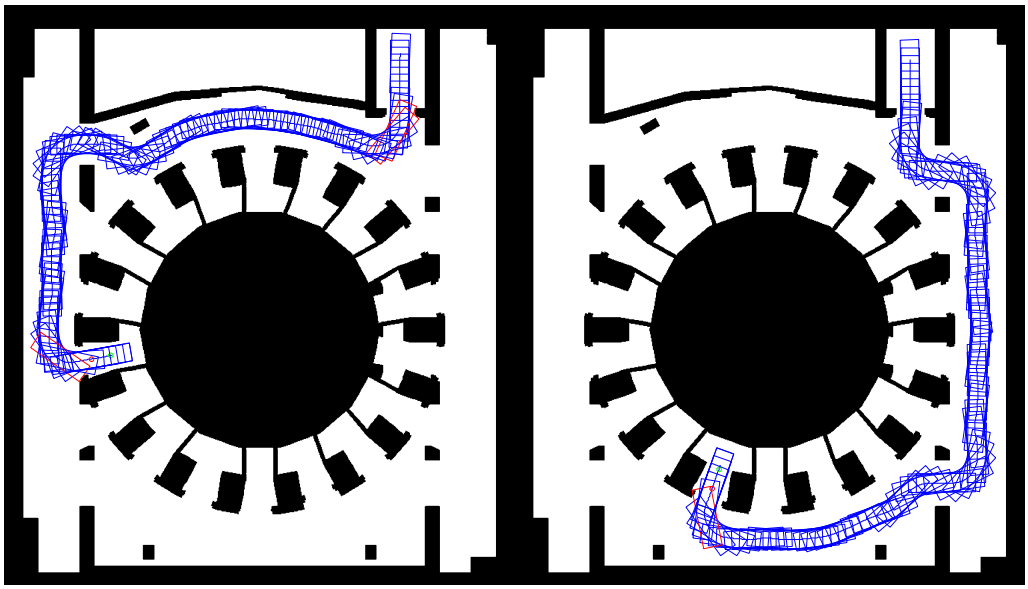


Figure 7.10: *Sequence of the CPRHS navigating and using the proposed collision detector.*

pixels, which mean a grid size of 5 centimeters. All paths were split in 100 iterations. In Figure 7.10 this collision detector is shown working, detecting the closest obstacle to the vehicle, its direction and also the distance. With this tool, it is possible to analyze the performance of the FM^2 in the ITER environments.

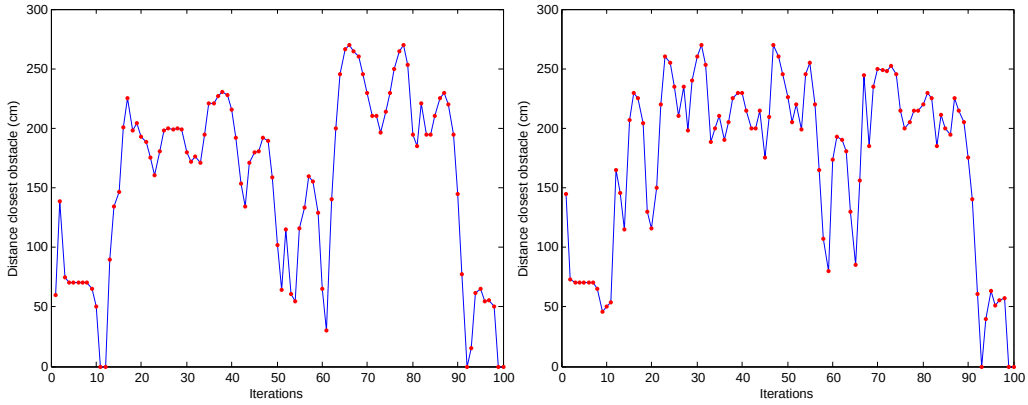
7.3 Performance of the FM^2 method in the ITER environment

Among the 18 possible trajectories, the simulation of those two which could be more representative and meaningful (port cells 10 and 13) are shown in Figures 7.11, 7.12 and 7.13, including also the distances to the closest obstacle and their velocity profiles of all iterations.



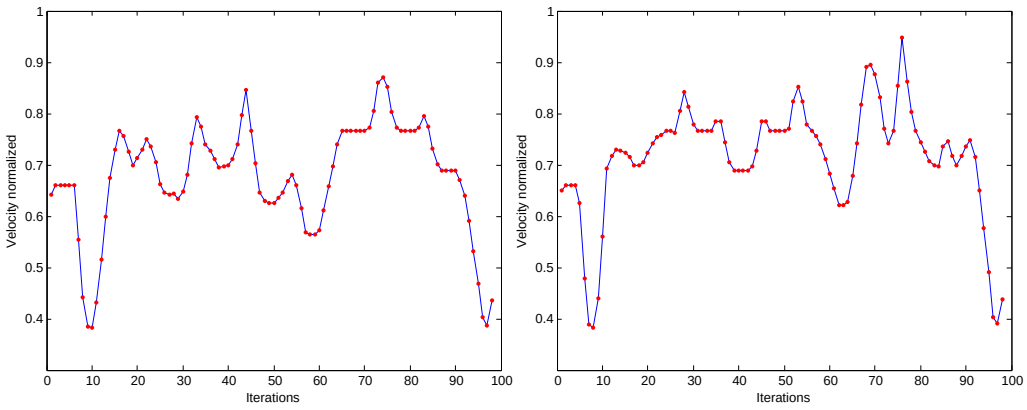
(a) *Port 10* **(b)** *Port 13*

Figure 7.11: *Path and CPRHS' poses from the lift to the ports. Red CPRHS means collision.*



(a) *Port 10* **(b)** *Port 13*

Figure 7.12: *Minimum distances during the motion.*



(a) *Port 10* **(b)** *Port 13*

Figure 7.13: *Velocities profiles.*

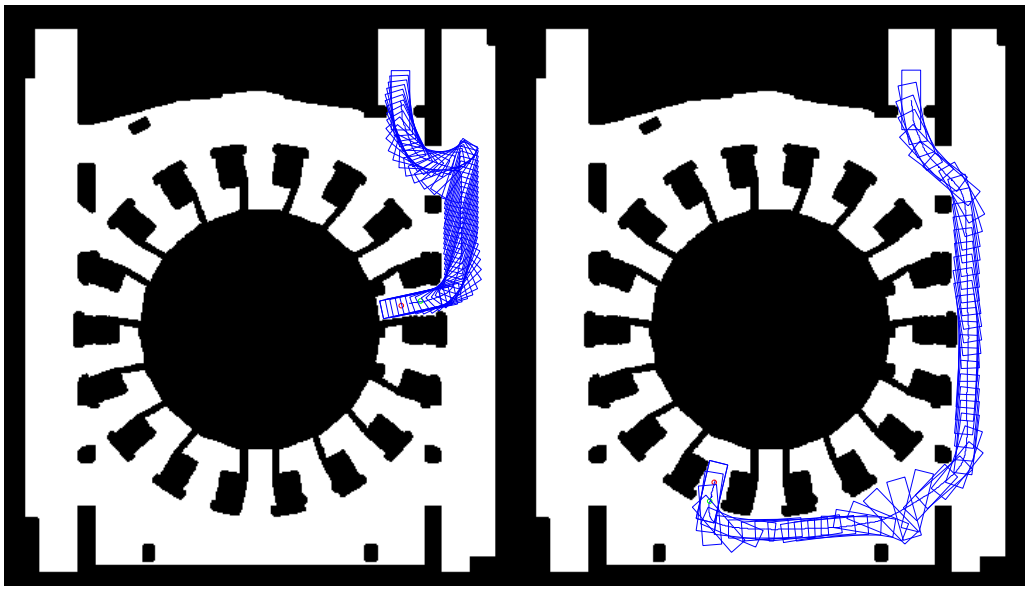
Simulations show that the FM^2 method, with both wheels of the CPRHS following the path, has collision points in the lift and when entering the port cells. This is due the fact that FMM and FM^2 does not take into account the kinematics of the vehicle when planning. However, thanks to the high mobility of the CPRHS and the FM^2 collision checking algorithm, the best way to avoid these collisions is to detect the collision points and allow the wheels to move outside the path prior the collision occurs. Therefore, it is mandatory to include a new layer in the path planning algorithm to remove these conflict points, leaving the rest of the path as is.

As additional information, the computation time of the trajectories use to be approximately 3.5 seconds (using Matlab® in an Intel Core 2 Duo 3 GHz PC). Furthermore, the time it takes to check collisions is between 20 and 40 milliseconds.

7.4 Planning in 3 dimensions with FM^2 in ITER environments

With the collision detection algorithm proposed above, it is possible to create a 3D C-space of the environment, with the two dimensions of the CPRHS's position and the vehicle's orientation as the third dimension. Computing a trajectory along the C-Space built taking into account the vehicle's dimensions, it is possible to guarantee the absence of collisions.

Since the FM^2 is based on the FMM, it can be applied to more than 2 dimensions easily. The C-space has been built iteratively placing the CPRHS in every position and with every possible angle. This is a slow task, but it can be done offline and once per map. Results are shown in Figure 7.14, in which maneuvers where forced to prove the collisions absence.



(a) *Port 1* **(b)** *Port 13*

Figure 7.14: *Paths computed in 3D, using the orientation as third dimension.*

It is worthy to mention that as the third dimension included in these experiments are the orientation of the vehicle, the interpretation of the gray level for the velocity profile and also the optimality in terms of time changes. For instance, the CPRHS, when moving in a straight line, has an orientation which is not parallel to the trajectory (this orientation minimizes the area spanned by the vehicle). Therefore, the third dimension, as it is not a spatial dimension, adds a behaviour that is not desirable for this application.

7.5 Conclusions

Throughout this chapter the FM^2 path planning method has been outlined and applied to the ITER-RH problem. Results show that a standalone FM^2 algorithm is not enough to satisfy the ITER requirements: there are collisions in very concrete points of the scenario. However, the path quality (in terms of smoothness and safety) is high in the rest of the points.

A variation of the standard FM^2 method has been proposed, planning 3-dimensional trajectories. These trajectories are collision-absent but the space spanned by the CPRHS could become a problem. This issue could be overcame by modifying the velocities map, but a deeper analysis is required.

The application of FM^2 to the ITER has still many challenges: the grid map discretization can become a problem is more resolution is required, making the algorithm slower. Even more, although the FM^2 is safe enough, the dimensions of the CPRHS provokes some collisions, which is the main point to be solved.

The next chapter presents a novel way of using FM^2 in the ITER which solves the problem presented here.

Chapter 8

Integration of FM^2 in the TES Program of the ITER Project

8.1 Introduction

The Trajectory Evaluator and Simulator (TES) shown in Figure 8.1 is a software application developed under the grants F4E-2008-GRT-016 and F4E-GRT-276-01 (European Joint Undertaking for ITER and the Development of Fusion Energy Work Programme 2008 and 2010) related with the optimization of trajectories for the CPRHS in TB and HCB.

The TES was developed under MATLAB® environment. The main goal of this application is to generate optimized trajectories between an initial and a final pose (a pose is defined by a position and orientation) for a vehicle with a rectangular 2D projection and rhombic kinematics, in a general scenario. In particular, this software has been applied to the CPRHS, the CTS and rescue casks in the scenarios of the ITER, e.g., between the lift and a Vacuum Vessel (VV) port cell in TB and the HCB. Besides the optimized trajectory generation, this application provides, among others, the following additional features (Vale, Valente, Fonte, Ribeiro, & Ferreiro, 2012):

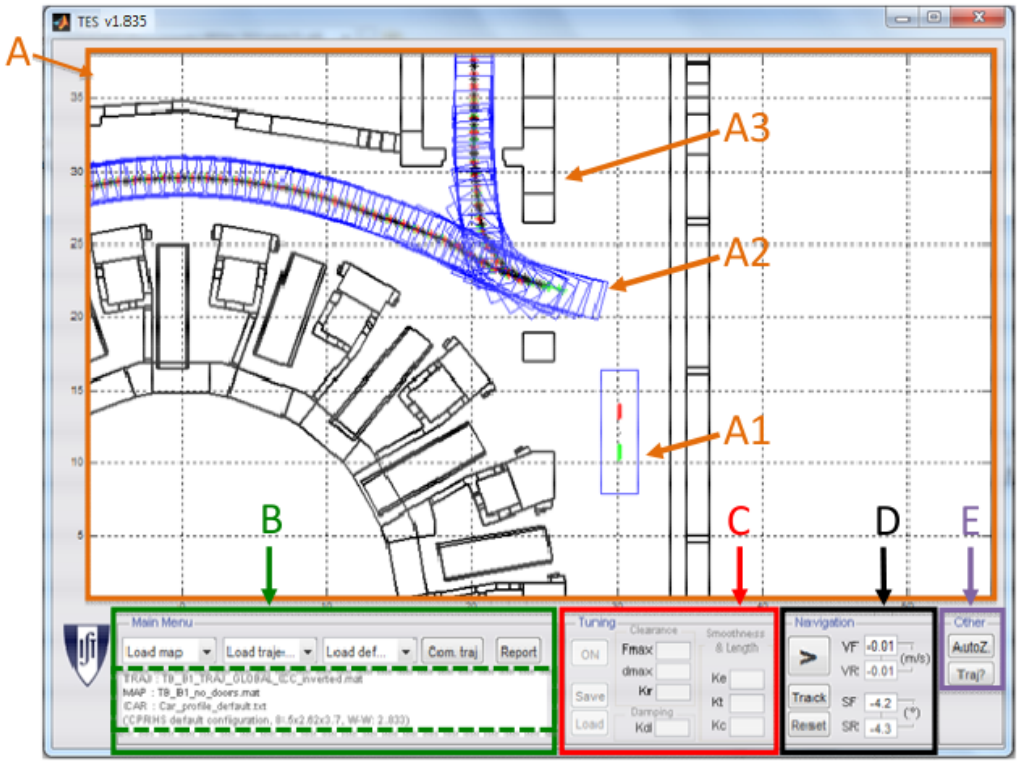


Figure 8.1: *A - Simulation window (A1 vehicle, A2 trajectory, A3 maps. B - Main menu. C - Tuning panel. D - Navigation panel. E - Tools panel.*

- Compute, load and save trajectories.
- Simulate vehicle following a path.
- Identify the most critical points of a trajectory, i.e., the most critical positions, where the vehicle is closer to the obstacles.
- Generate reports and plots with the history of velocities and minimum distances to obstacles along the optimized paths, as shown in Figure 8.2.

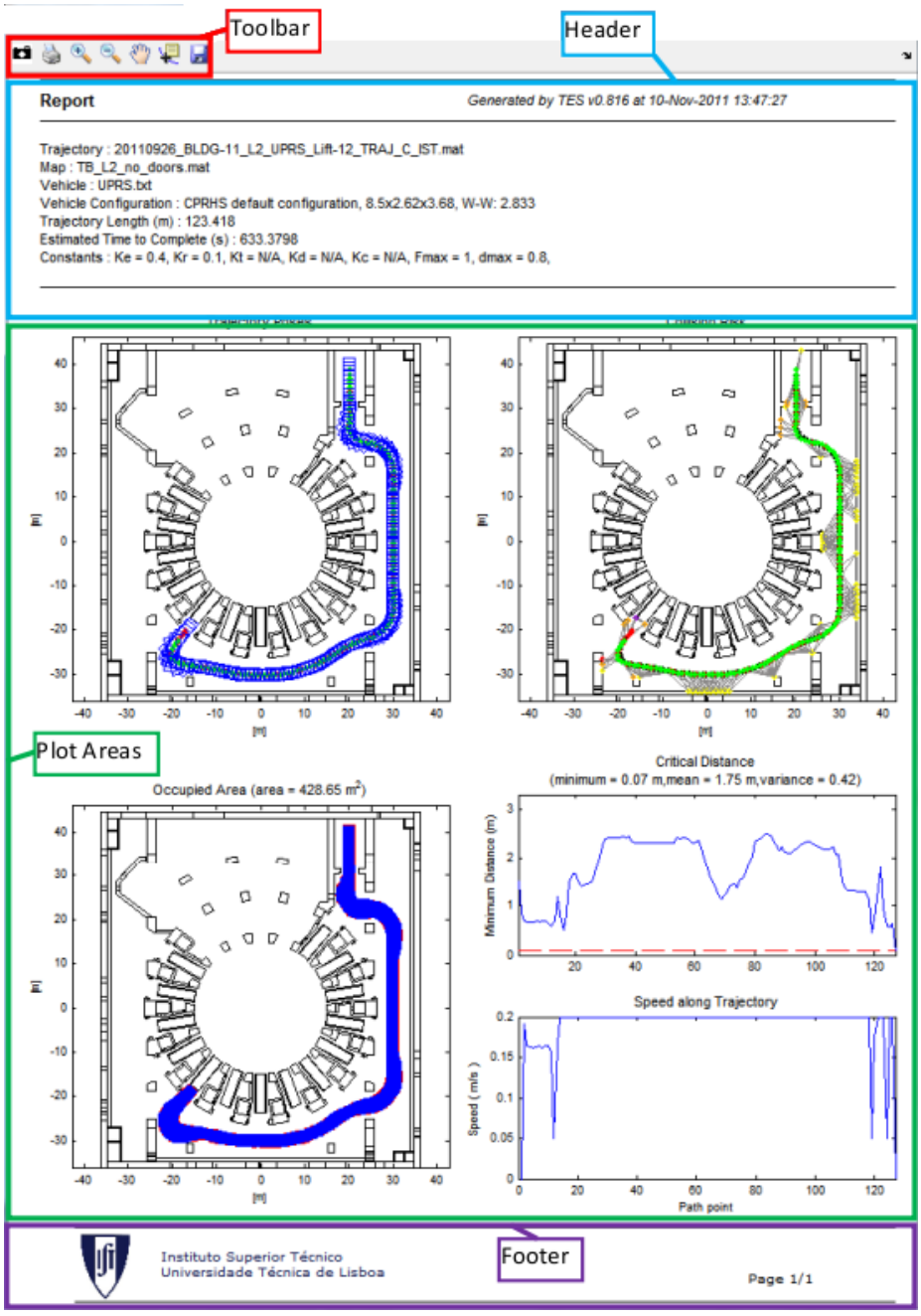


Figure 8.2: *Example of a report provided by TES.*

The main objective of the TES environment is to provide a complete analysis and validation for the trajectories that are going to be applied during ITER operation. It also contains control strategies for the path following, vehicle dynamics and localization algorithms, simulating very accurately the CPRHS missions. In the following section the current path planning procedure in TES is detailed. In section 8.3 this procedure is modified in order to include FM^2 as initialization algorithm. In section 8.4 a complete set of simulations are carried out, providing very interesting results. As there are some trajectories which require maneuvers, section 8.5 analyzes this problem and the performance of our proposed approach. Finally, in section 8.6 the saturated variation of FM^2 is included, improving the results even more.

8.2 Path planning procedure in the TES program

The vehicle is required to move along a path that simultaneously maximizes the clearance to obstacles and reducing the distance between the start and goal poses (position and orientation). The proposed motion planning methodology is based on a three step approach, (Fonte et al., 2010), as illustrated in Figure 8.3.

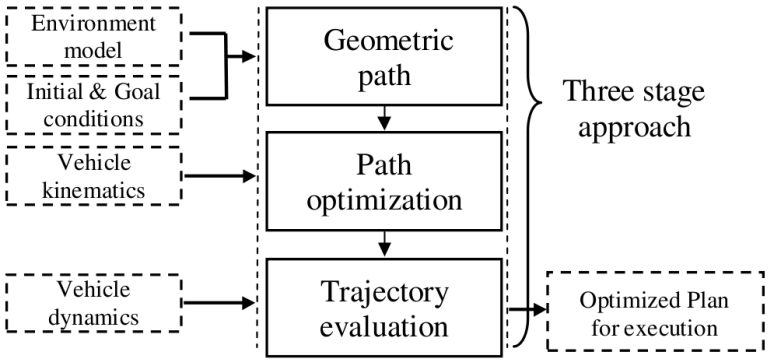


Figure 8.3: *Path planning procedure in TES.*

Geometric path evaluation: given the environment model and initial and goal poses, an initial geometric path is found. At this point the aim is to find a

path connecting the initial and goal objectives that can act as an initial condition for the next path optimization stage.

Path optimization: this module receives the preceding geometric solution as input and returns an optimized path.

Trajectory evaluation: in this final module, the speed profile is defined along the optimized path transforming it into a trajectory, which is the output of the proposed motion planning methodology.

8.2.1 Geometric path evaluation

From the 3D CAD models, a 2D representation is obtained by projecting at floor level all the relevant elements that might conflict with the CPRHS. The levels of TB and HCB are well structured scenarios that can be modeled as a set of planar walls, whose footprint is a line segment and thus the 2D map can be considered as a set of line segments. The 2D map is decomposed into a set of triangles, by using Constrained Delaunay Triangulation (Chew, 1987), to account for all walls (Figure 8.4). Afterwards the algorithm determines the set of sequence of triangles that contain and link the start and goal positions, as illustrated in Figure 8.5. Each sequence of triangles is then converted into a sequence of points (mid point of the common edge of two consecutive triangles) yielding a path. The shortest and feasible path is chosen as the geometric path, acting as the initial condition for the path optimization module as shown in Figure 8.6.

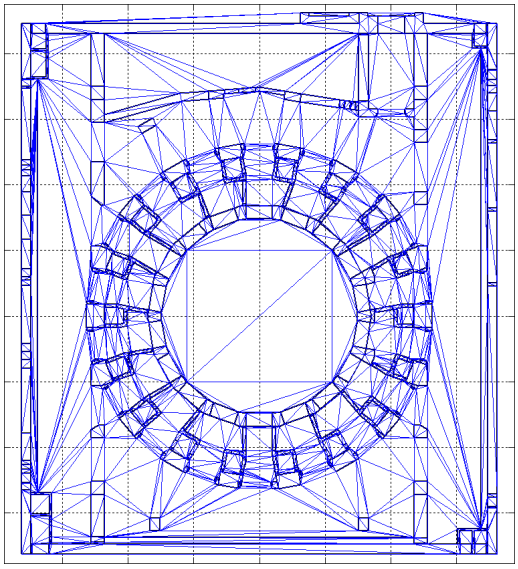
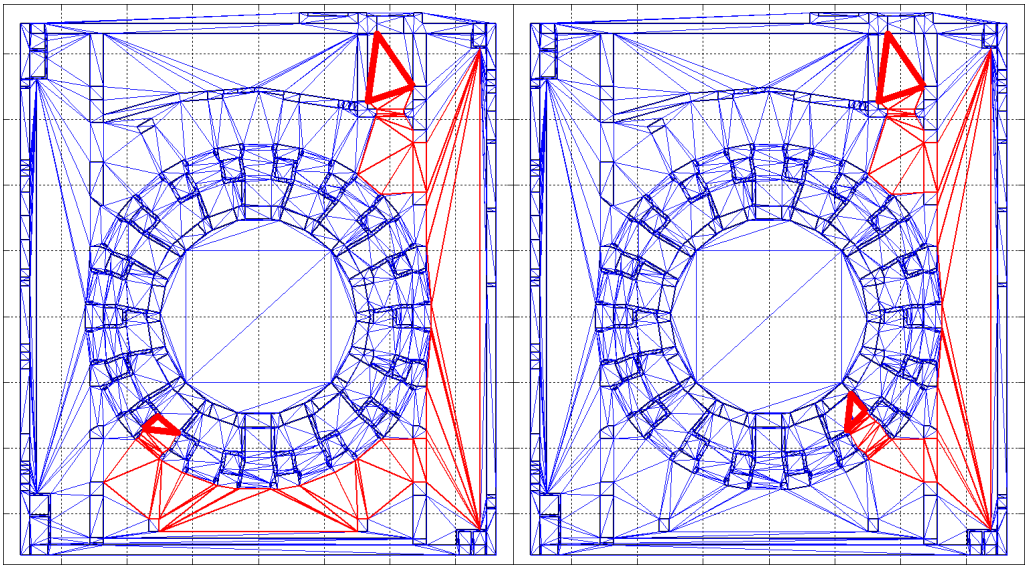
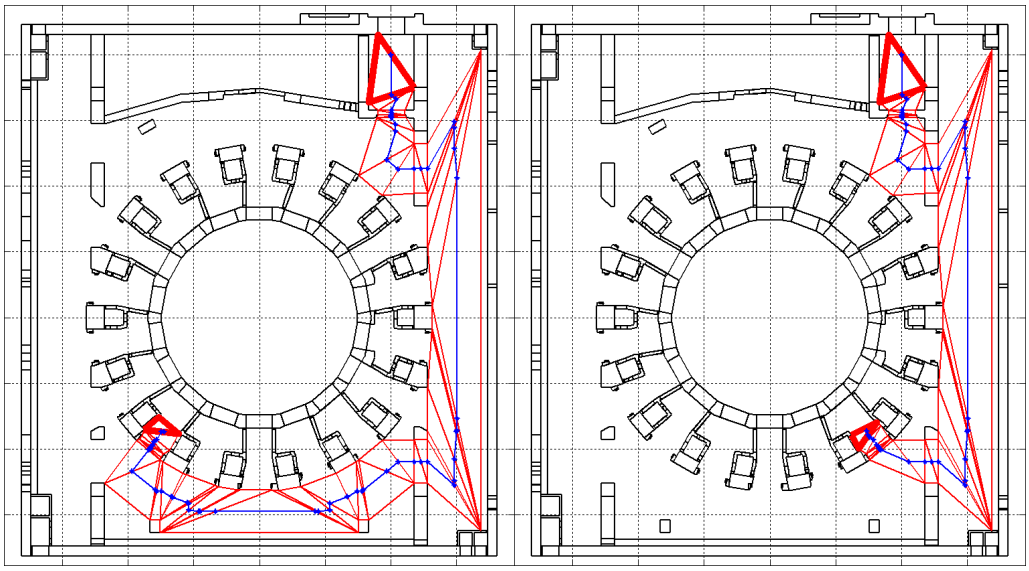


Figure 8.4: *The initial map with the generated Constrained Delaunay Triangulation.*



(a) *Port 12* **(b)** *Port 16*

Figure 8.5: *Sequence of triangles that link the start and goal positions.*



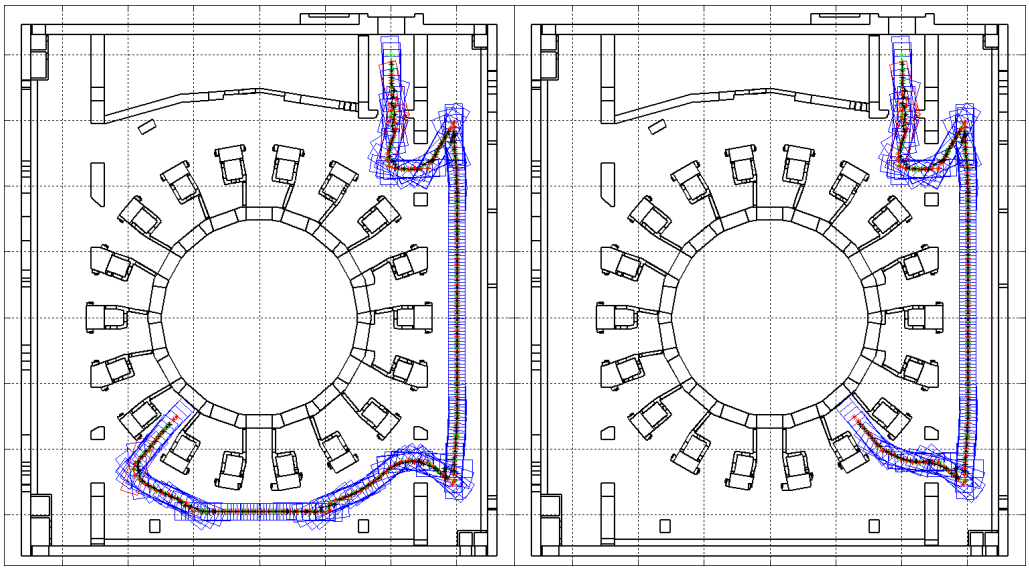
(a) *Port 12* **(b)** *Port 16*

Figure 8.6: *Sequence of points that generate the initial path of the procedure.*

8.2.2 Path optimization

The initial geometric path does not guarantee a collision free path for a rigid body, such as the CPRHS, as shown in the second image of Figure 8.7 and thus may be unfeasible. Moreover this path is not smooth. To obtain an optimized path, two criteria are included in the algorithm (Fonte et al., 2010): clearance from obstacles, by increasing the distance from the vehicle to the walls and path smoothness, entailing getting shorter and smoother paths without slacks. To address the referred issues, the optimization procedure uses the elastic band concept (Quinlan & Khatib, 1993), where the path is modeled as an elastic band, similar to a series of connected springs, subjected to two types of forces: internal and external forces. The first are the internal contraction forces, whose magnitude is proportional to the amplitude of displacement and determine that the path becomes retracted and shorter. The repulsive forces are responsible for keeping the path, and consequently the vehicle, away from obstacles (see Figure 8.8). After this procedure, the final paths are shown in Figure 8.9. Details about stopping criteria are detailed in section 8.4.

The procedure until the optimization is completed is summarized in figure 8.10.



(a) *Port 12* **(b)** *Port 16*

Figure 8.7: *Simulation of the CPRHS executing the initial path.*

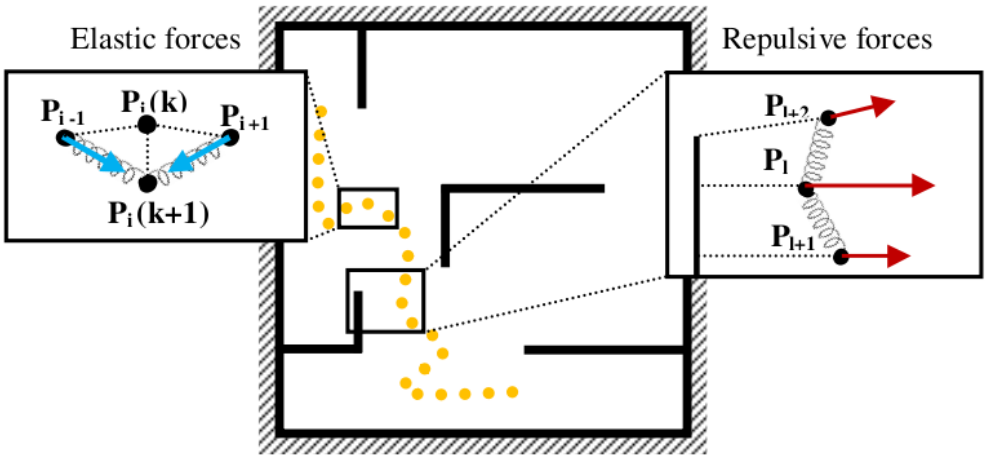
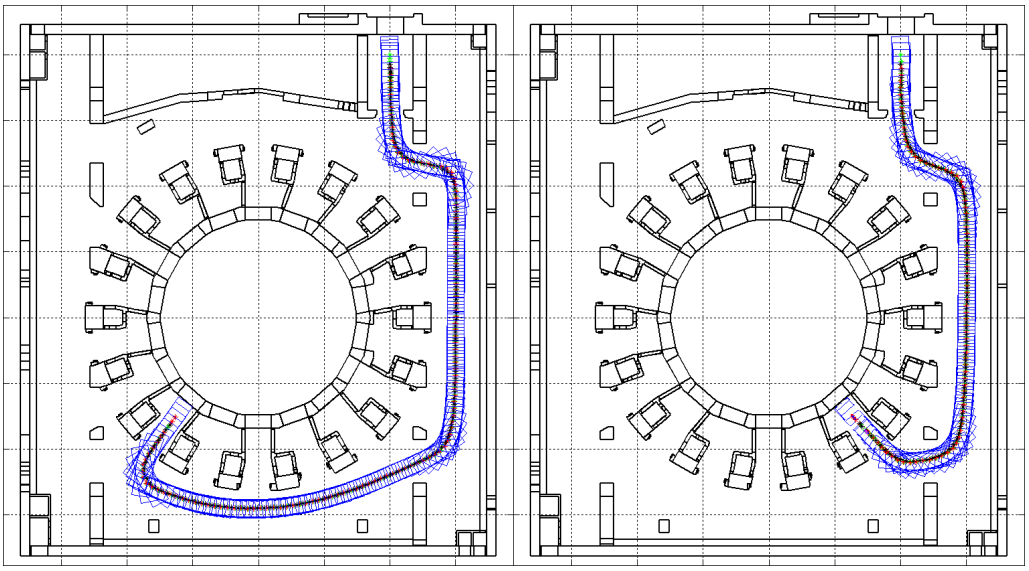


Figure 8.8: *Schema of the elastic bands concept applied in the path optimization.*



(a) *Port 12* **(b)** *Port 16*

Figure 8.9: *Simulation of the CPRHS executing the optimized path.*

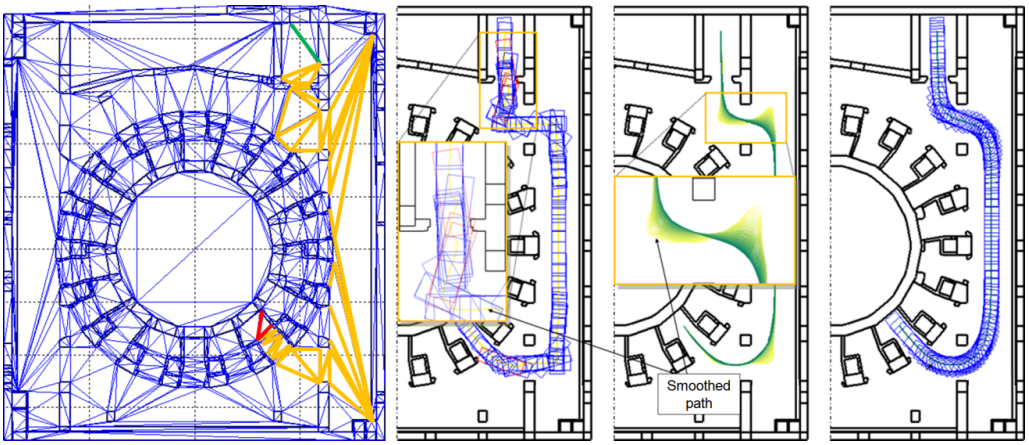


Figure 8.10: *From left to right: the initial map with the generated CDT and the computed sequence of triangles between start and goal points, initial geometric path, path optimization and final optimized trajectory.*

8.2.3 Trajectory evaluation

The final trajectory is obtained by defining the speed of the vehicle at each point of the optimized path. In order to reduce the risk of collision in the case of major malfunction, the speed is reduced once the distance to the nearest obstacle decreases below a threshold value. Dynamic constrains, such acceleration and speed limits, are also considered (Fonte et al., 2010).

Concretely, the CPRHS velocities are evaluated as a linear function of the minimum distance to obstacles. When this distance is above a specified threshold, a maximum allowable velocity for the CPRHS, v_{max}, is assumed. The velocity profile will thus resemble the minimum distance evolution along the path, as illustrated in figure 8.11.

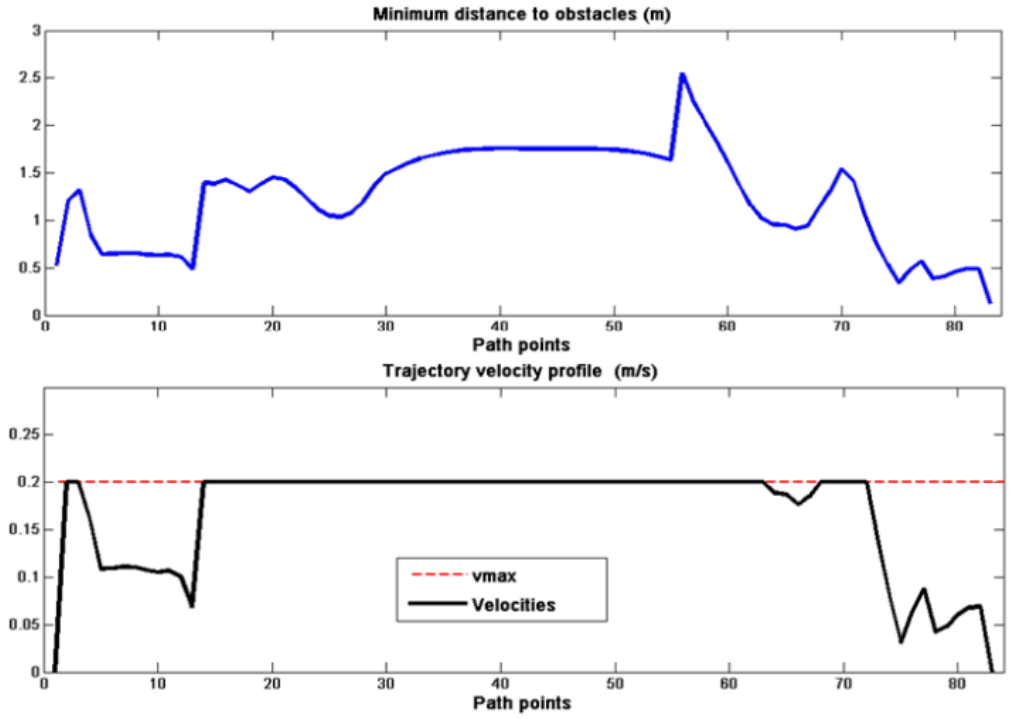


Figure 8.11: *Top: distance to the closest obstacle. Bottom: CPRHS velocity profile.*

8.2.4 Geometric path evaluation issues

A geometric representation of the environment has some advantages since it provides a very accurate representation of the scenario and it does not depends on grid cell discretization. However, in complex scenarios, the geometric representation results in some issues. In particular with the CDT, the representation requires a huge number of triangles, yielding to a computational effort. In addition, the CDT may result on sharp initial paths still far from the optimal one, as shown in Figure 8.7 and Figure 8.16. As a consequence, the optimization takes longer. To overcome these issues, in this chapter a new initialization method is proposed based on the FMM which is detailed in the next section.

8.3 Replacing geometric path evaluation with FM^2

The trajectory provided by FM^2 algorithm does not take into account the kinematics of the vehicle, neither it assures that it is feasible, since clashes might occur. However, the trajectory given by FM^2 algorithm is closer to a final solution, when compared to the initial geometric path obtained with CDT, as illustrated in the right image of Figure 8.12. Taking into account the work flow for trajectory optimization described in (Fonte et al., 2010), the geometric path initialization using CDT can be replaced by the FM^2, as illustrated in Figure 8.13.

Since the FM^2 algorithm returns a single path to be followed by both wheels of the vehicle, comparisons will be made considering only line guidance. In addition, most of the nominal operations are accomplished using line guidance.

8.4 Simulated results

In order to assess the quality of the trajectories obtained using either CDT or FM^2 initializations a comparison criteria is defined. The trajectories are compared in terms of clearance, smoothness, computation time and number of iterations

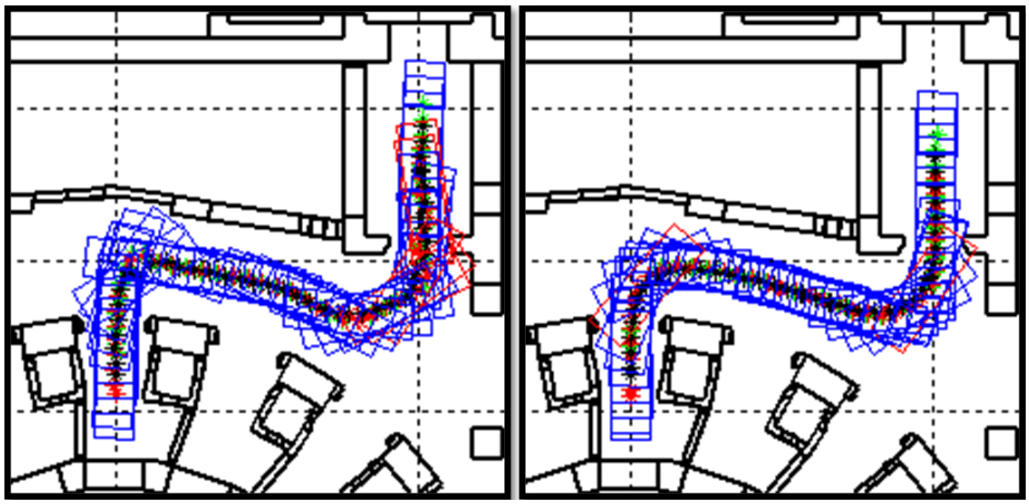


Figure 8.12: *Left - initial geometric path obtained with CDT; Right - trajectory obtained with FM^2.*

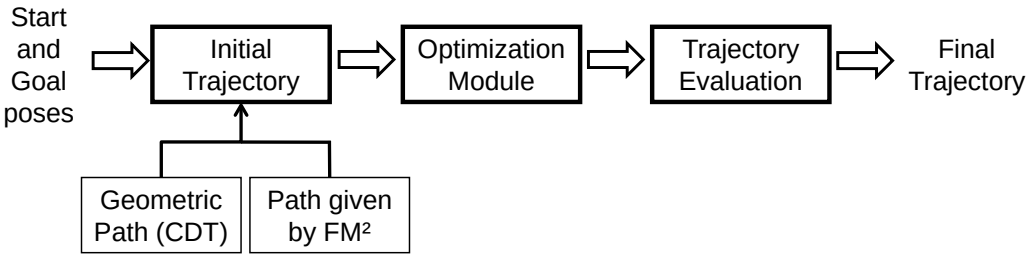


Figure 8.13: *Work flow for trajectory optimization.*

required in the optimization module.

In Figure 8.14 it is shown a schematic of the evolution of the path during the optimization process. The path connects the fixed initial and final points and has 3 additional points. The index n identifies the iteration of the optimization process.

At each iteration it is computed the variation of the path in relation to the previous iteration. The variation is computed evaluating the distance between each point of the path at iteration n and the line segment defined by the two nearest points of the path from iteration $n-1$, as illustrated in Figure 8.15. The 20 highest distances are selected and their median is computed. The stopping

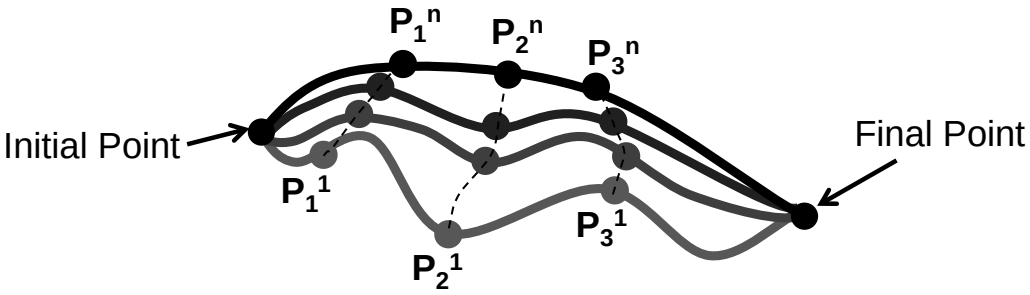


Figure 8.14: *Schema of the path evolution in each iteration during the trajectory optimization.*

criteria is achieved when the median value is below a threshold (defined as 0.02 m) or if the number of iterations reaches 70.

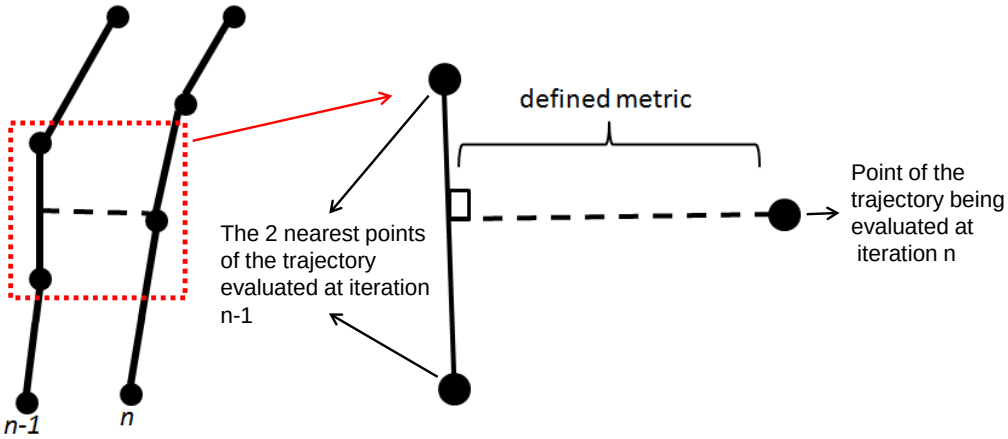


Figure 8.15: *Definition of the variation of the path between consecutive iterations: distance evaluated to a single point.*

The simulated results presented in this chapter were obtained using the map of level B1 of TB (in order to allow the reader to compare with the previous chapter). The CDT provides a rough initial path mainly in the vicinity of the lift, highly subject to the shape of the triangles, as shown in Figure 8.16. The FM^2 method is independent to the geometric representation used in CDT to generate an initial path.

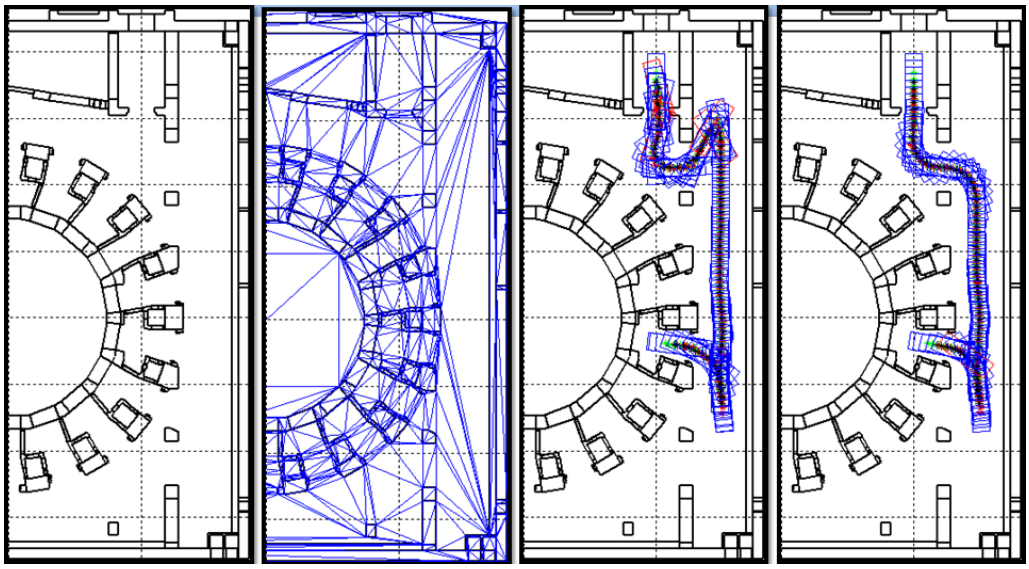
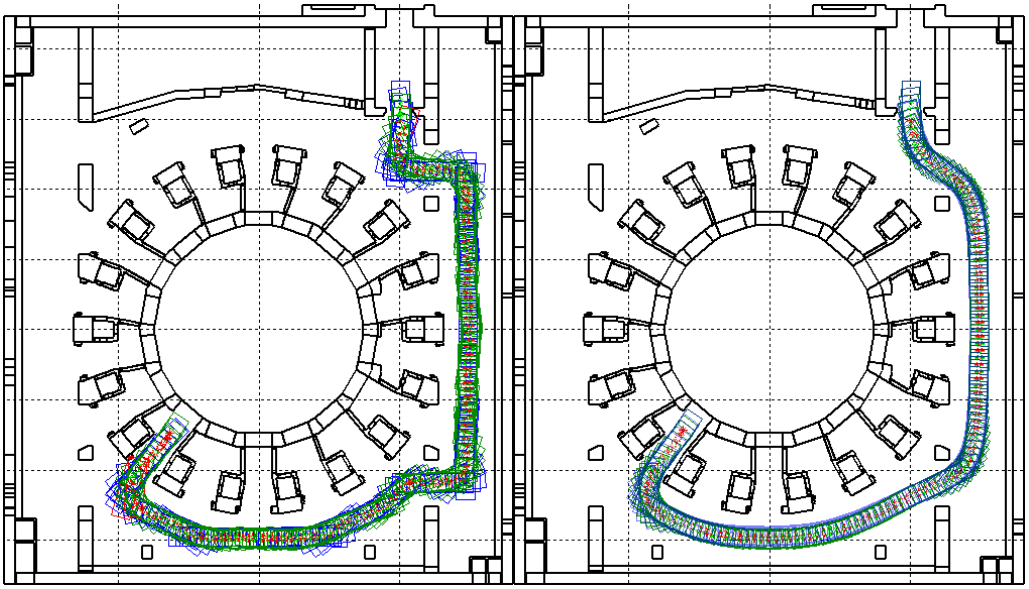


Figure 8.16: *From left to right: map of level B1 in TB, CDT of the map, geometric path obtained from CDT and path obtained with FM^2.*

In Figure 8.17 it is presented a comparison between the initial and final trajectories using CDT and FM^2, for port 12. The map was slightly modified to reduce the effect of the triangulation in CDT in the vicinity of the lift, as illustrated in Figure 8.16. Initially both present clashes, but the initial trajectory using FM^2 is smoother. However, the final optimized trajectories using the CDT and FM^2 initialization are very similar as illustrated in Figure 8.17. The variation of the 20 highest distances and the respective median along the iterations are shown in Figure 8.18, where the methodology with the FM^2 initialization required less number of iterations to converge. The differences between the optimized trajectories can be identified when comparing the minimum distances to the closest obstacle, as depicted in Figure 8.19. The trajectory using FM^2 initialization in general has slightly higher obstacles clearance.



(a) *Initial trajectories* **(b)** *Optimized trajectories*

Figure 8.17: *Trajectories for port 12: FM^2 (green) and CDT (blue).*

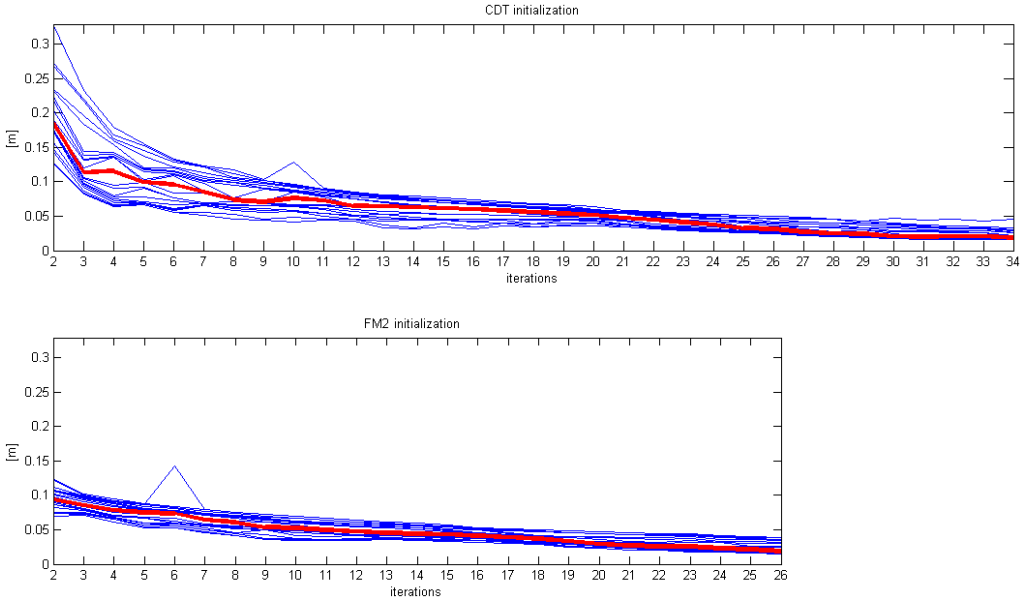


Figure 8.18: *Variation of the median (in red) along iterations for port 12 in level B1 of TB.*

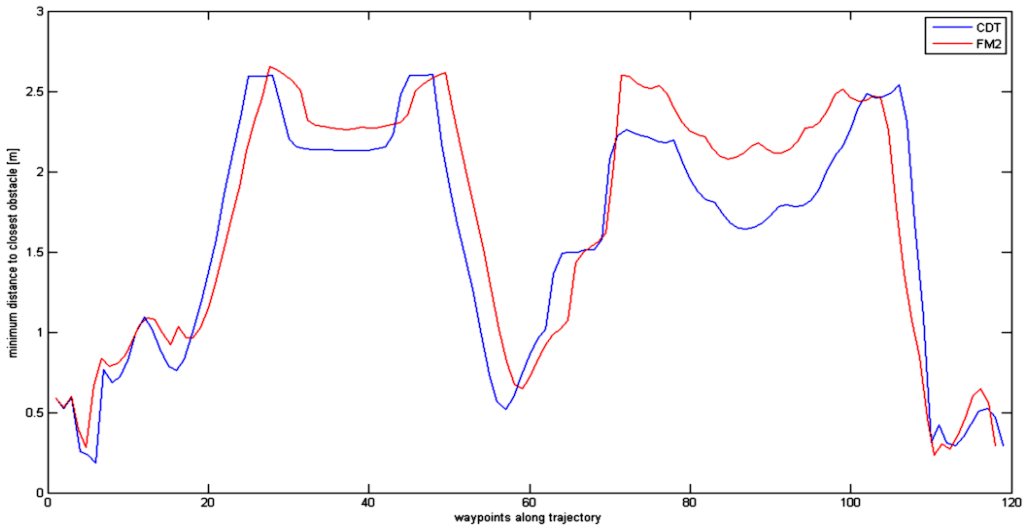


Figure 8.19: *Comparison between the minimum distances along the optimized trajectories using the CDT and FM^2 initializations for port 12 in level B1 of TB.*

In Figure 8.20 it is presented a summary of the comparison of results between FM^2 and CDT initializations. In general the FM^2 initialization requires less computation time and number of iterations to converge, over the CDT initialization, which allows to decrease the computational effort that is needed for the optimization.

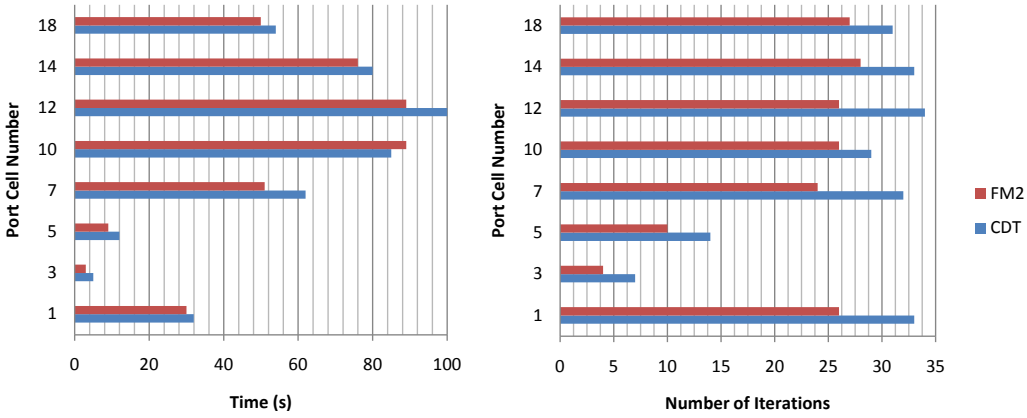


Figure 8.20: *Comparisons of computational time (left) and number of iterations (right) for trajectory optimization using CDT and FM^2.*

8.5 Inclusion of maneuvers

There are trajectories in TB that must include one or even two maneuver to assure the required safety margin to the closest obstacles. Two examples of trajectories with one and with two maneuvers are depicted in Figure 8.21.

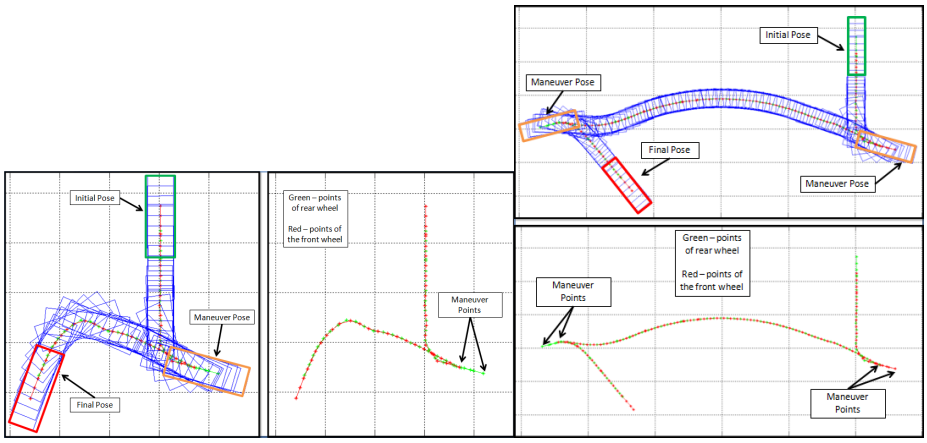


Figure 8.21: *Examples of trajectories which requires maneuvers. The maneuver pose is defined by the location of the wheels.*

In Table 8.1 it is detailed which of the trajectories require maneuvers, which are most of them.

Table 8.1: *Number of maneuvers for every port of the TB.*

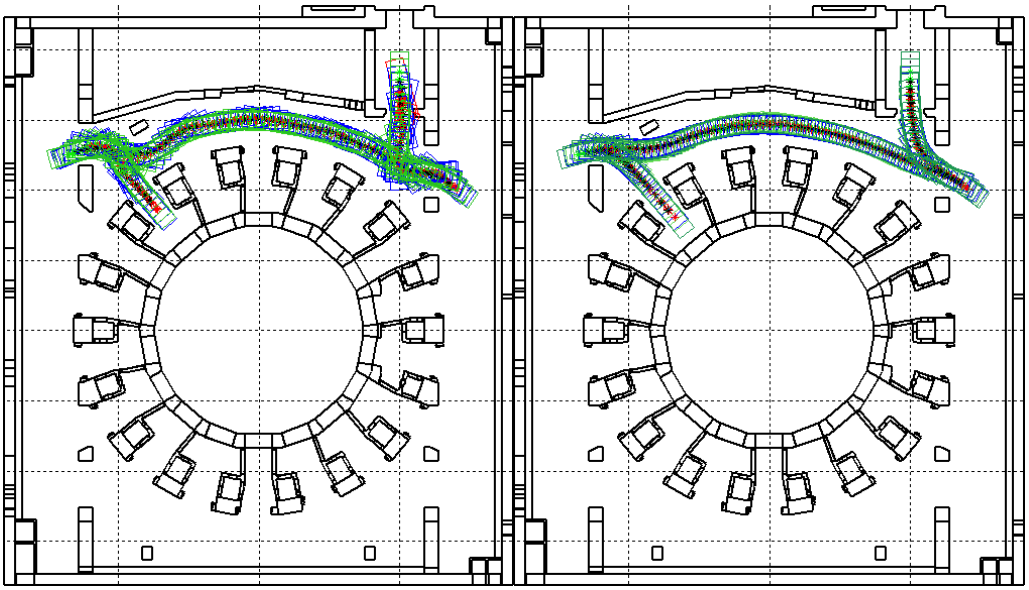
Port	4	5	6	7	8	9	10	15	16	17	18
# of maneuvers	1	1	1	2	1	1	2	1	1	1	1

From the path planning point of view, these maneuvers are treated as independent paths which are concatenated before the optimization procedure. Therefore, for n maneuvers, it will be necessary to compute $n + 1$ paths. Due to this fact, it is possible to ensure that the integration of FM^2 will perform as well as done in the previous sections. In any case, some simulations have been carried out within TES, as detailed in Table 8.2 and depicted in Figure 8.22. As

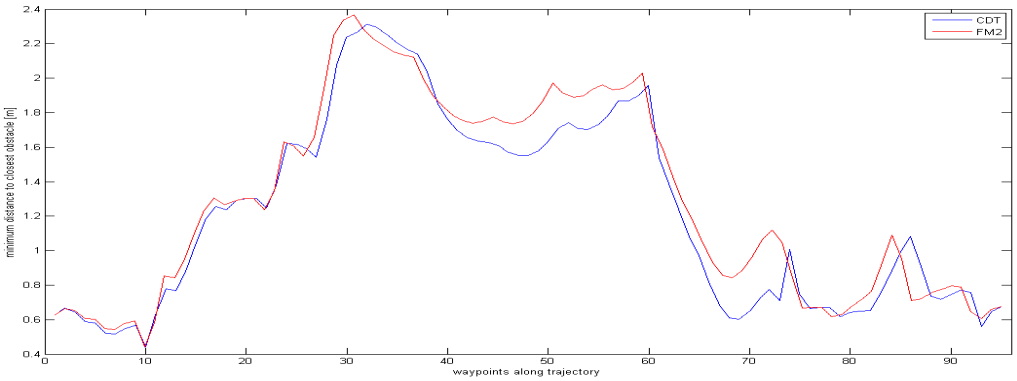
expected, there is a reduction in the computation time, and a improvement on the minimum distances according to the previous section and the final paths are very similar with both CDT and FM^2 initialization.

Table 8.2: *Results of maneuvers simulations in terms of iterations and time elapsed.*

		CDT Initialization		FM^2 Initialization	
		# of Iterations	Time elapsed (s)	# of Iterations	Time elapsed (s)
Port cell	7	32	62	25	53
	18	34	76	29	65



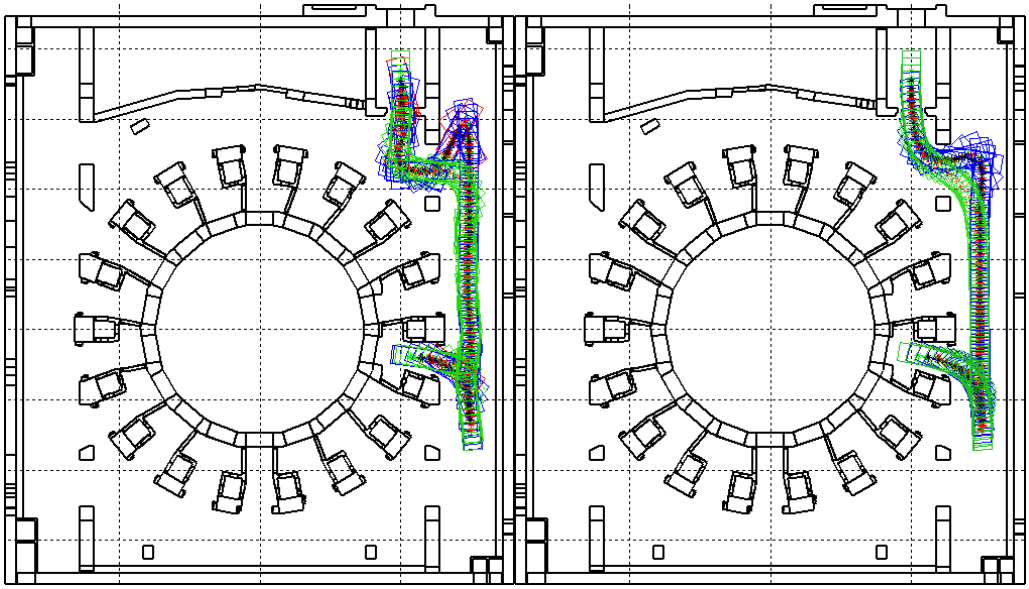
(a) *Initial trajectories* **(b)** *Optimized trajectories*



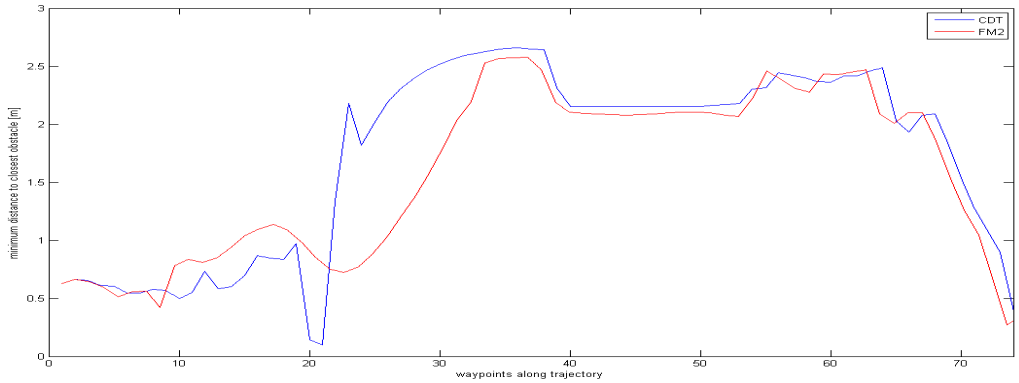
(c) *Distances comparison*

Figure 8.22: *Maneuvers in port cell 7: FM^2 (green) and CDT (blue).*

Figure 8.23 shows an example of maneuvers on a different port.



(a) *Initial trajectories* **(b)** *Optimized trajectories*

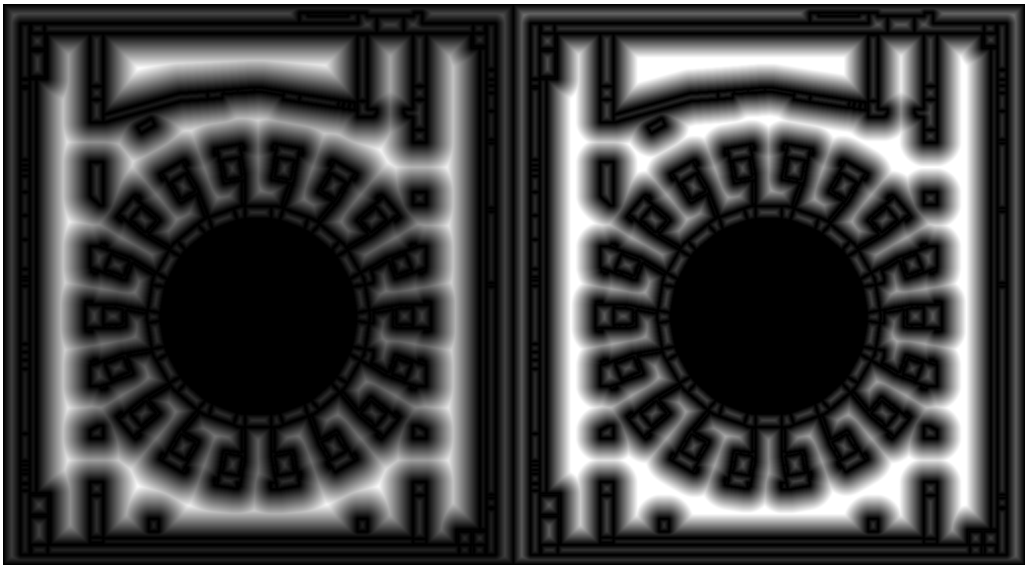


(c) *Distances comparison*

Figure 8.23: *Maneuvers in port cell 18: FM^2 (green) and CDT (blue).*

8.6 Performance of the saturated variation of FM^2

In order to analyze if the saturated variation of FM^2 provides any improvement, the same ports of the previous section have been studied. The saturation level for the velocities map was experimentally set to 0.7. Although this saturation level could not be really meaningful, in Figure 8.24 a comparison against the standard FM^2 velocities map and the saturated version is given.



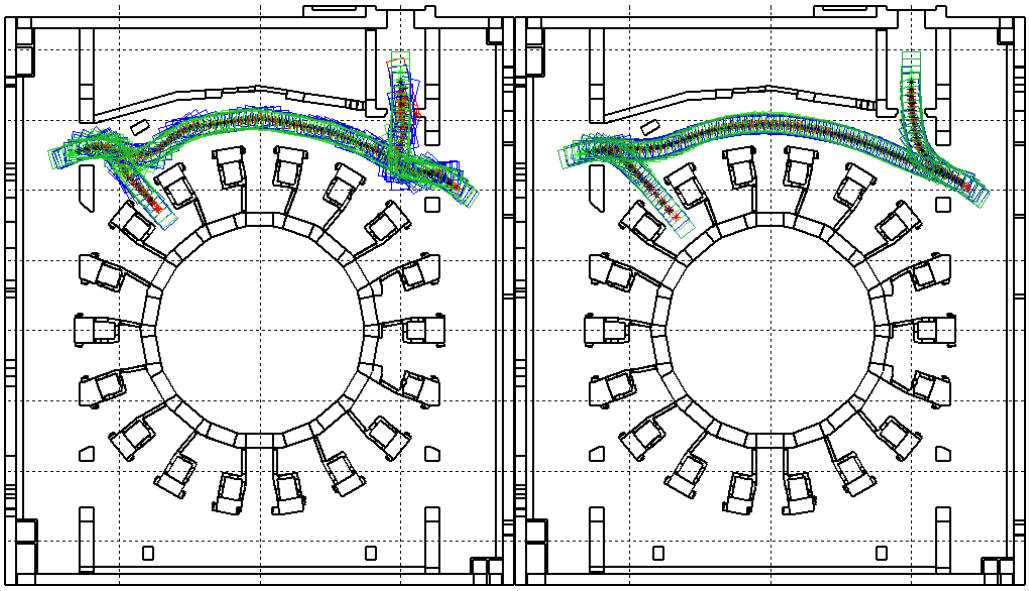
(a) *No saturation.* **(b)** *Saturation level: 0.7.*

Figure 8.24: *Comparison of the different velocities map* **W** *employed.*

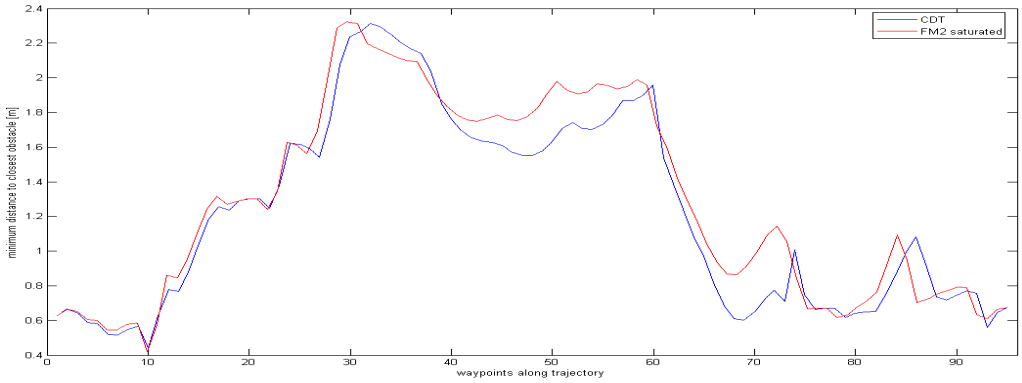
The results provided in Table 8.3 show that the initialization using the saturated variation of FM^2 outperforms even more the CDT initialization since it is faster than the standard FM^2 initialization. The minimum distances and the final paths remain almost the same as when using the standard FM^2 initialization, as shown in Figure 8.25 and Figure 8.26.

Table 8.3: *Comparison of FM^2 and its saturated variation in terms of iterations and time elapsed.*

		FM^2 Initialization		Saturated FM^2 Initialization	
		# of Iterations	Time elapsed (s)	# of Iterations	Time elapsed (s)
Port cell	7	25	53	24	51
	18	29	65	27	50

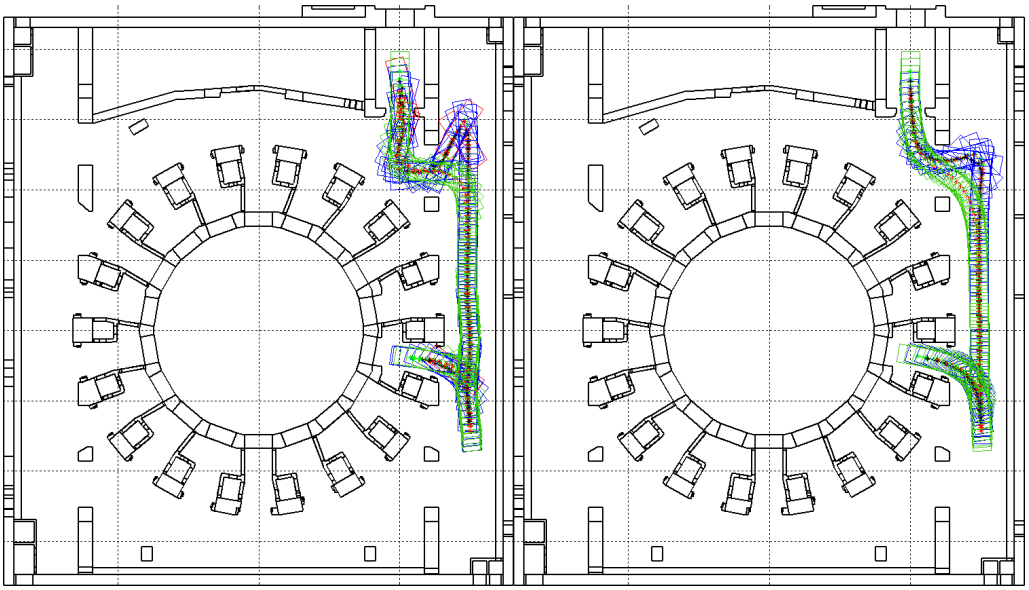


(a) *Initial trajectories* **(b)** *Optimized trajectories*

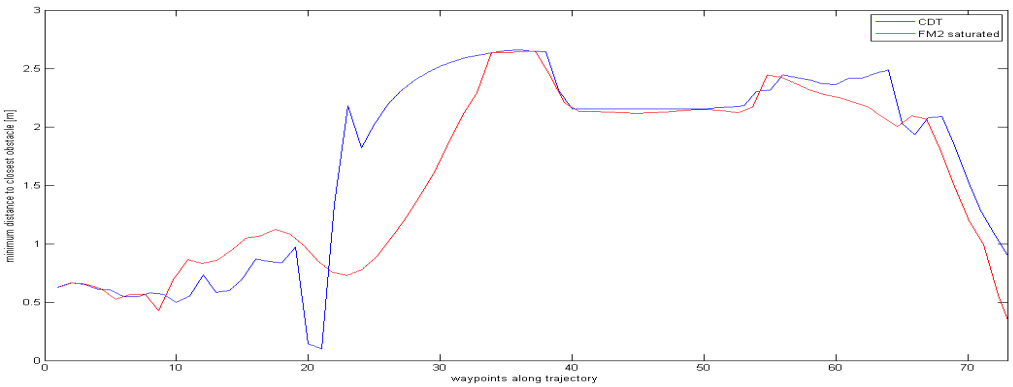


(c) *Distances comparison*

Figure 8.25: *Maneuvers in port cell 7: saturated FM^2 (green) and CDT (blue).*



(a) *Initial trajectories* **(b)** *Optimized trajectories*



(c) *Distances comparison*

Figure 8.26: *Maneuvers in port cell 18: saturated FM^2 (green) and CDT (blue).*

8.7 Conclusions and future work

This chapter presented an improvement to the motion planning methodology used to compute the trajectory optimization in remote handling operations of ITER or in other cluttered environments. The geometric path initialization using CDT was replaced by a powerful method: the FM^2. Given the simulated results, the optimization procedure is faster using the FM^2 when compared with the CDT. The FM^2 is also independent to the triangle representation, provides a smoother initial path and has no local minima. In terms of clearance the final trajectories with both methods are similar with small improvements when using the FM^2.

We remark that the velocities profiles are not compared in this case because both TES and FM^2 implicitly compute these profiles in the same way: as a linear function of the distance to the closest obstacle. Therefore, we consider that this comparison is not necessary.

Further more, the analysis carried out with the saturated variation of FM^2 indicates that there is still an improvement margin in the optimization procedure. However, this turns into a more complex problem in which the saturation value has to be carefully chosen.

The future work in this section focuses on speeding up the optimization algorithm, since there are some parts of the FM^2 initialization path which do no require to be optimized. The application of this procedure to other different scenarios is also a very interesting research topic.

Chapter 9

Conclusions and Future Work

Although specific conclusions and future work for each chapter have been given, in this chapter the main conclusions of the whole document are extracted.

9.1 Conclusions

As shown in previous theses and journal papers, the Fast Marching Method is a very versatile one, not only for path planning but also for more complex applications which require a path planning step (Garrido, 2008),(Sanctis, 2010),(Jurewicz, 2010). This thesis goes further with the application of the Fast Marching Method and the Fast Marching Square planning method. We have detailed how to integrate the Fast Marching Square method to very different problems: robot formation path planning in chapter 4, adaptation of the path planning to the environment in chapter 5 or to previous experiences (or learning) in chapter 6. Finally, in chapters 7 and 8 the Fast Marching Square method has been applied to a real, very complex scenario which supposes the Remote Handling operations in the ITER project, where the requirements are very strong and restrictive.

Along all the chapters, it has been noticeable the capability of the Fast Marching Square algorithm to be adapted to every different problems with very good

results. The main point to remark is that, while in other approaches the planning method is used as a tool to achieve the objective, whilst in our case is the proper path planning method which accomplishes the mission. In other words, we can talk about a *generalized* path planning method because most of the planning problems (if not all of them) can be undertaken with Fast Marching Square, regardless the number of dimensions of the problem, since Fast Marching Method is defined for any number of dimensions.

However, the main drawback of Fast Marching Square is that it is based on a grid cell representation of the environment. The Fast Marching Method has a low computational complexity of $O(n\ log(n))$ although it can be approximated with an $O(n)$ implementation (Yatziv et al., 2005), where n represents the number of cells. Therefore, since Fast Marching Square consists on applying the Fast Marching Method twice, its computational complexity is also $O(n\ log(n))$ or $O(n)$, depending on the implementation.

Nevertheless, the number of cells n increases exponentially when the number of dimensions of the environment gets higher. Because of this, Fast Marching Square is useful up to 3 dimensions but for a higher number of dimensions it starts to be very slow. Fortunately, most of the applications can be formulated into a 2D or 3D problem. For example, instead of computing a 6-dimensional path for a robotic arm with 6 degrees of freedom, it could be computed the end-effector coordinates in 3D and apply inverse kinematics.

9.2 Future work

Very detailed future work guidelines have been proposed in each chapter. The results obtained and the ideas for new developments turn Fast Marching Square into a very interesting field of research. Further more, other works parallel to this thesis have been carried out so new, improved versions of the Fast Marching Square have been designed.

For instance, the Fast Marching Square Star (FM^2*) incorporates an heuristic

to improve the computational time when calculating a new path, as A* does with Dijkstra. Also, a directional Fast Marching Square (dFM^2) has been developed. In this new version, the velocity profile of the robot not only depends on the velocities map, but also on its gradient. So far, with the standard Fast Marching Square, when a robot is very close to a wall or obstacle but is getting farther from it, the velocity is low without considering that the robot is trying to get far. With this new version, the velocity profile and also the paths are modified in order to obtain more logical and optimal behaviours of the robot.

The application of these two new variants to the proposed problems in this thesis could improve the performance and new challenged could be faced.

Finally, another, totally different work proposal is to create a sampling-based version of the Fast Marching Method. This could suppose the removal of its main drawback: grid cell decomposition and, therefore, apply the same Fast Marching Method-based applications to even higher dimensions with very good computational times.

9.3 Relevance of the work

The work carried out within this thesis have been recognized by the researchers community. Some journal papers have been published (Garrido, Moreno, Gómez, & Lima, 2012), (Gómez, Lumbier, Garrido, & Moreno, 2012) and accepted (Valero-Gómez, Gómez, Garrido, & Moreno, 2012). The work has been exposed in many conferences, both international (Garrido, Moreno, Lima, & Gómez, 2012), (Gómez, Álvarez, Garrido, & Moreno, 2012), (Gómez, Arismendi, Garrido, & Moreno, 2012) and national (Gómez, Garrido, & Moreno, 2012), (Gómez, Garrido, Moreno, Vale, et al., 2012a). A paper has been also submitted to ICRA2013 (Gómez et al., 2013) and a book chapter is going to be published in the next weeks (Gómez, Garrido, Moreno, Vale, et al., 2012b). Also, an implementation of the FM^2 method has been carried out in (Giakoumidis, Bak, Gómez, Llenga, & Mavridis, 2012), where an Unmanned Ground Vehicle (UGV) is assisted in the

path planning with an Unmanned Air Vehicle (UAV).

Appendix A

List of Acronyms

Acronyms

CDT Constrained Delaunay Triangulation

CPRHS Cask and Plug Remote Handling Systems

CTS Cask Transfer Sytem

DARPA Defense Advanced Research Project Agency

DMP Dynamic Movement Primitives

DOF Degrees of Freedom

DUE Dynamic Uncertain Environment

EVT Extended Voronoi Transform

FMM Fast Marching Method

FM^2 Fast Marching Square

HCB Hot Cell Building

HMM Hidden Markov Models

IROS IEEE/RSJ International Conference on Intelligent Robotics and Systems

ITER International Thermonuclear Experimental Reactor

ML-RRT Manhattan-like Rapidly-exploring Random Trees

MLT-RRT Manhattan-like Transition-based Rapidly-exploring Random Trees

RH Remote Handling

RRT Rapidly-exploring Random Trees

TB Tokamak Building

TES Trajectory Evaluator and Simulator

T-RRT Transition-based Rapidly-exploring Random Trees

VFM Voronoi Fast Marching

VV Vacuum Vessel

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship Learning via Inverse Reinforcement Learning. In *International Conference on Machine learning* (pp. 1–6).

Asano, T., Asano, T., Guibas, L. J., Hershberger, J., & Imai, H. (1986). Visibility of Disjoint Polygons. *Algorithmica*, *1*(1), 49–63.

Attali, D., Boissonnat, J.-D., & Lieutier, A. (2003). Complexity of the Delaunay Triangulation of Points on Surfaces: the Smooth Case. In *Symposium on Computational Geometry* (pp. 201–210).

Balch, T., & Arkin, R. C. (1998). Behavior-based Formation Control for Multirobot Systems. *IEEE Transactions on Robotics and Automation*, *14*(12), 926–939.

Barranquand, J., Langlois, B., & Latombe, J. C. (1992). Numerical Potential Field Techniques for Robot Path Planning. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*(2), 224–241.

Beard, R. W., Lawton, J., & Hadaegh, F. Y. (2001). A Coordination Architecture for Spacecraft Formation Control. *IEEE Transactions on Control Systems Technology*, *9*(6), 777–790.

Berenson, D., Abbeel, P., & Goldberg, K. (2012). A Robot Path Planning Framework That Learns From Experience. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3671–3678).

Berg, M. de, Kreveld, M. van, Overmars, M., & Schwarzkopf, O. (2008). *Computational Geometry: Algorithms and Applications* (3rd ed.). Springer-Verlag.

Cai, Y., & Yang, S. X. (2012). A Survey on Multi-robot Systems . In *World Automation Congress (WAC)* (pp. 1–6).

Calinon, S. (2009). *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press.

Calinon, S., Sardellitti, I., & Caldwell, D. G. (2010). Learning-based Control Strategy for Safe Human-robot Interaction Exploiting Task and Robot Redundancies. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 249–254).

Chew, L. P. (1987). Constrained Delaunay Triangulations. In *Annual Symposium on Computational Geometry* (pp. 215–222).

Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., et al. (2007). *Principles of Robot Motion*. MIT Press.

Cortes, J., Jaillet, L., & Siméon, T. (2008). Disassembly Path Planning for Complex Articulated Objects. *IEEE Transactions on Robotics*, *24*(2), 475–481.

DARPA. (2011). *Urban Challenge, last visit: November, 2012.* Available from `http://archive.darpa.mil/grandchallenge`

Das, A. K., Fierro, R., Kumar, V., Ostrowski, J. P., Spletzer, J., & Taylor, C. J. (2002). A Vision-Based Formation Control Framework. *IEEE Transactions on Robotics and Automation*, *18*(5), 813–825.

Delaunay, B. (1934). Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, *7*, 793–800.

Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, *1*, 269–271.

Du Toit, N. E., & Burdick, J. W. (2012). Robot Motion Planning in Dynamic, Uncertain Environments. *IEEE Transactions on Robotics*, *28*(1), 101–115.

Egerstedt, M., & Hu, X. (2001). Formation Constrained Multi-agent Control. *IEEE Transactions on Robotics and Automation*, *17*(6), 947–951.

Ettlin, A., & Bleuler, H. (2006). Randomised Rough-Terrain Robot Motion Planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5798–5803). IEEE.

Fierro, R., Song, P., Das, A., & Kumar, V. (2002). Cooperative Control of Robot Formations. In *Cooperative Control and Optimization.* Kluwer Academic Press.

Fiorini, P., & Shiller, Z. (1998). Motion Planning in Dynamic Environments Using Velocity Obstacles . *International Journal of Robotics Research*, *17*(7), 760–772.

Fonte, D., Valente, F., Vale, A., & Ribeiro, I. (2010). A Motion Planning Methodology for Rhombic-like Vehicles for ITER Remote Handling Operations. In *IFAC Symposium on Intelligent Autonomous Vehicles* (p. 443-448).

Frenet, F. (1852). Sur les Courbes à Double Courbure. *Journal de Mathématiques Pures et Appliquées*, *1*(17), 437–447.

Garrido, S. (2008). *Robot Planning and Exploration Using Fast Marching*. Master's thesis, Carlos III University of Madrid.

Garrido, S., Moreno, L., Abderrahim, M., & Blanco, D. (2009). FM2: A Real-time Sensor-based Feedback Controller for Mobile Robots. *International Journal of Robotics and Automation*, *24*(1), 3169–3192.

Garrido, S., Moreno, L., Abderrahim, M., & Martin, F. (2006). Path Planning for Mobile Robot Navigation Using Voronoi Diagram and Fast Marching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2376–2381).

Garrido, S., Moreno, L., & Blanco, D. (2008). Exploration of a Cluttered Environment using Voronoi Transform and Fast Marching Method. *Robotics and Autonomous Systems*, *56*(12), 1069–1081.

Garrido, S., Moreno, L., & Blanco, D. (2009). Exploration and Mapping Using the VFM Motion Planner. *IEEE Transactions on Instrumentation and Measurement*, *58*(8), 2880–2892.

Garrido, S., Moreno, L., Blanco, D., & Martin, F. (2009). Smooth Path Planning for Non-holonomic Robots using Fast Marching. In *Proceedings of the 2009 IEEE International Conference on Mechatronics* (pp. 1–6).

Garrido, S., Moreno, L., Blanco, D., & Muñoz, M. L. (2007). Sensor-based Global Planning for Mobile Robot Navigation. *Robotica*, *25*(2), 189–199.

Garrido, S., Moreno, L., Gómez, J. V., & Lima, P. U. (2012). General Path Planning Methodology for Leader-Followers based Robot Formations. *International Journal of Advanced Robotic Systems*, Accepted.

Garrido, S., Moreno, L., & Lima, P. U. (2011). Robot Formations Motion Planning using Fast Marching. *Robotics and Autonomous Systems*, *59*(9), 675–683.

Garrido, S., Moreno, L., Lima, P. U., & Gómez, J. V. (2012). Robot Formations Motion Planning using Fast Marching. In *ROBOT: Robótica Experimental* (pp. 233–240).

Ghosh, S. K., & Mount, D. M. (1991). An Output-sensitive Algorithm for Computing Visibility Graphs. *SIAM Journal on Computing*, *20*(5), 888–910.

Giakoumidis, N., Bak, J. U., Gómez, J. V., Llenga, A., & Mavridis, N. (2012). Pilot-scale Development of a UAV-UGV Hybrid with Air-Based UGV Path Planning. In *International Conference on Frontiers of Information Technology.*

Godage, I. S., Branson, D., Guglielmino, E., & Caldwell, D. G. (2012). Path Planning for Multisection Continuum Arms . In *IEEE International Conference on Mechatronics and Automation* (pp. 1208–1213).

Gómez, J. V., Álvarez, D., Garrido, S., & Moreno, L. (2012). Kinesthetic Teaching via Fast Marching Square. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Gómez, J. V., Arismendi, C., Garrido, S., & Moreno, L. (2012). On Path Planning: Adaptation to the Environment using Fast Marching. In *IEEE Conference on Evolving and Adaptive Intelligent Systems* (pp. 74–79).

Gómez, J. V., Garrido, S., & Moreno, L. (2012). Adaptive Robot Formations using Fast Marching Square working under Uncertainty Conditions. In *IEEE Workshop on Advanced Robotics and its Social Impacts* (pp. 68–71).

Gómez, J. V., Garrido, S., Moreno, L., Vale, A., Valente, F., & Ferreira, J. (2012a). Estudio de Funcionamiento del Algoritmo FM2 Aplicado al ITER. In *2º Workshop Programa TechnoFusión.*

Gómez, J. V., Garrido, S., Moreno, L., Vale, A., Valente, F., & Ferreira, J. (2012b). Performance Study of the FM2 Planning Method for Remote Handling Operations in ITER. In *Programa TechnoFusión.*

Gómez, J. V., Lumbier, A., Garrido, S., & Moreno, L. (2012). Planning Robot Formations with Fast Marching Square including Uncertainty Conditions. *Robotics and Autonomous Systems*(Accepted).

Gómez, J. V., Vale, A., Valente, F., Ferreira, J., Garrido, S., & Moreno, L. (2013). Fast Marching in Motion Planning for Rhombic like Vehicles Operating in ITER . In *IEEE International Conference on Robotics and Automation (ICRA).*

Greene, B. R. (1999). *The Elegant Universe: Superstrings, Hidden Dimensions, and the Quest for the Ultimate Theory*. Vintage Books.

Hart, P., Nilsson, N., & Raphael, B. (1968, july). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on*, *4*(2), 100–107.

Iehl, R., Cortés, J., & Siméon, T. (2012). Costmap Planning in High Dimensional Configuration Spaces . In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (pp. 166–172).

Jaillet, L., Cortés, J., & Siméon, T. (2010). Sampling-Based Path Planning on Configuration-Space Costmaps. *IEEE Transactions on Robotics*, *26*(4), 635–646.

Jbabdi, S., Bellec, P., Toro, R., Daunizeau, J., Pélégrini-Issac, M., & Benali, H. (2008). Accurate Anisotropic Fast Marching for Diffusion-based Geodesic Tractography. *International Journal of Biomedical Imaging*, *2008*.

Jurewicz, P. (2010). *Smooth Path-planning for a Robot Manipulator Using Probabilistic Methods*. Master's thesis, Carlos III University of Madrid.

Karaman, S., & Frazzoli, E. (2011). Incremental Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research*, *30*(7), 846–894.

Khansari-Zadeh, S. M., & Billard, A. (2011). Learning Stable Nonlinear Dynamical Systems with Gaussian Mixture Models. *IEEE Transactions on Robotics*, *27*(5), 943–957.

Kuffner, J. J., & LaValle, S. M. (2000). RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *IEEE International Conference on Robotics and Automation (ICRA)* (Vol. 2, pp. 995–1001).

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.

LaValle, S. M. (2011). Motion Planning. *IEEE Robotics and Automation Magazine*, *18*(1), 79–89.

Lee, D., & Ott, C. (2010). Incremental Motion Primitive Learning by Physical Coaching using Impedance Control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4133–4140).

Leonard, N. E., & Fiorelli, E. (2001). Virtual Leaders, Artificial Potentials and Coordinated Control of Groups. In *IEEE Conference on Decision and Control* (pp. 2968–2973).

Li, H., Xue, Z., Cui, K., & Wong, S. T. C. (2011). Diffusion Tensor-based Fast Marching for Modeling Human Brain Connectivity Network. *Computerized Medical Imaging and Graphics*, *35*(3), 167–178.

MacArthur, E. Z., & Crane, C. D. (2007). Compliant Formation Control of a Multi-Vehicle System. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation* (pp. 479–484).

Malfaz, M., Garrido, S., & Blanco, D. (2012). Application of the Fast Marching Method for Outdoor Motion Planning in Robotics. *Robotics and Autonomous Systems*(Accepted).

Melchior, N., & Simmons, R. (2012). Graph-Based Trajectory Planning through Programming by Demonstration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Michel, D., & McIsaac, K. (2012). New Path Planning Scheme for Complete Coverage of Mapped Areas by Single and Multiple Robots . In *IEEE International Conference on Mechatronics and Automation* (pp. 1233–1240).

Ogren, P., Fiorelli, E., & Leonard, N. E. (2003). Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment. *IEEE Transactions on Automatic Control*, *49*(8), 1292–1302.

Osher, S., & Sethian, J. A. (1988). Fronts Propagating with Curvature-dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, *79*(1), 12–49.

Quinlan, S., & Khatib, O. (1993). Elastic Bands: Connecting Path Planning and Control. In *IEEE International Conference on Robotics and Automation (ICRA)* (Vol. 2, pp. 802–807).

Ribeiro, I., Damiani, C., Tesini, A., Kakudate, S., Siuko, M., & Neri, C. (2011). The Remote Handling Systems for ITER. *Fusion Engineering and Design*, *89*, 471–477.

Ribeiro, I., Lima, P. U., Aparício, P., & Ferreira, R. (1997). Conceptual Study on Flexible Guidance and Navigation for ITER Remote Handling Transport Casks. In *IEEE/NPSS Symposium on Fusion Engineering* (pp. 969–972).

Roy, N., Gordon, G. J., & Thrun, S. (2003). Planning under Uncertainty for Reliable Health Care Robotics. In *International Conference on Field and Service Robots* (Vol. 24, pp. 417–426).

Sadowska, A., Kostic, D., Wouw, N. van de, Huijberts, H., & Nijmeijer, H. (2012). Distributed Formation Control of Unicycle Robots. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1564–1569).

Sanctis, L. D. (2010). *Determinación de Trayectorias Óptimas Sobre Superficies Tridimensionales Utilizando Fast Marching*. Master's thesis, Carlos III University of Madrid.

Sanhoury, I. M. H., Amin, S. H. M., & Husain, A. R. (2012). Switching between Formations for Multiple Mobile Robots via Synchronous Controller . In *IEEE International Colloquium on Signal Processing and its Applications* (pp. 352–357).

Schaal, S., Peters, J., Nakanishi, J., & Ijspeert, A. (2003). Learning Movement Primitives. In *International Symposium on Robotics Research (ISRR)* (Vol. 15, pp. 561–572).

Serret, J. A. (1851). Sur Quelques Formules Relatives à la Théorie des Courbes à Double Courbure. *Journal de Mathématiques Pures et Appliquées*, *1*(16), 193–207.

Sethian, J. A. (1996). *Level Set Methods and Fast Marching Methods*. Cambridge University Press.

Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press.

Shen, Y. (2009). Location Prediction for Tracking Moving Objects. In *WRI Global Congress on Intelligent Systems* (pp. 362–366).

Shkolnik, A. C., & Tedrake, R. (2009). Path Planning in 1000+ Dimensions Using a Task-space Voronoi Bias. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2061–2067).

Siegwart, R., & Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*. Bradford Company.

Slotine, J., & Li, W. (1991). *Applied Nonlinear Control*. Prentice Hall.

Tadokoro, S., Hayashi, M., Manabe, Y., Nakami, Y., & Takamori, T. (1995). On Motion Planning of Mobile Robots Which Coexist and Cooperate With Humans . In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Vol. 2, pp. 518–523).

Tanner, H. G. (2004). ISS Properties of Non-Holonomic Vehicles. *Systems and Control Letters*, *53*(3-4), 229–235.

Tao, Y., Faloutsos, C., Papadias., D., & Liu, B. (2004). Prediction and Indexing of Moving Objects with Unknown Motion Patterns. In *ACM SIGMOD International Conference on Management of Data* (pp. 611–622).

Tian, Y., & Sarkar, N. (2012). Formation Control of Mobile Robots subject to Wheel Slip. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4553–4558).

Turpin, M., Michael, N., & Kumar, V. (2012). Decentralized Formation Control with Variable Shapes for Aerial Robots. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 23–30).

Urcola, P., & Montano, L. (2009). Cooperative Robot Team Navigation Strategies Based on an Environment Model. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4577–4583).

Vale, A., Valente, F., Fonte, D., Ribeiro, I., & Ferreiro, J. (2012, January). Trajectory Evaluator and Simulator, User Manual [Computer software manual].

Valente, F., Vale, A., Fonte, D., & Ribeiro, I. (2011). Optimized Trajectories of the Transfer Cask System in ITER. *Fusion Engineering and Design*, *86*, 1967–1970.

Valero-Gómez, A., Gómez, J. V., Garrido, S., & Moreno, L. (2012). Fast Marching Methods in Path Planning. *IEEE Robotics and Automation Magazine*(Accepted).

Wang, D. (2009). *A Generic Force Field Method for Robot Real-time Motion Planning and Coordination*. Doctoral dissertation, Faculty of Engineering and Information Technology.

Xu, B., Stilwell, D. J., & Kurdila, A. (2010). A Receding Horizon Controller for Motion Planning in the Presence of Moving Obstacles. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 974–980).

Yang, A., Naeem, W., Irwin, G. W., & Li, K. (2012). Novel Decentralised Formation Control for Unmanned Vehicles. In *IEEE Intelligent Vehicles Symposium* (pp. 13–18).

Yang, K., Li, M., Liu, Y., & Jiang, C. (2010). Multi-points Fast Marching: A novel Method for Road Extraction. In *International Conference on Geoinformatics* (pp. 1–5).

Yatziv, L., Bartesaghi, A., & Sapiro, G. (2005). A Fast O(n) implementation of the Fast Marching Algorithm. *Journal of Computational Physics*, *212*, 393–399.

Zhou, Z., Jian, Y., Wen-Xia, Z., & Jin-Ping, Z. (2012). Virtual-Leader-follower Structure and Finite-time Controller based Cooperative Control of Multiple Autonomous Underwater Vehicles. In *Chinese Control and Decision Conference* (pp. 3670–3675).