

Planning Robot Formations with Fast Marching Square Including Uncertainty Conditions

Javier V. Gómez, Alejandro Lumbier, Santiago Garrido and Luis Moreno

*Robotics Lab., Carlos III University of Madrid, Spain
{jvgomez, alumbier, sgarrido, moreno}@ing.uc3m.es*

Abstract

This paper presents a novel algorithm to solve the robot formation path planning problem working under uncertainty conditions such as errors in the robot's positions, errors when sensing obstacles or walls, etc. The proposed approach provides a solution based on a leader-followers architecture (real or virtual leaders) with a prescribed formation geometry that adapts dynamically to the environment. The algorithm described herein is able to provide safe, collision-free paths, avoiding obstacles and deforming the geometry of the formation when required by environmental conditions (e. g. narrow passages). To obtain a better approach to the problem of robot formation path planning the algorithm proposed includes uncertainties in obstacles' and robots' positions. The algorithm applies the Fast Marching Square (FM²) method to the path planning of mobile robot formations, which has been proved to work fast and efficiently. The FM² method is a path planning method with no local minima that provides smooth and safe trajectories to the robots creating a time function based on the properties of the propagation of the electromagnetic waves and depending on the environment conditions. This method allows to easily include the uncertainty reducing the computational cost significantly. The results presented here show that the proposed algorithm allows the formation to react to both static and dynamic obstacles with an easily changeable behavior.

Keywords: robot formation motion planning, path planning, formation control, Fast Marching, Fast Marching Square, uncertainty

1. Introduction

Due to their wide range of applications (surveillance, cooperative mapping, etc), robot formations have become one of the most interesting 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 main difficulty of robot formations is maintaining a pose (position and orientation) for each individual robot depending on the poses of other robots with a common objective to reach a desired goal. One of the main problems is that the position of the robots 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.

So far, different approaches have been proposed to solve the robot formation control problem. Beard et al [1] classify the different approaches in three main groups: *leader-following*, 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 [2]. These leaders can be real robots or, as proposed in [3], virtual leaders. The second proposed group is *behavioral*, where several behaviors are weighted in order to give a motion plan to each vehicle [4]. 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 [5].

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 [6] or in very populated groups [3]. Other techniques are based on those virtual potentials, such as the inclusion of springs and dampers [7], [8] 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 [9]. These approaches require a method to switch between geometries when the environment demands it [10]. In *dynamic formations*, the geometric structure is can be distorted in the presence of obstacles and environmental conditions [11].

The algorithm proposed herein focuses on a dynamic leader-follower architecture. In a previous paper [6], we tested how robot formations behave under the Voronoi Fast Marching (VFM) method. The results obtained motivated an intensive study on how this method behaves when dealing with uncertainty in the position of robots and obstacles. Furthermore, in this paper the basic planning method is updated to the Fast Marching Square (FM²) method [12]. FM² is introduced in robot formations as an alternative to the VFM method, including new advantages.

The rest of the paper is organized as follows. In Section 2, the FM² algorithm is summarized. Section 3 explains how the FM² is applied to robot formations, proposing a basic algorithm and including variations to this basic algorithm depending on the objective. Results are also shown in this section. Finally, conclusions and future work are pointed out in Section 4.

2. Fast Marching and Fast Marching Square

2.1. Introduction to Fast Marching and Level Sets

In 1996, J. Sethian proposed the Fast Marching (FM) algorithm to approximate the viscosity solution of the Eikonal equation (1) for every position \mathbf{x} . Although FM is applicable to any number of dimensions we focus on the 2D case. Therefore, let's assume a 2D map, where $\mathbf{x} = (x, y)$ is a point on the map with the respective coordinates in relation to a Cartesian referential, the frontwave arrival time function for every point of the map $D(\mathbf{x})$ and the velocity of the wave propagation $W(\mathbf{x})$ in each point \mathbf{x} . Let's also assume that a wave starts propagating at $\mathbf{x}_0 = (x_0, y_0)$ at time $D(\mathbf{x}_0) = 0$ and with velocity $W(\mathbf{x}) \geq 0$. The Eikonal equation allows updating the time of arrival the propagating frontwave $D(\mathbf{x})$ at each point \mathbf{x} of the map, in which the propagation speed depends on the point $W(\mathbf{x})$, according to:

$$|\nabla D(\mathbf{x})|W(\mathbf{x}) = 1 \tag{1}$$

The level set $\{\mathbf{x}/D(\mathbf{x}) = t\}$ of the solution represents the wave front advancing with a velocity $W(\mathbf{x})$ for a certain medium. The resulting function $D(\mathbf{x})$ is a distance function, and if the velocity $W(\mathbf{x})$ is constant, it can be seen as the Euclidean distance function to a set of points from a given one, usually the goal points. If the medium is non-homogeneous and the velocity $W(\mathbf{x})$ is not constant, the function $D(\mathbf{x})$ represents the distance function

measured with the metrics $W(\mathbf{x})$ or the arrival time of the wave front to point \mathbf{x} .

The FM method is used to solve the Eikonal equation and is very similar to the Dijkstra algorithm [13] that finds the shortest paths on graphs, with the difference that FM is applied to continuous media. Using a gradient descent on the distance function $D(\mathbf{x})$, one is able to extract a good approximation of the shortest path (geodesic) in various settings: Euclidean distance with constant $W(\mathbf{x})$ and a weighted Riemannian manifold with varying $W(\mathbf{x})$.

Discretizing the gradient $\nabla D(\mathbf{x})$ according to [14] it is possible to solve the Eikonal equation at each point \mathbf{x} , which corresponds to the row i and column j of a grid map. Simplifying the notation as shown in (2), the equation (1) becomes (3):

$$\begin{aligned} D_1 &= \min(D(i-1, j), D(i+1, j)) \\ D_2 &= \min(D(i, j-1), D(i, j+1)) \end{aligned} \quad (2)$$

$$\left(\frac{D(i, j) - D_1}{\Delta x}\right)^2 + \left(\frac{D(i, j) - D_2}{\Delta y}\right)^2 = \frac{1}{W(i, j)^2} \quad (3)$$

The FM consists on solving equation (3) in which everything is given except $D(i, j)$. This process is iterative, starting at the source point of the wave (or waves) where $D(i_0, j_0) = 0$. The following iteration solves the value $D(i, j)$ for the neighbours of the points solved in the previous iteration. Using as an input a binary grid map, in which velocity $W(i, j) = 0$ (black) means obstacle and $W(i, j) = 1$ (white) means free space, the output of the algorithm is a map of distances as shown in Figure 1. These distances are the time of arrival of the expanding wave at every point of the map.

In the case of more than one wave expanding, the same process is applicable. In this case, there will be as many points with $D(i_0, j_0) = 0$ as waves expanding. When two waves reach each other, the propagation continues as if they were only one wave, as shown in Figure 2.

Finally, in Figure 3 an example of path planning with the FM method is shown.

2.2. The Fast Marching Square Algorithm

The trajectories generated in the original work by Sethian [15] (see Figure 3) on the FM method are optimal according to the minimal Euclidean distance criterion, but it creates paths which run too close to obstacles and

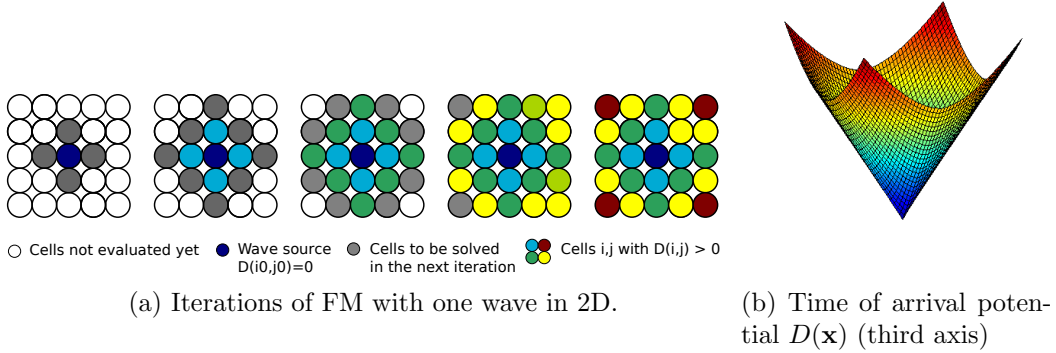


Figure 1: Fast Marching Method with one propagating wave.

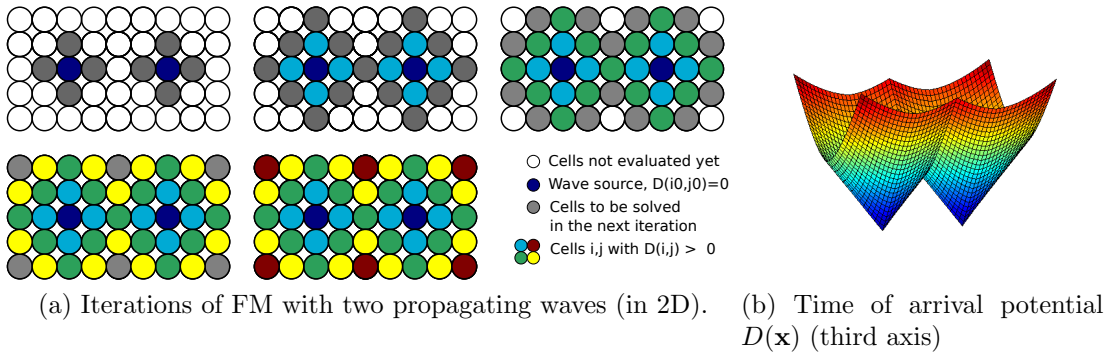


Figure 2: Fast Marching Method with two waves.

are not smooth. These facts turn FM 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.

The FM^2 algorithm can be summarized in the following steps:

1. **Modelling.** The sensory information is included directly in black and white cells avoiding complex modelling or information fusion.
2. **Object enlarging.** The objects detected in the previous step are enlarged by the radius of the mobile robot. This way the objects and walls are dilated to ensure that the robot does not collide or accept passages narrower than its size.

3. **FM 1st step.** Using the map obtained after the enlarging, a wave is propagated from all the points which represent obstacles and walls (black, velocity value 0). This wave propagation is performed using the FM method. The result of this step is a potential map, $W(\mathbf{x})$, in which the value for each pixel is proportional to the distance to the closest obstacle.

This potential map is represented in a gray scale, in which black (velocity value 0) represents walls and obstacles. When the cells are further from them, they become lighter (velocity tends to value 1), representing the *safeness* of the cells. This map can be interpreted as a velocities map or a refraction index map because of the similar effect in Geometrical Optics, where light rays travel in curved trajectories in media with changing refraction index. Due to this analogy, the laws that govern the transmission of electromagnetic waves and light are used to calculate the robot's trajectories.

4. **FM 2nd step.** The Fast Marching Method is applied again on the velocities map obtained in the previous step. A new potential $D(\mathbf{x})$ appears, representing the propagation of an electromagnetic wave, where the time of arrival is added as the third axis in 2D (or the fourth in 3D). The origin of the wave is the goal point and it propagates until it reaches the current position of the robot. Then, gradient descent is applied from the current position the robot. The trajectory obtained is the geodesic in the potential $D(\mathbf{x})$. Since there is only one wave in this step, this potential $D(\mathbf{x})$ will have only one local minimum, located at the goal point.

If the robot, when following the path, is given a reference velocity according to the velocities map, then the path is optimal in terms of time according to the metric defined in the previous step. In referece to the analogy with Geometric Optics, time optimality is justified by the principle of Fermat: **'Light travels the path which takes least time'**.

The results from applying the steps of this algorithm are shown in Figure 4 for a real map obtained using a laser range scanner.

The FM² method can be included in the *sensor-based global planner* paradigm. It dose not have the typical problems of these methods [16]: trap situations due to local minima, no passage between closely spaced obstacles, and oscillations in presence of obstacles or narrow corridors. This

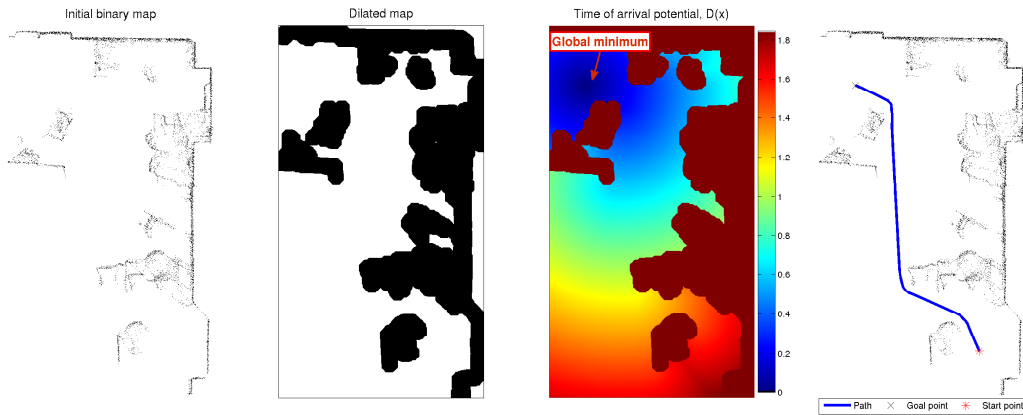


Figure 3: Path planning with the standard FM method. From left to right - Initial binary map obtained by a range scanner sensor. Dilated binary map. Output of the FM method, time of arrival function $D(\mathbf{x})$. Path obtained after applying gradient descent on $D(\mathbf{x})$.

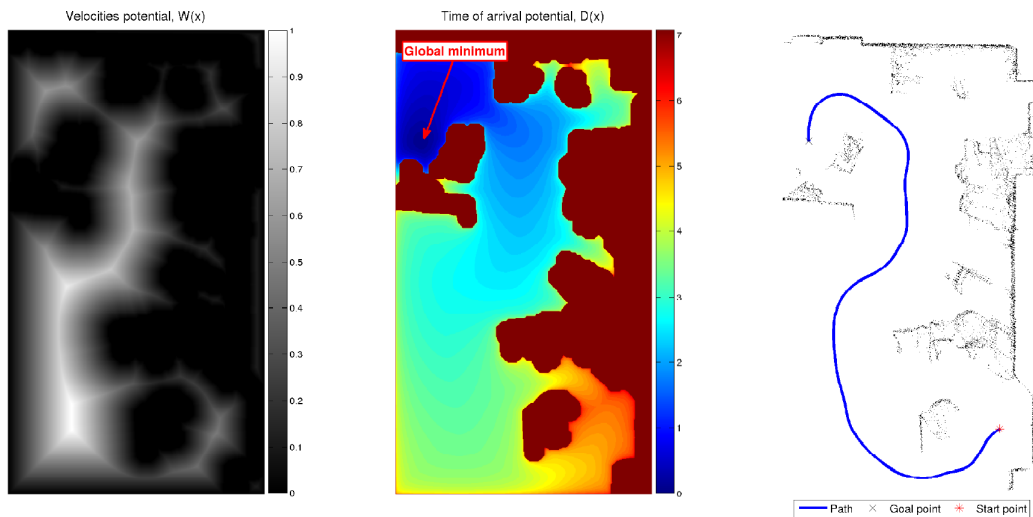


Figure 4: Steps of the FM² method. From left to right: velocities potential applied over the dilated map of Figure 3. Time of arrival function $D(\mathbf{x})$, it is possible to appreciate how the wave expands in a different way than in 3. Finally, path obtained applying gradient descent over the $D(\mathbf{x})$ potential: smooth and safe.

method seeks trajectories with adequate properties (smoothness, continuous curvature, etc.) and it is conceptually close to the navigation functions of Rimón-Koditschek [17], because the potential field has only one local minimum located at the goal point. Concretely, the key characteristics of the FM² method are:

- *Absence of local minima.* Expanding only one wave when computing the second potential $D(\mathbf{x})$ assures that there are no local minima. Since the velocities of the first potential are always non negative (independently on the type of environment, obstacles, etc) it is impossible to have local minima. In the worst case a saddle point can occur, but this is not problem in the gradient descent step.
- *Fast response.* The simple treatment of sensor information and the low complexity order of the algorithm allows a good response in terms of computation time.
- *Smooth trajectories.* As long as the velocities map does not have discontinuities, the trajectories provided will be smooth and do not need to be refined.
- *Reliable trajectories.* The planner provides safe and reliable trajectories avoiding the problem of coordination between local collision avoidance and global planners.
- *Completeness.* As the method consists in the propagation of a wave, it will always find a path from the initial position to the goal position, if a solution exists.

3. Robot Formation Path Planning with Fast Marching Square

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.

In this paper, 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 [6] to the FM² path planner method. This change is motivated by the fact that FM² 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 FM²: 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 FM² 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 FM², the robots of the formation could crash into each other. Thus, integrating the potential given by FM² 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.

3.1. Base Algorithm

The FM² uses a two-step potential to compute the path: the first step creates a potential which can be interpreted as a velocities potential, 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 FM² 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 FM method to the binary map \mathbf{W}_0 , according to the FM 1st step of the FM² method (section 2.2).
- The second potential \mathbf{D} is calculated applying the FM method on the potential \mathbf{W} .

- An initial path for the leader is calculated applying gradient descent on the potential \mathbf{D} , according to the FM² method.
- So far the algorithm described is the application of FM² 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 $\mathbf{W}_{\mathbf{o}i}^t$ 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 $\mathbf{W}_{\mathbf{o}i}^t$.
 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 5.
 4. The potentials \mathbf{D}_i^t are calculated applying the FM method 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 FM² 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 6. It is a base which assures the correct navigation of a robot formation through different environments, avoiding obstacles and adapting to narrow passages. In [6] many additional techniques are proposed to improve the time or behaviour performance of the algorithm. These techniques such as maximum

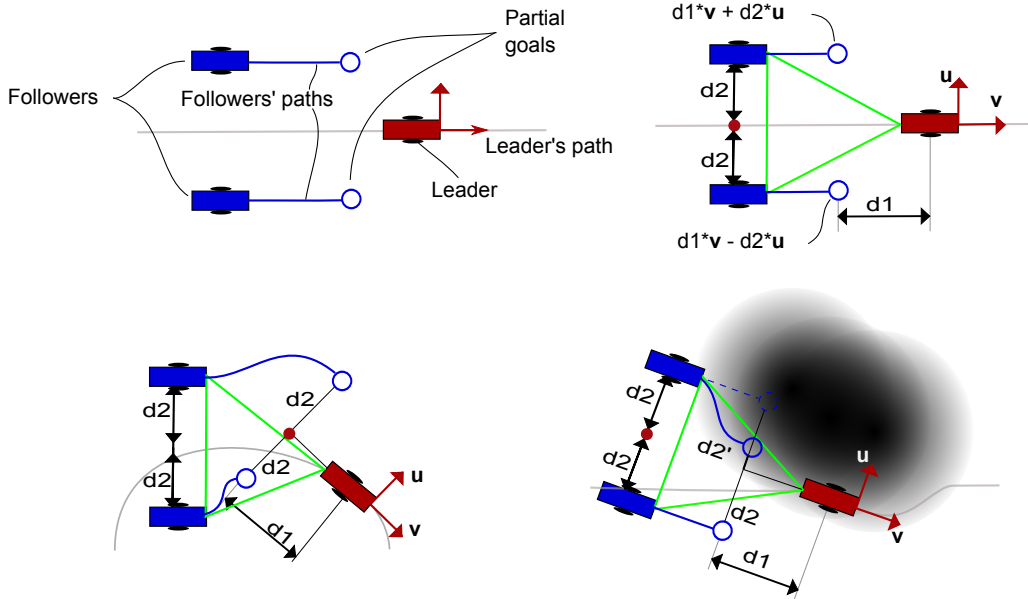


Figure 5: Top left - Main components of the robot formation algorithm. Top right - Reference geometric definition of a simple, triangle-shaped robot formation. Bottom left - Behaviour of the partial goals depending on the leader's pose. Bottom right - Behaviour of the partial goals depending on the obstacles of the environment.

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 7 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. We will employ this type of robot formation throughout the paper, but some experiments will also be shown, featuring more robots in the formation and with different shapes. In the Figure 8 the complete sequence of movement is shown.

3.2. Including uncertainty conditions

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

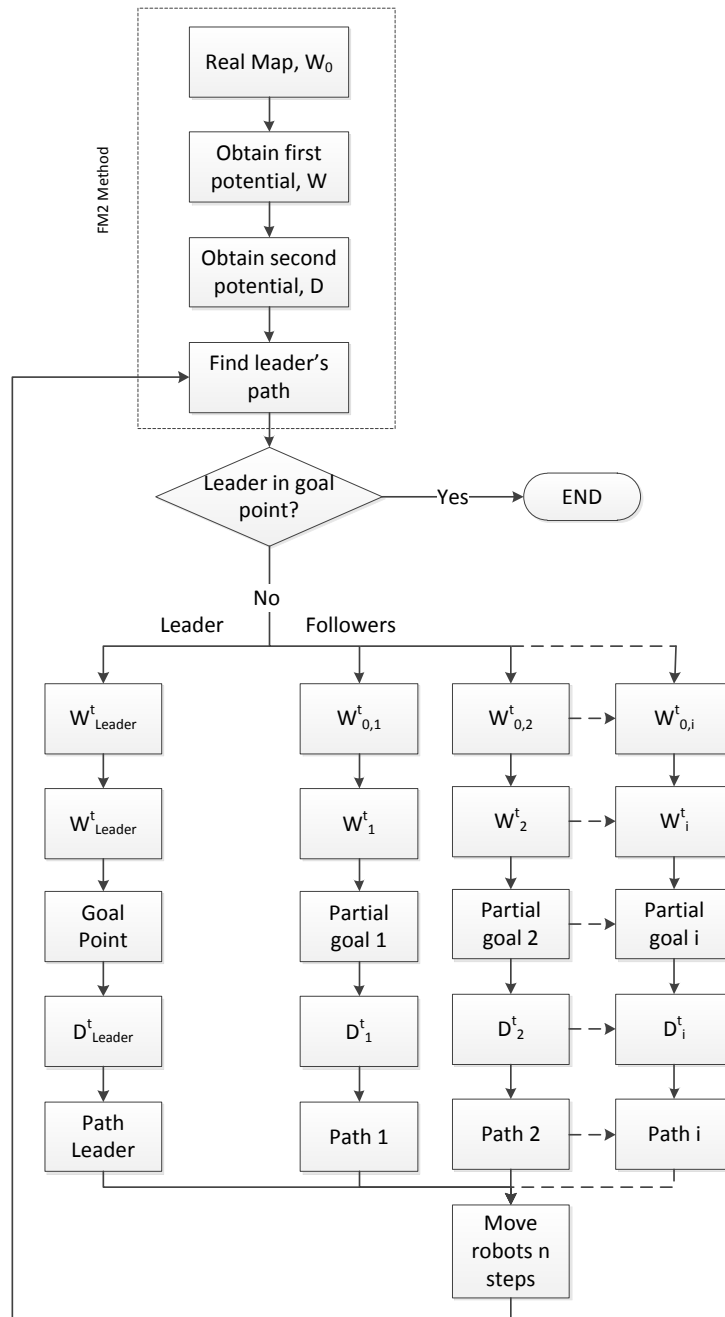


Figure 6: Flowchart of the basic robot formation planning algorithm using FM².

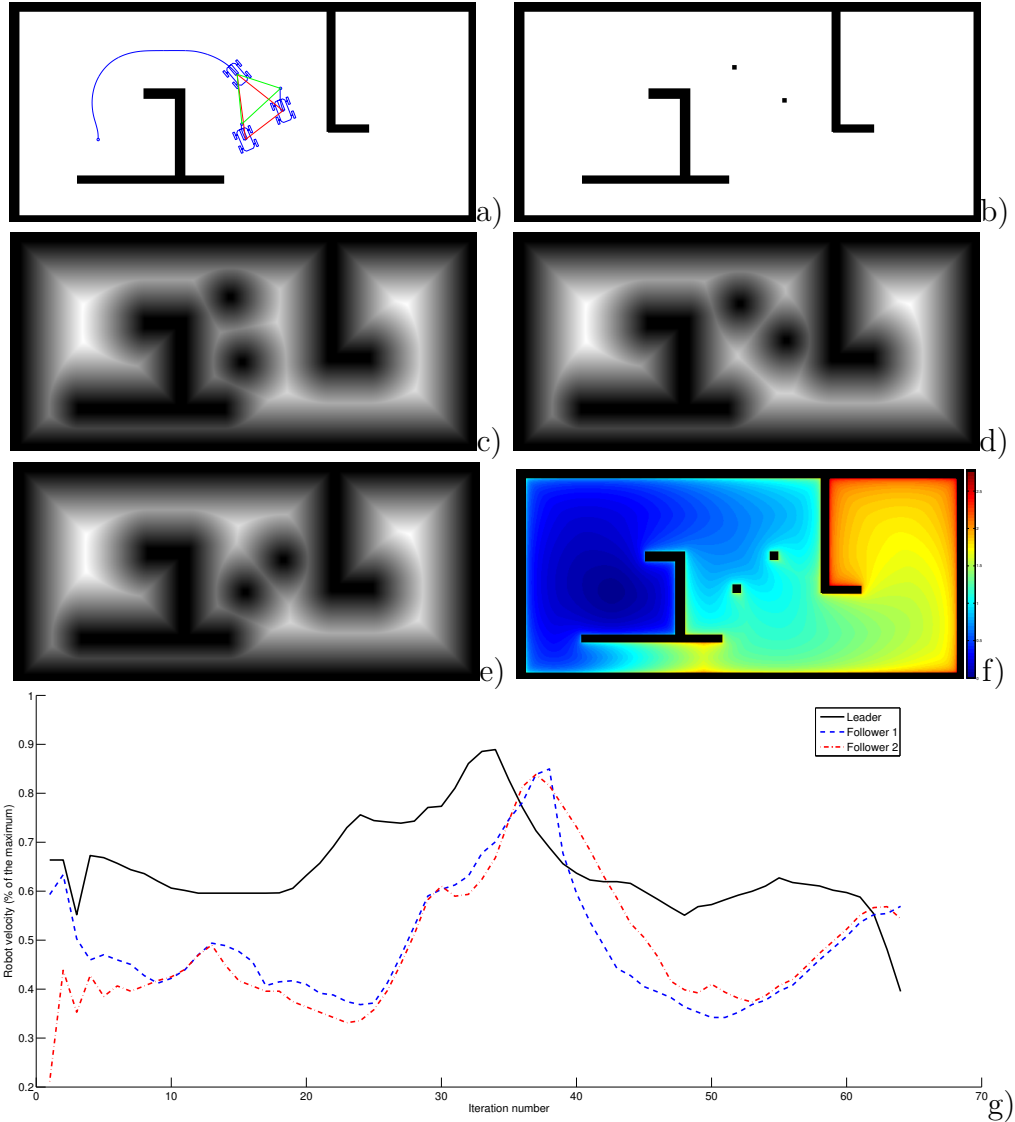


Figure 7: a) Snapshot of the formation moving. The leader follows the blue path. The green triangle (leader-partial goals) is the desired formation and the red triangle (leader-followers) is the current formation. b) How the follower 2 sees the other 2 robots in the binary map. c) First potential, \mathbf{W}_1^t , for the follower 1. d) \mathbf{W}_2^t . e) \mathbf{W}_{leader}^t . f) Second potential, \mathbf{D}_{leader}^t , for the leader. g) Reference velocities for each robot of the formation.

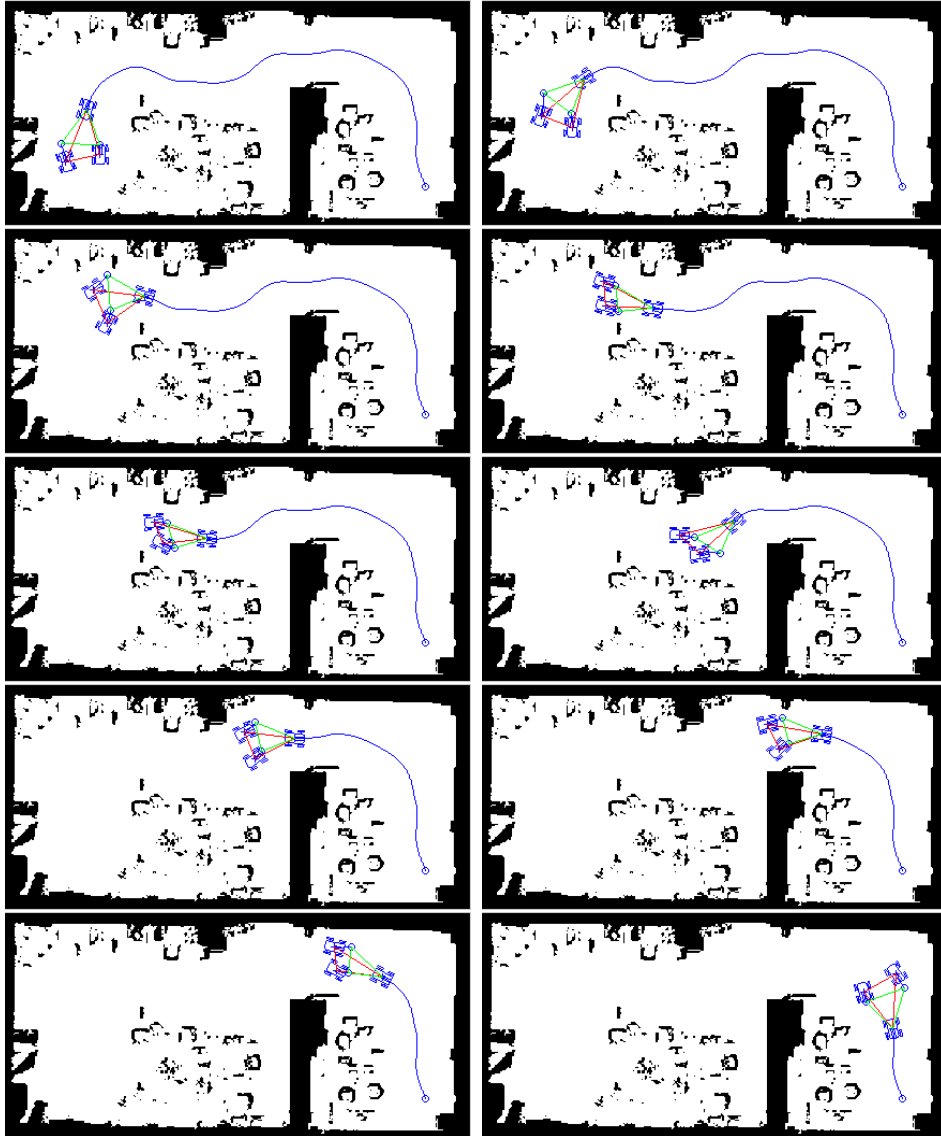


Figure 8: Sequence of movement of a robot formation with the basic path planning algorithm, simulated in a map of our laboratory obtained with SLAM techniques.

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 9. 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:

- 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 means totally uncertain and 0 means no uncertainty).
- In the middle of this map a zone with value 0 is included. The size of this black zone is equal to the dimensions $l_i \times w_i$, representing the robot on the measured position.
- The FM algorithm 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

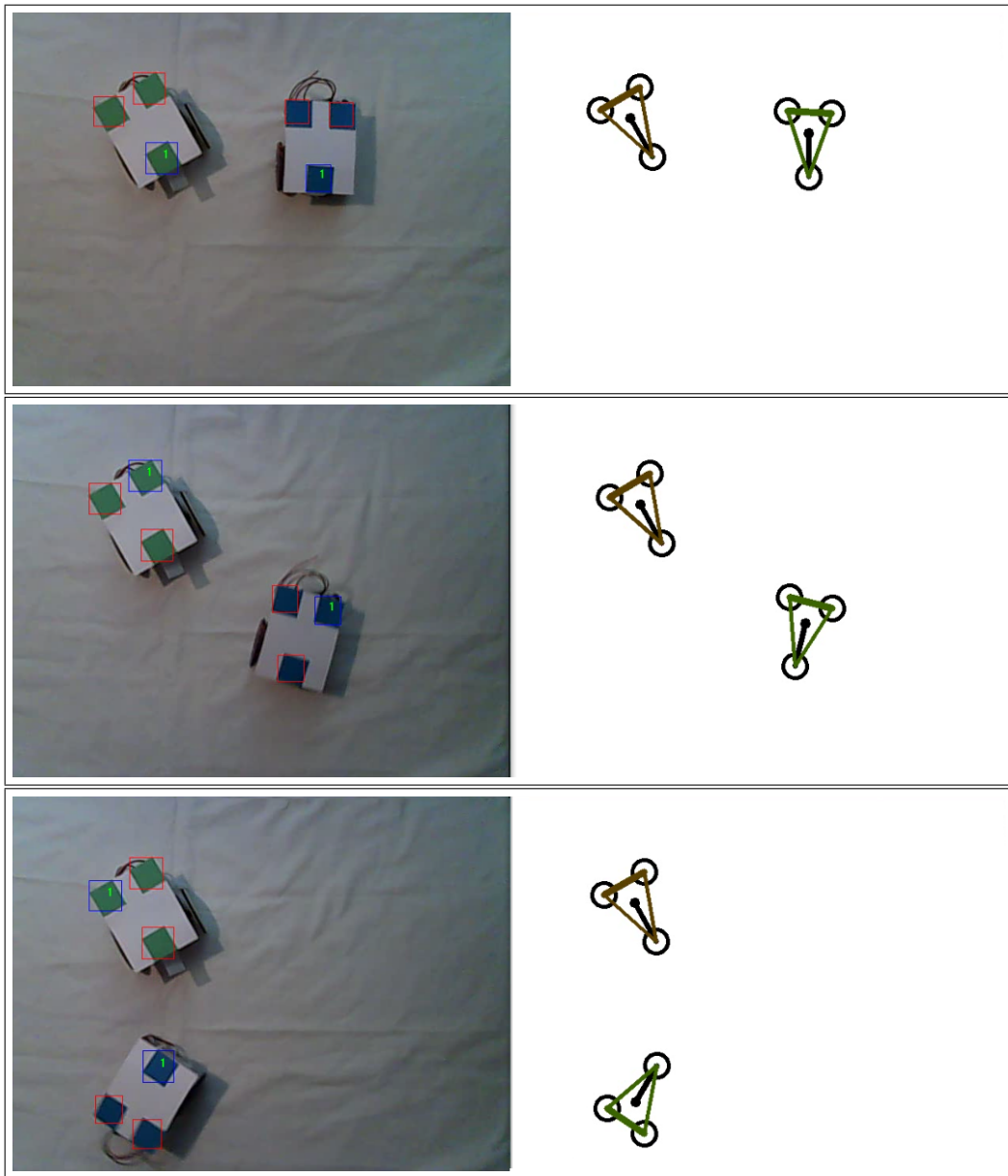


Figure 9: Example of a system which is able to capture the position and orientation of the robots by using a camera and colour labels on the robots.

uncertainty function $\mathbf{W}\mathbf{r}_i$ 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 $\mathbf{W}\mathbf{r}_i$. Thus, \mathbf{W}_j^t is updated with the position of robot i with the uncertainty included.

$$\mathbf{W}_j^t = \min(\mathbf{W}_j^t, \mathbf{W}\mathbf{r}_i)$$

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 3.1 by including it in place of steps 1 and 2 of the loop.

With this method, summarized in Figure 10, 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 where 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 11. Full sequences of movements are shown in Figures 12, 13, 14, testing different robot formation shapes.

For n robots, the FM method 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 8 and 12, it is possible to see that the motion of the formation is not highly modified. However, the inclusion of the other robots

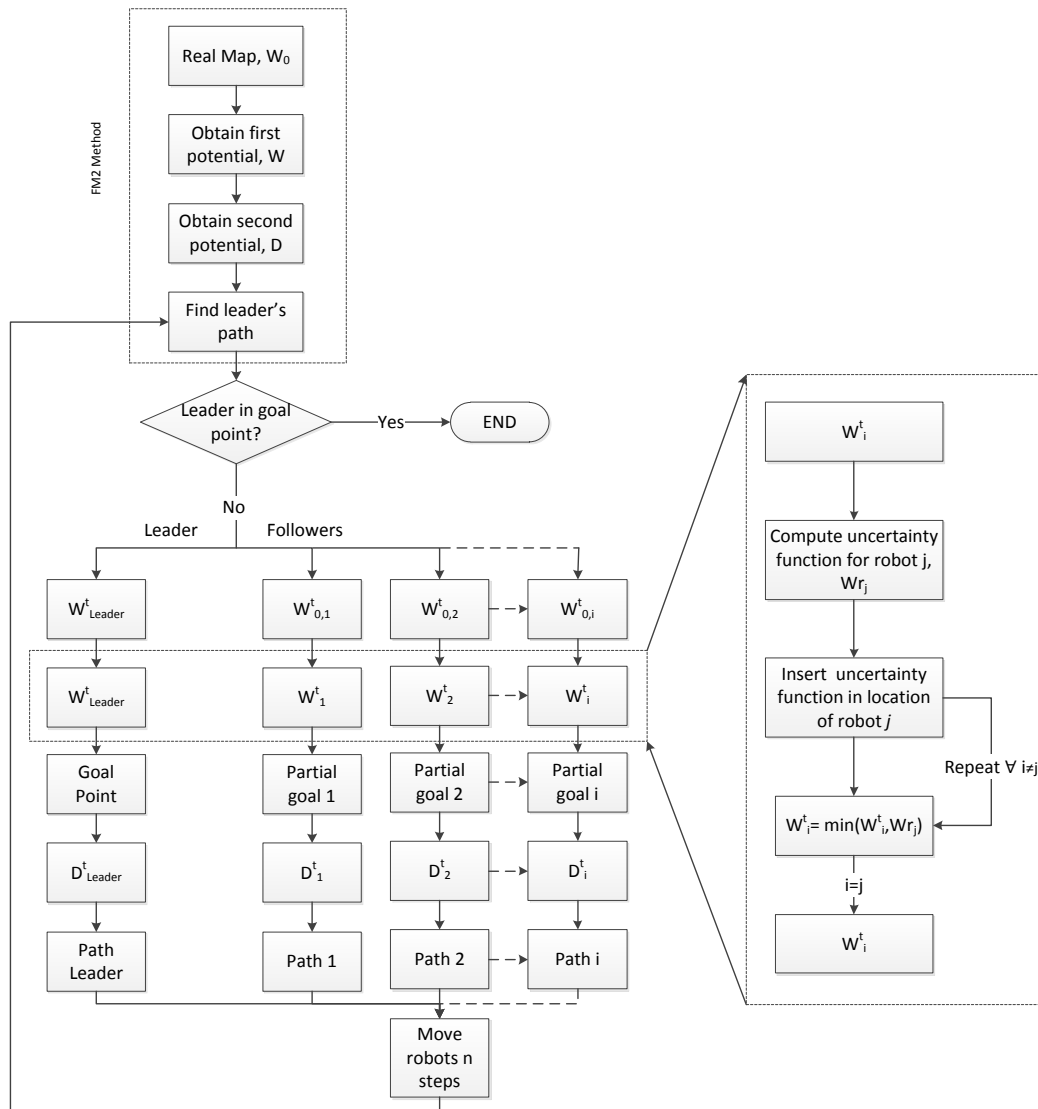


Figure 10: Flowchart of how the velocities map is modified for every robot in the formation.

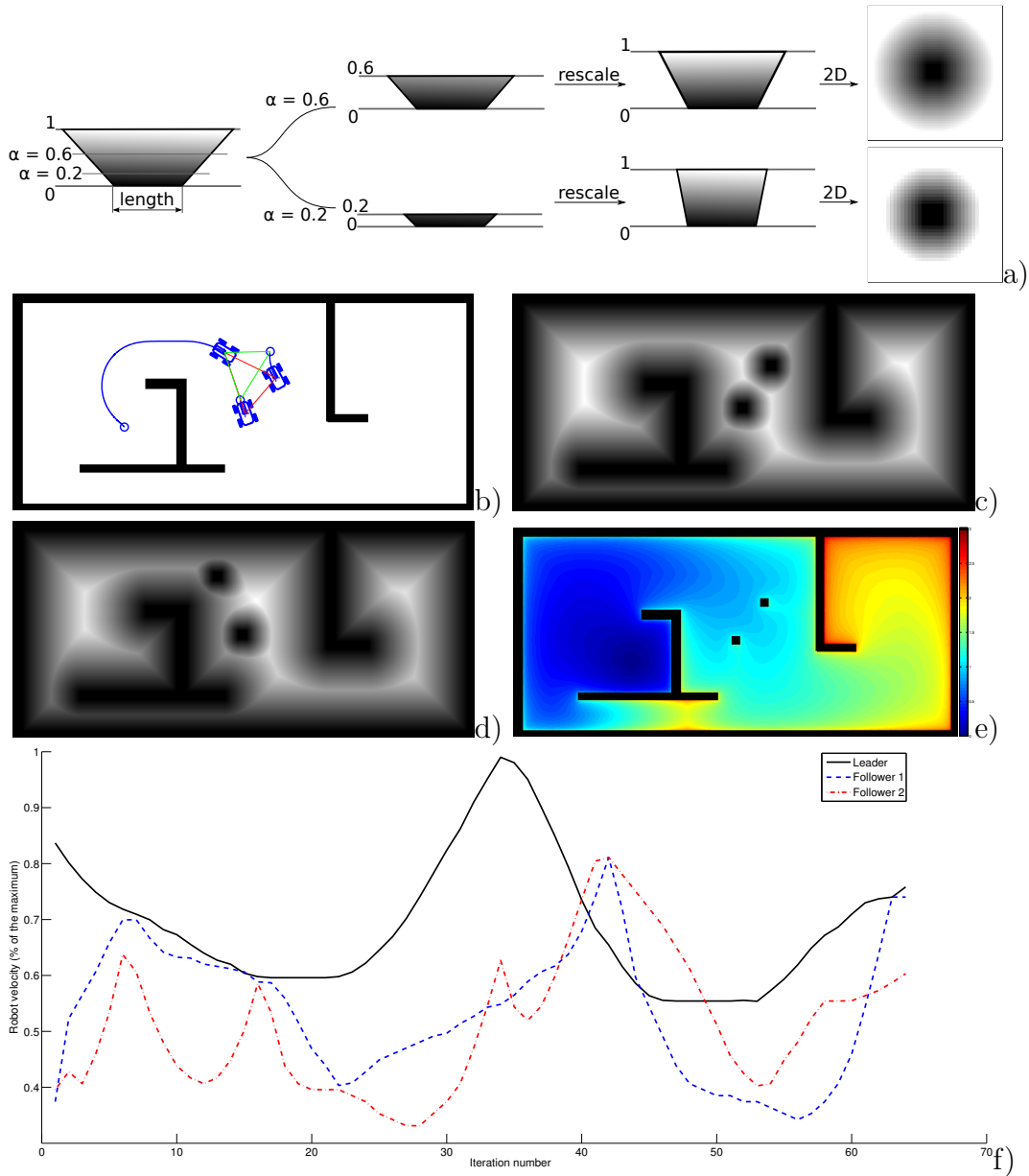


Figure 11: a) Intuitive generation of the uncertainty function $\mathbf{W}r_i$ depending on α for one dimension and its extension to 2 dimensions. b) Current position of the robot formation. c) First potential \mathbf{W}_{leader}^T of the leader at time T, where the other two robots of the formation are taken into account using their uncertainty function. d) First potential \mathbf{W}_1^T , for follower 1. e) Second potential, \mathbf{D}_{leader}^t , for the leader. f) Reference velocities for each robot of the formation.

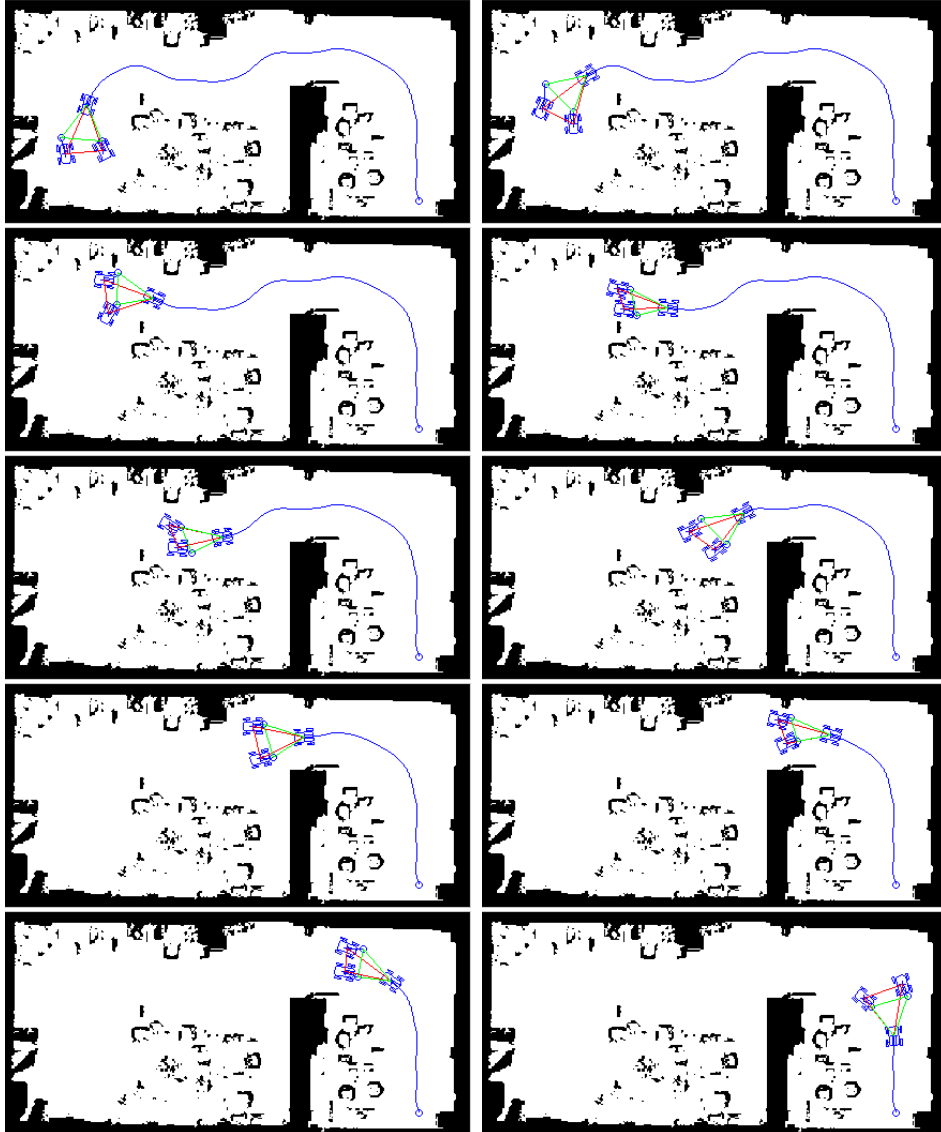


Figure 12: Sequence of movement of a robot formation with the proposed path planning algorithm.

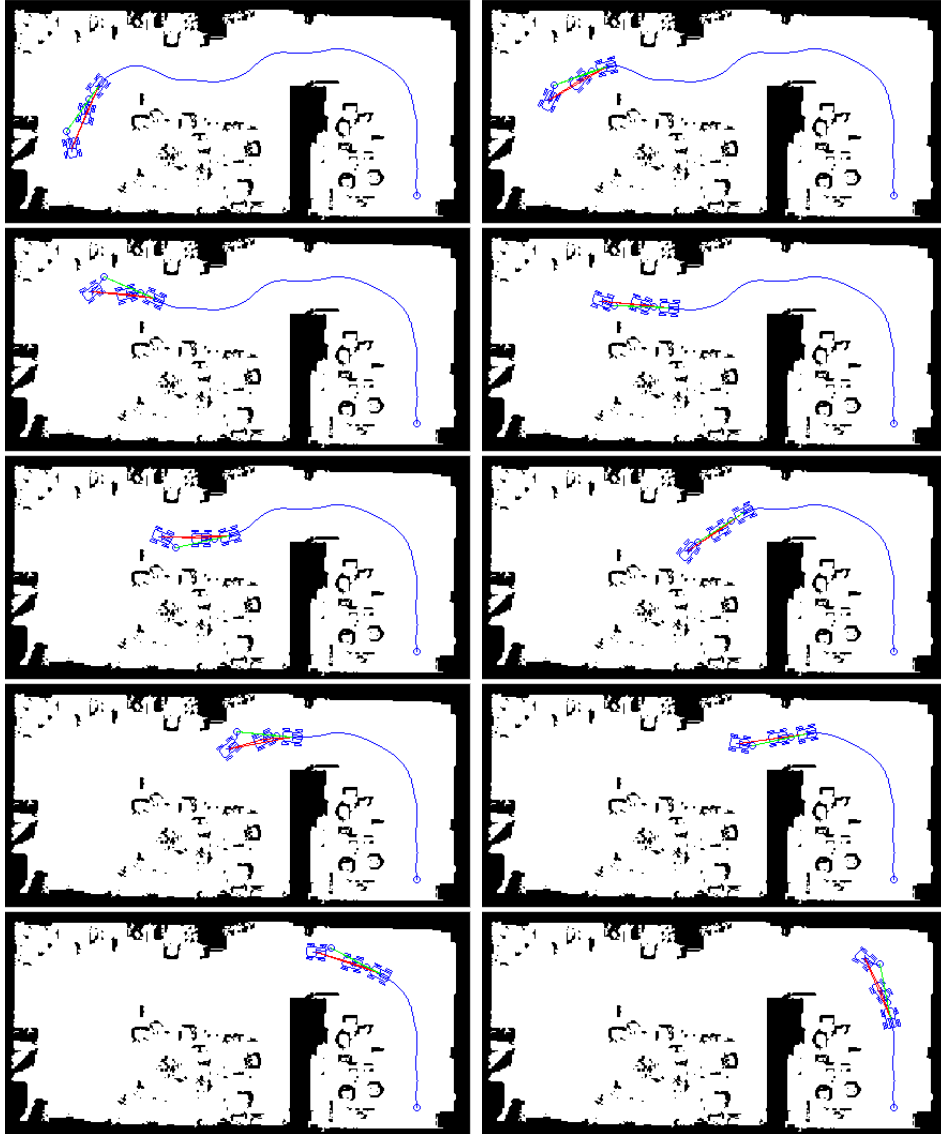


Figure 13: Sequence with a different formation. This time, 3 robots travels in a line.

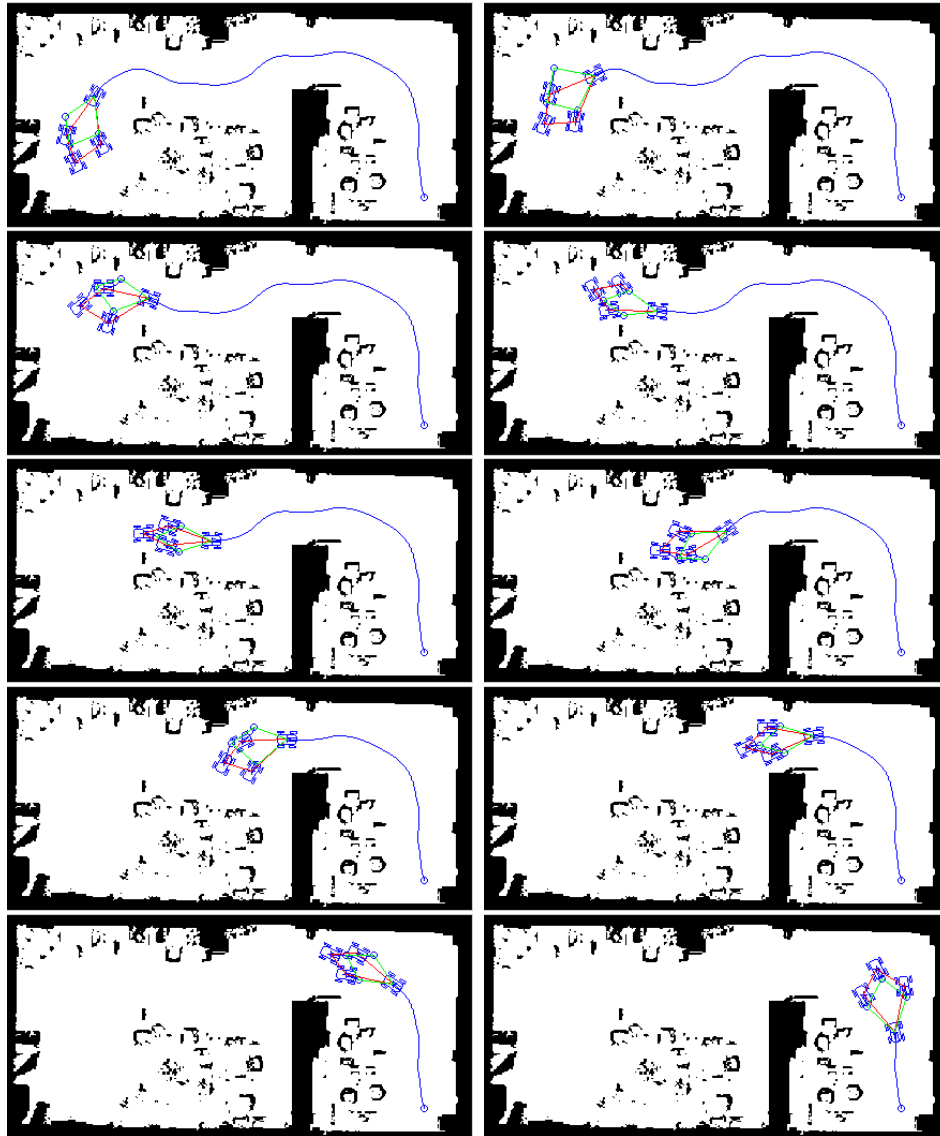


Figure 14: Sequence of movements of a robot formation composed of 4 robots.

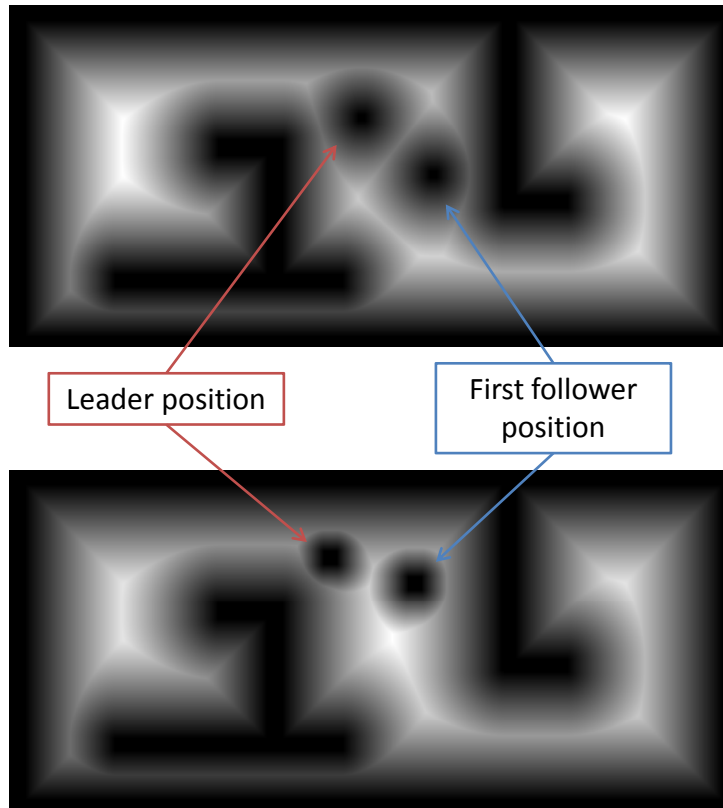


Figure 15: Comparison of the velocities map created for the second follower with the basic algorithm (top) and using uncertainty functions (bottom).

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 15. Therefore, the robots behave normally until they are in places where 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.

3.3. Velocity saturation

In environments with large, open areas the FM² 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 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 [12] 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 16 shows the underlying characteristics of this variation. A full sequence of movement is included in Figure 17. 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.

3.4. 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

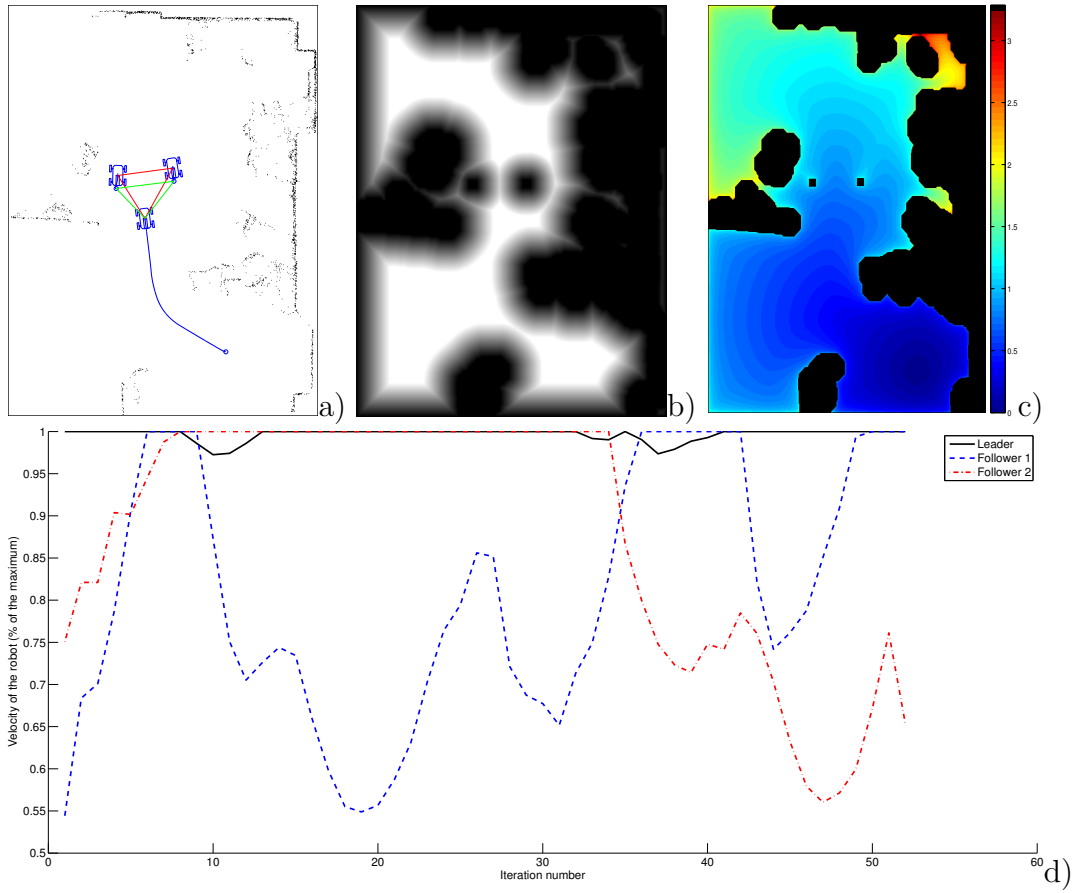


Figure 16: a) Snapshot of the formation moving on a real environment. b) Velocities map of the leader at time T, \mathbf{W}_{leader}^T . c) Second potential of the leader at time T, \mathbf{D}_{leader}^T . d) Reference velocities for the robots.

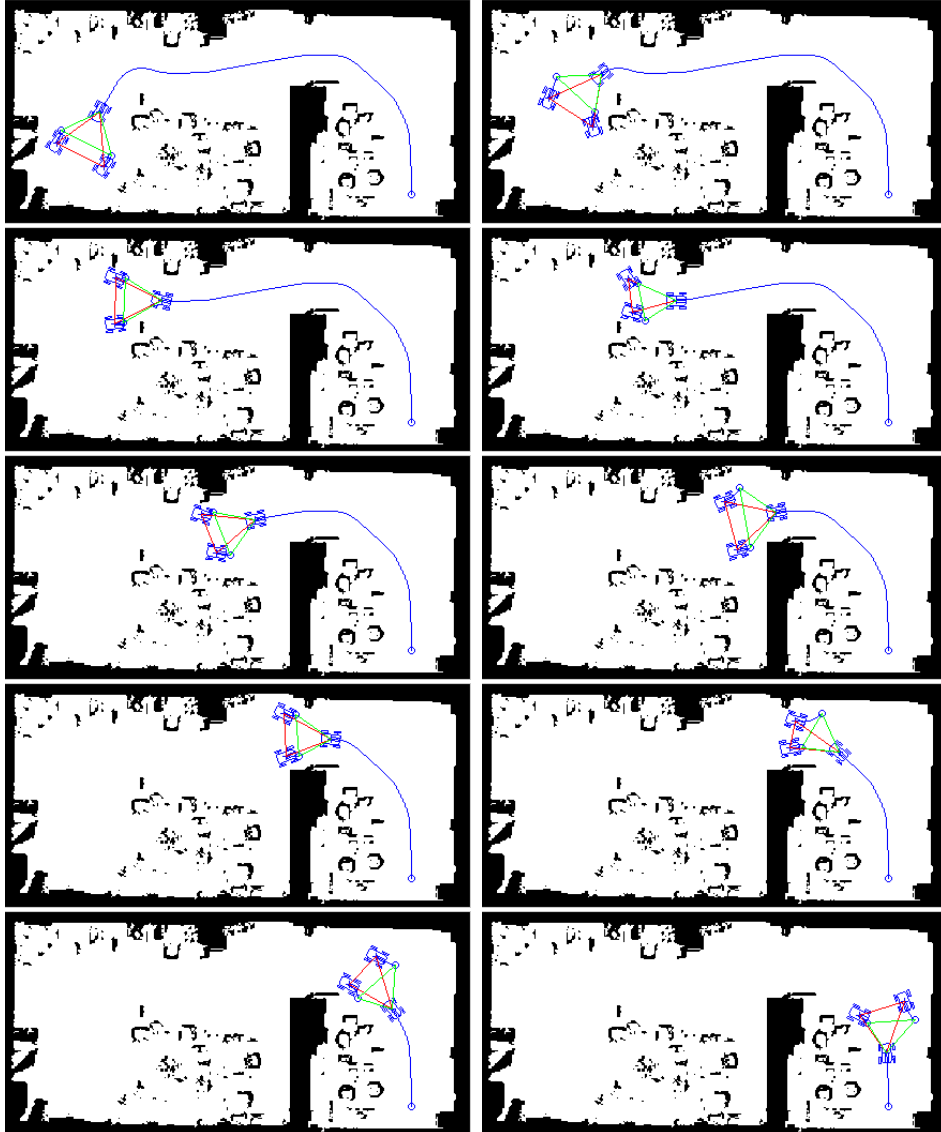


Figure 17: Sequence of movements of a robot formation using velocities map saturation.

noise. It is possible to deal with this uncertainty in the same way as done in section 3.2:

- 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 3.2 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.

With this method a low computational cost is achieved when dealing with dynamic obstacles, since the underlying algorithm is the same proposed in section 3.2. 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 18. A complete sequence of movement in a real laboratory, obtained through a 360° field-of-view range scanner, is shown in Figure 19, where velocity saturation was also applied. Predictive algorithms such as [18, 19], 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 [20]. The main advantage of the proposed method is that using FM² has a similar result and it is easier to implement. Also, the way Gaussian functions can be merged with the FM² requires a deep study while the advantages are not remarkable in comparison with the proposed method.

4. Conclusions and Future Work

All the graphs included, except Figure 9, 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 FM² 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 5)

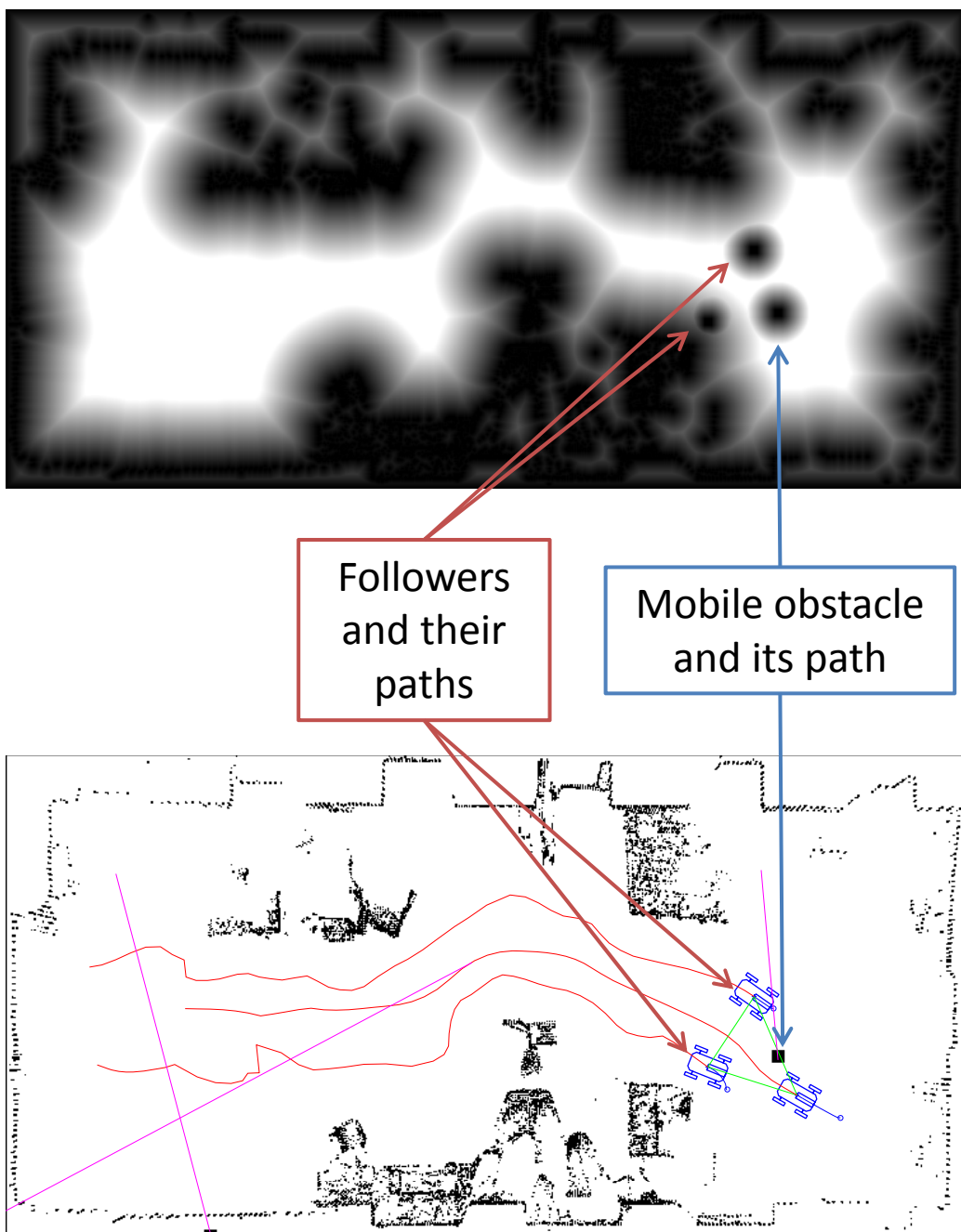


Figure 18: Top - Velocities map of the leader at time T , \mathbf{W}_{leader}^T , with the followers and obstacles included with their uncertainty function. Bottom - Current position of the formation and the obstacle.

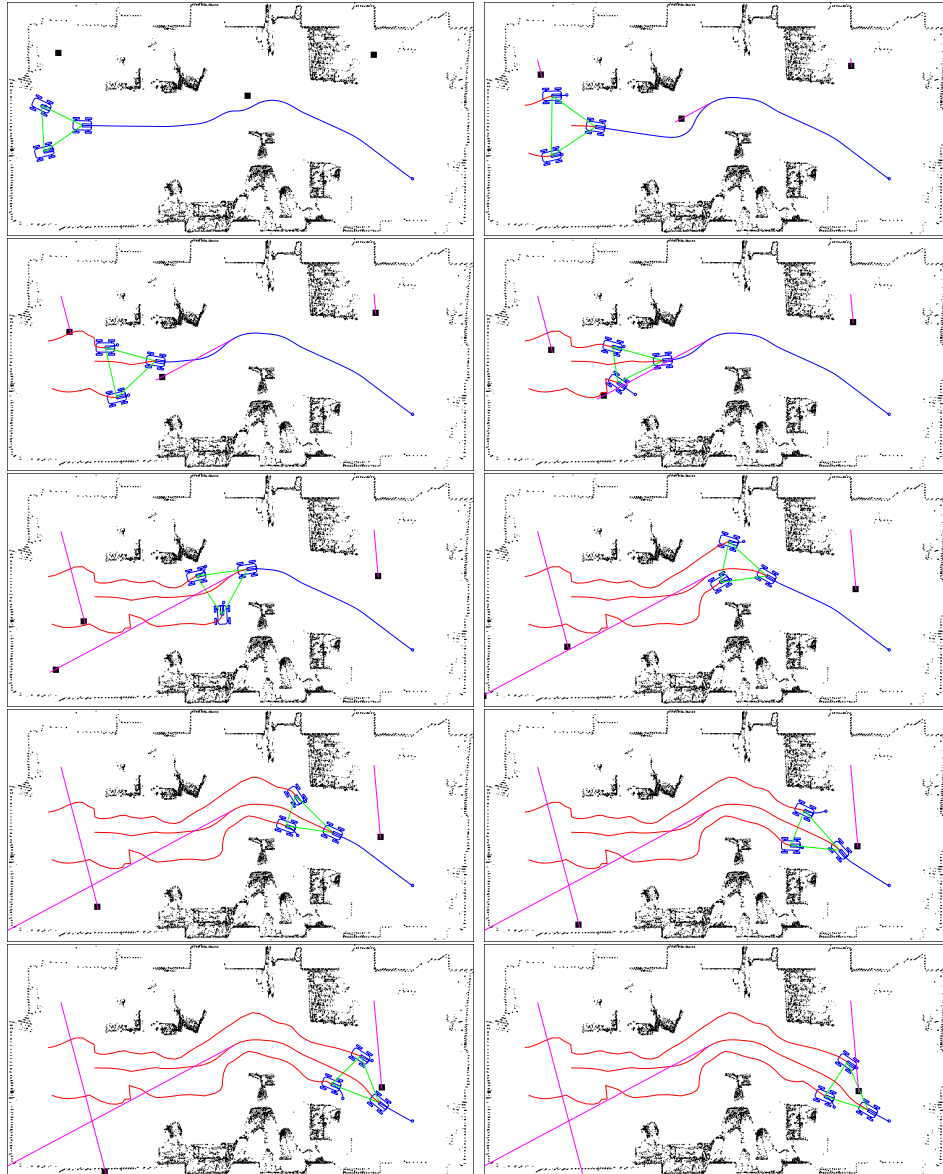


Figure 19: Navigation sequence of a formation in a real environment. The trajectories of the robots (red) and obstacles (pink) are included (the sharp trajectories of the followers are due to simulation discretization).

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 8, 12, 13, 14, 17 and 19 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.

All these tests show that the proposed method, in combination with the FM² 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 [6] 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 approaches. 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 paper represents a novel approach to solve robot formation motion planning which is robust enough to work under uncertainty conditions.

Future work in robot formation using FM² 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).

Future work is also related to expanding the proposed algorithm to outdoor environments, where all the robots of the formation will not be on the same plane (as occurs in 2D); and also to study how to create formations in 3D cases, for example, in unmanned air vehicles (UAVs) flight control.

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

5. Acknowledgements

This work is included in the project number DPI2010-17772 founded by the Spanish Ministry of Science and Innovation. The authors also want to gratefully acknowledge the contribution of Adrian Jiménez Cámara to this paper and its experiments.

References

- [1] R. W. Beard, J. Lawton, F. Y. Hadaegh, A Coordination Architecture for Spacecraft Formation Control, *IEEE Transactions on Control Systems Technology* 9 (2001) 777–90.
- [2] H. G. Tanner, ISS Properties of Non-Holonomic Vehicles, *Systems and Control Letters* 53 (2004) 229–35.
- [3] N. E. Leonard, E. Fiorelli, Virtual Leaders, Artificial Potentials and Coordinated Control of Groups, in: *Proc. of the 40th IEEE Conference on Decision and Control*, pp. 2968–73.
- [4] T. Balch, R. C. Arkin, Behavior-based Formation Control for Multirobot Systems, *IEEE Transactions on Robotics and Automation* 14 (1998) 926–39.
- [5] M. Egerstedt, X. Hu, Formation Constrained Multi-agent Control, *IEEE Transactions on Robotics and Automation* 17 (2001) 947–51.
- [6] S. Garrido, L. Moreno, P. U. Lima, Robot Formations Motion Planning using Fast Marching, *Robotics and Autonomous Systems* 59 (2011) 675–83.
- [7] P. Urcola, L. Montano, Cooperative Robot Team Navigation Strategies Based on an Environment Model, *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2009) 4577–83.
- [8] E. Z. MacArthur, C. D. Crane, Compliant Formation Control of a Multi-Vehicle System, in: *Proc. of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 479–84.
- [9] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, C. J. Taylor, A Vision-Based Formation Control Framework, *IEEE Transactions on Robotics and Automation* 18 (2002) 813–25.

- [10] R. Fierro, P. Song, A. Das, V. Kumar, Cooperative Control of Robot Formations, in: R. Murphey, P. Pardalos (Eds.), Cooperative Control and Optimization, Kluwer Academic Press, Hingham, MA, 2002.
- [11] P. Ogren, E. Fiorelli, N. E. Leonard, Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment, *IEEE Transactions on Automatic Control* 49 (2003) 1292–302.
- [12] S. Garrido, L. Moreno, M. Abderrahim, D. Blanco, FM2: A Real-time Sensor-based Feedback Controller for Mobile Robots, *International Journal of Robotics and Automation* 24 (2009) 3169–92.
- [13] E. W. Dijkstra, A Note on Two Problems in Connexion With Graphs, *Numerische Mathematik* 1 (1959) 269–71.
- [14] S. Osher, J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, *Journal of Computational Physics* (1988) 12–49.
- [15] J. Sethian, *Level Set Methods*, Cambridge University Press, 1996.
- [16] Y. Koren, J. Borenstein, Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation, in: Proc. of the IEEE International Conference on Robotics and Automation, pp. 1398–404.
- [17] E. Rimon, D. Koditschek, Exact Robot Navigation Using Artificial Potential Functions, *IEEE Transactions on Robotics and Automation* 8 (1992) 501–18.
- [18] Y. Tao, C. Faloutsos, D. Papadias, B. Liu, Prediction and Indexing of Moving Objects with Unknown Motion Patterns, ACM Press, New York, NY, USA, 2004, pp. 611–22.
- [19] Y. Shen, Location Prediction for Tracking Moving Objects, 2009 WRI Global Congress on Intelligent Systems (2009) 362–6.
- [20] D. Wang, A Generic Force Field Method for Robot Real-time Motion Planning and Coordination, Ph.D. thesis, Faculty of Engineering and Information Technology, 2009.