Álvarez, David; Gómez, Javier V; Garrido, Santiago; Moreno, Luis (2015) 3D Robot Formations Path Planning with Fast Marching Square. *Journal of Intelligent and Robotic Systems*, 80(3-4), pp.: 507–523.

DOI: https://doi.org/10.1007/s10846-015-0187-1

Journal of Intelligent and Robotic Systems manuscript No.
(will be inserted by the editor)

3D Robot Formations Path Planning with Fast Marching Square

David Álvarez · Javier V. Gómez · Santiago Garrido · Luis Moreno

Received: date / Accepted: date

Abstract This work presents a path planning algorithm for 3D robot formations based on the standard Fast Marching Square (FM^2) path planning method. This method is enlarged in order to apply it to robot formations motion planning. The algorithm is based on a leader-followers scheme, which means that the reference pose for the follower robots is defined by geometric equations that place the goal pose of each follower as a function of the leader's pose. Besides, the Frenet-Serret frame is used to control the orientation of the formation. The algorithm presented allows the formation to adapt its shape so that the obstacles are avoided. Additionally, an approach to model mobile obstacles in a 3D environment is described. This model modifies the information used by the FM^2 algorithm in favor of the robots to be able to avoid obstacles. The shape deformation scheme allows to easily change the behaviour of the formation. Finally, simulations are performed in different scenarios and a quantitative analysis of the results has been carried out. The tests show that the proposed shape deformation method, in combination with the FM^2 path planner, is robust enough to manage autonomous movements through an indoor 3D environment.

Keywords Robot Formations Motion Planning · Formation Control · Fast Marching Square

1 Introduction

In recent years, research on multi-robot systems (MRS) in 3D environments has increased exponentially due to the price drop of unmanned aerial vehicles (UAV) and the advent of micro-air vehicles (MAV) as a popular robotic test bed. Furthermore, in many tasks the use of multi-agent systems increases in overall mission performance, flexibility and robustness without augmenting the capacity of each UAV unit [1]. Based on these characteristics, typical applications for MRS are: exploration [2],

Robotics Lab, Department of Systems and Automation
University Carlos III of Madrid, Leganés, 28911, Spain
E-mail: {dasanche, jvgomez, sgarrido, moreno}@ing.uc3m.es

search and rescue [3], surveillance [4,5], and many others. In order to achieve a good performance in any of these applications several research topics need to be addressed, such as: modelling and control of such agents [6], collision avoidance [7], mapping and state estimation with such agents [8] or formation control and planning [9].

In formation control, a group of coordinated robots have to perform a specific task trying to keep a certain geometric configuration. The coordination of the robots is one of the essential topics in the field of MRS. When operating in close proximity, limited space or in a collaborative task, the movements of the robots have to be planned and coordinated efficiently. In real world applications, this synchronised navigation requires a computationally fast solution so that travel speed can be maintained.

There are many issues to be considered when designing a controller for mobile robot formation, such as: the stability of the formation, the controllability of the shape patterns or the safety of the movements. All these issues need to be addressed while the formation is moving along different scenarios. Therefore, the configuration of the formation should evolve over time in order to meet the different constraints. This evolution can be quantified using different metrics that characterize it. Only a few approaches have been found in the literature on how to do this measurement, mainly because of its difficulty to be generalised. However, some of the most used criteria for 2D formations can be easily extended to 3D formations. [10,11] use two similar performance metrics for the evaluation of their experiments: average position error, which is the average displacement of each robot from their predefined positions in the formation; percentage of time in formation, that reflects the amount of time in which the robots keep the geometry of the formation. [12,13] use a the formation evaluation criterion called uniform dispersion, which evaluates if the same distance is kept between all neighbour robots with a maximum tolerance of ε_d. In this paper, the average position error of each robot along the path is computed. Moreover, since the environments used in the simulations are quite challenging, an analysis based on the distance of the robots to obstacles is also considered so that the safety of the presented algorithm can be evaluated.

Many strategies can be found in the literature describing how to control the evolution of the formation. In [14] the multi-agent coordination problem is studied under the framework of control Lyapunov functions. The main idea is that every robot has a control Lyapunov function and there exists a global Lyapunov function for the whole formation which is a weighted sum of individual Lyapunov function of each robot. The main drawback is the mathematical complexity needed to obtain satisfactory results. Other works use an approach based on potential fields which are combined in order to get the desired behaviour of the formation [15]. The major problem of these approaches is the existence of local minima. In other behaviour-based approaches [16] each robot has basic primitive actions that generate the desired behaviour in response to sensory input. Possible schemas include collision avoidance and goal seeking. Virtual structure introduced by [17] is defined as a collection of agents that maintain a desired geometric configuration. The algorithm has three main steps: the virtual structure is aligned with the position of the robots, then a trajectory for every agent is obtained, and finally each robot follows its own path. This approach is capable of maintaining a highly precise formation and has mainly been used for satellite formation control [18]. However, due to its high computational complexity

is very difficult to apply it to multi-UAV control. In the case of leader-followers approach, a common scheme is the model predictive controller [19] which was recently introduced for holonomic robots [20]. The major focus of most methods is mainly to maintain the formation based on pre-planned paths and static environment assumption.

In this research a Leader-Followers (LF) approach, which widens the work presented in [21], has been used. In this strategy, a robot (that could be virtual) is designated as the leader of the formation and follows a trajectory towards the goal point. At the same time, follower robots are positioned behind it according to a default geometry shape that can change, within a given range, trying to accommodate to the environment conditions [22–24]. An advantage of this strategy lies on its simple implementation since no feedback loop from followers to the leader is needed. This is due to the leader's behaviour, which can be considered as egoistic, since in the path planning phase the formation itself is not considered, and in the motion phase it also does not take into consideration the followers robots. This approach leads to paths that are optimal for the leader, while they might not be as good for the followers. Although a bigger robot describing the formation could be considered in planning phase, avoiding the path for the formation to go through narrow passages, our approach considers that any path can be accomplished if the deformation scheme is good enough. Besides, this way the algorithm can find a solution even in the presence of very constrained situations. Therefore, our deformation scheme allows the robots to adapt their paths in the situations where their movements are constrained by the environment, which is one of the key points of the algorithm. Another important characteristic is that it depends on the leader's motion, so it is very important to have a very good path planning and tracking because once the leader loses its path, its error is fully propagated to all the followers and both the mission and coordination objectives could fail.

In the presented approach, the leader's path is calculated using the Fast Marching Square (FM^2) path planning method, which ensures obtaining very safe and smooth paths [25]. For the followers, a predefined geometry evolves dynamically, while the leader is covering the path, based on a velocities map calculated as a first step of FM^2. The manner the deformation is produced is based on the algorithm described in [26], which shows an easy way to deal with robot priorities when going through very narrow environments, with a very low mathematical complexity. However, only static scenarios are addressed in the previous work. In this research, a 3D modelling of dynamic obstacles based on a grey scale local map is introduced. Besides, a slight modification of the algorithm is introduced so that all the robots are able to take into account the moving obstacles while covering the path. Finally, a quantitative analysis on: the performance of the algorithm with respect to the amount of deformation needed to avoid the obstacles, and the safety of the solution in terms of distance to obstacles, is presented. Furthermore, qualitative results are shown for different complex scenarios.

The next sections of the paper are organised as follows. Section 2 introduces the Fast Marching Square path planning method. In section 3 the application of FM^2 to the robot formation problem in static and dynamic scenarios is explained. In section 4 simulation results for difference scenarios and a qualitative analysis of the

performance of the algorithm are shown. Finally, in section 5 conclusions and future work are addressed.

2 Fast Marching Square path planning method

2.1 The Fast Marching Method

If we consider a light ray travelling through different materials, the Fermat's principle says that its path is the one that consumes a minimum time. This is specially interesting in our application, because if we have only a source of light waves (goal point for the path), each point in the space is connected with the source with a path that it is parameterised by the time of arrival of the wave. Moreover, considering the set of all the points of the domain with the time as last coordinate, we can create a Lyapunov surface in which the level curves are isochronal and the Fermat's paths are orthogonal to them. This means that it is impossible for the method to have local minima. Graphically, this can be seen in figure 1, which represents the funnel potential of the light wave propagation with a constant refraction index.

From the point of view of path planning, these considerations lead us to think that computing the same path for robots can lead to a high save of time. Mathematically, the propagation of the light is given by the Eikonal equation. In [27] an approximation of the solution for this equation was proposed, the Fast Marching Method (FMM). Let us assume a 2D map, where $\mathbf{x} = (x, y)$ is a point on the map with the coordinates in relation to a Cartesian referential, $D(\mathbf{x})$ is the front wave arrival time function and $W(\mathbf{x})$ is the velocity of the wave propagation. Besides, we assume that a wave starts propagating at time $D = 0$ with velocity W always non-negative. The Eikonal equation (1) defines the time of arrival of the propagating front wave, $D(\mathbf{x})$, at each point $\mathbf{x}$, in which the propagation speed depends on the point, $W(\mathbf{x})$, according to:

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

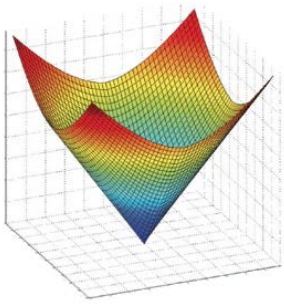

Fig. 1: Lyapunov surface when the propagation wave starts in a point and the refraction index is constant.

Discretizing the gradient ∇D according to [28] it is possible to solve the Eikonal equation at each point $p(x_i, y_j)$, where i and j are the row and column of a grid map, as follows:

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

$$\left(\frac{D_{i,j} - D_1}{\triangle x}\right)^2 + \left(\frac{D_{i,j} - D_2}{\triangle y}\right)^2 = \frac{1}{W_{i,j}^2} \tag{3}$$

The FMM consists on solving $D_{i,j}$ for every point of the map starting at the source point of the wave where $D_{i_0,j_0} = 0$. The following iterations solve the value $D(i, j)$ for the neighbours of the points solved in the previous one. Using as an input a binary grid map, the output of the algorithm is similar to the distance transform, but in this case is continuous, not discrete. These distances have the meaning of the time of arrival of the expanding wave at every point in the map. After applying the FMM, gradient descent can be used from any point of the map of distances to obtain a path towards the source of the wave, which works as a goal point. This is valid only if one wave has been employed to generate the map of distances. The main advantage of this method is that the path obtained is optimal in distance, like the example in figure 2.

2.2 Fast Marching Square Method

As we can see in figure 2, although optimal in distance, it is obvious that the path produced by FMM is not safe in terms of distance to the obstacles, nor feasible in terms of the abruptness of the turns that the path requires. These problems lead us to consider using the FM^2 method as path planner. The FM^2 [29] solves these two main disadvantages. It is based on applying the FMM twice. After the first time it is applied, the resulting map of distances is considered as a map of velocities for the second FMM step. This means that the second time the wave is propagated, the velocity at which it moves can be different at every point in the map. Furthermore,

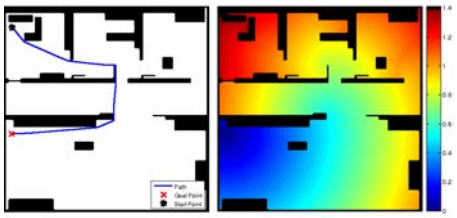

Fig. 2: Example of a path obtained with the FMM. The left side shows the original map and the path calculated. In the right there is the map of distances computed with FMM.

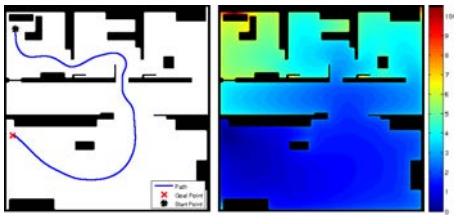

Fig. 3: Example of a path obtained with the FM^2. The left side shows the resulting path of the algorithm in the initial map. In the right side there is the time of arrival map.

this velocity is proportional to the distance to the closest obstacle, meaning that the wave is faster when it is far from obstacles. This produces important differences in the path that is computed, as it can be seen in figure 3.

The proposed FM^2 algorithm has several properties which make it very good for path planning purposes [24, 26]. Most important ones include:

- *No local minima*: as long as only one wave is employed to generate the map of distances, FMM ensures that there is a single global minimum at the source point of the wave (goal of the path).
- *Completeness*: the method finds a path if it exists and notifies in case of no feasible path.
- *Smooth trajectories*: the planner is able to provide a smooth motion plan which can be executed by the robot motion controller. In other words, the plan does not need to be refined.
- *Reliable trajectories*: it provides safe (in terms of distance to obstacles) and reliable trajectory (free from local traps). This avoids the coordination problem between the local collision avoidance controllers and the global planners, when local traps or blocked trajectories exist in the environment.
- *Fast response*: if the environment is static, the map of velocities is calculated only once. Since the FMM can be implemented with a complexity order of $O(n)$ [30], building the map of velocities is a fast process.

2.3 3-Dimensional Fast Marching Square

Since the FM^2 algorithm is based on the standard FMM, it is extensible to more than 2D, as it is done in this work on 3D robot formations planning. The algorithm works exactly in the same way as the 2D version, with the only difference that the front wave becomes a spatial curve. Also, the time response is a little slower since the size of the grid that models the environment is much bigger. Despite this, all the properties of the FM^2 remain in a n-dimensional environment. This is the main fact that leads us to use this algorithm as path planner.

3 Robot Formation planning with FM^2

The algorithm described next is an extension of [31,32], in which the FM^2 path planning method is used to control 2D formation in different scenarios. Although the general idea of the algorithm was presented in [21], the necessary improvements for adapting this method to dynamic environments will be explained.

The algorithm is based on a leader-followers scheme, which is used to control the robot formation. The reference pose for the follower robots is defined by equations that define a geometric shape for the formation. This means that the goal poses along the path of each follower are a function of the leader's pose, as introduced in section 3.1. The leader can be a robot, a person or even a virtual leader. It is important to say that the path for the leader is computed in an egotistic way, not taking into account the other robots in the formation. This might lead to situations where the followers may move too close to obstacles, or even collide with them. These situations are avoided using the shape deformation scheme introduced in 3.2. Both the path planning and the shape deformation are based on FM^2. This algorithm provides a two-level artificial potential field which repels the robots from obstacles and has no local minima. Integrating the potential given by FM^2 and the shape deformation scheme for the follower robots, each robot has at each moment one single potential attracting it into the objective, but repelling it from obstacles and other robots. The main requirement when integrating all the potentials is to do it in a way that does not create local minima. Figure 4 shows the steps of the algorithm on a triangle-shaped robot formation. Although it is a 2D shape, it has been chosen because it is easier to understand the behaviour the followers to avoid colliding with obstacles and among themselves.

3.1 Robot pose coordination

The aforementioned figure 4 shows the basic schema for controlling a robots formation in a 2D scene. However, when the formation performs in a 3D environment, a problem arises when dealing with the orientation of the followers. While in 2D, normal and tangent vectors can be used because they are unique, this cannot be directly applied in 3D, since there exist an infinite number of perpendicular vectors to the tangent to the path. All these vectors are contained in a plane that is perpendicular to the tangent vector, as depicted in figure 5.

In order to be able to use the proposed robot formation planning method in 3D, a third reference has to be specified so that the reference shape of the formation can be determined in a unique form. In this case, a generalization using a relative reference based on the Frenet-Serret formulae [33,34] is proposed. It consists on extracting the local characteristics of the path as a third reference, this way it allows the formation to be environment-independent when defining the reference geometry.

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 parameterization of a continuous, differentiable curve C in a Euclidean space $\mathbb{R}^3$. Let us denote $\mathbf{T}(t)$, $\mathbf{N}(t)$ and $\mathbf{B}(t)$ as the unit tangent vector, the

unit normal vector and the unit binormal vector respectively. Let us denote also the curvature as $\kappa(t)$ and the torsion as $\tau(t)$, the Frenet-Serret formulae are:

$$\mathbf{T}'(t) = \kappa(t)|\mathbf{r}'(t)|\mathbf{N}(t) \tag{4}$$

$$\mathbf{N}'(t) = -\kappa(t)|\mathbf{r}'(t)|\mathbf{T}(t) + \tau(t)|\mathbf{r}'(t)|\mathbf{B}(t) \tag{5}$$

$$\mathbf{B}'(t) = -\tau(t)|\mathbf{r}'(t)|\mathbf{N}(t) \tag{6}$$

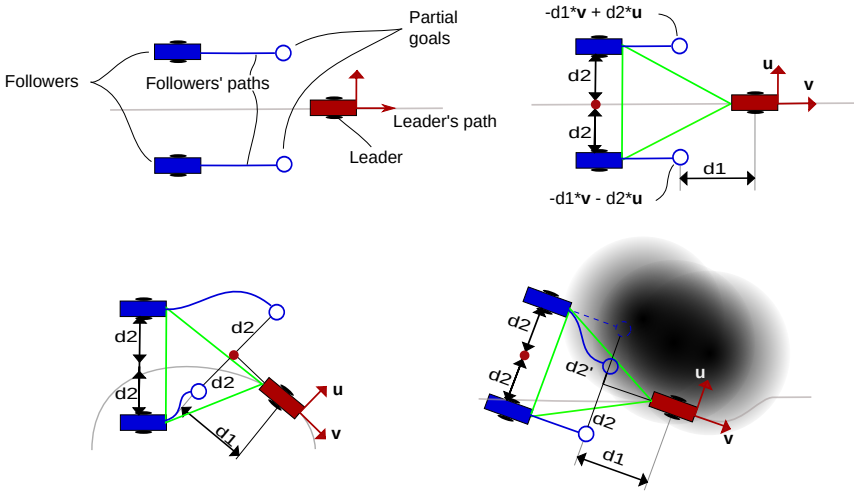

Fig. 4: Top left - Main components of the robot formation algorithm. Top right - Reference geometric definition of a simple, triangle-shaped robot formation, note that the definition is based on vectors **u** and **v** (tangential an perpendicular to the path, respectively). Bottom left - Behaviour of the partial goals depending on the leader's pose. Bottom right - Behaviour of the partial goals depending on the obstacles in the environment.

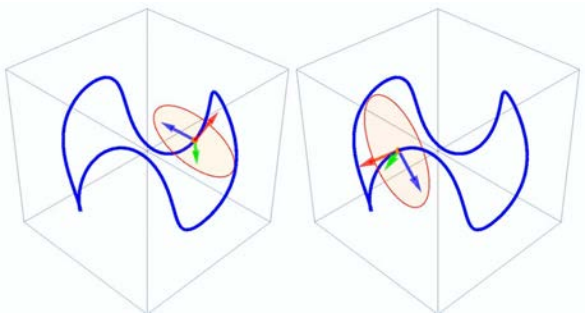

Fig. 5: The red vector represents the tangent to the trajectory and the red circle is the perpendicular plane to this vector.

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

$$\mathbf{T}(t) = \frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|} \; \mathbf{N}(t) = \frac{\mathbf{T}'(t)}{|\mathbf{T}'(t)|} \; \mathbf{B}(t) = \mathbf{T}(t) \times \mathbf{N}(t) \tag{7}$$

The graphical representation of the Frenet trihedron at a certain point of a curve has already been shown in figure 5, where the red vector is $\mathbf{T}(t)$, the blue vector is $\mathbf{N}(t)$ and the green vector represents $\mathbf{B}(t)$. The main advantage of using the Frenet trihedron is that among the infinite possible vectors perpendicular to the tangent vector, the one chosen is in the direction of the curvature **N** (or normal acceleration). Furthermore, the direction of the vectors in the frame change continuously, this is an important property when using it as a reference for the geometry formation, in order to have smooth changes in its shape. So, from a formation control point of view, by combining vectors **T**, **N** and **B** any shape can be given to the formation. One example of a quadrangular pyramid shaped formation is depicted in figure 6.

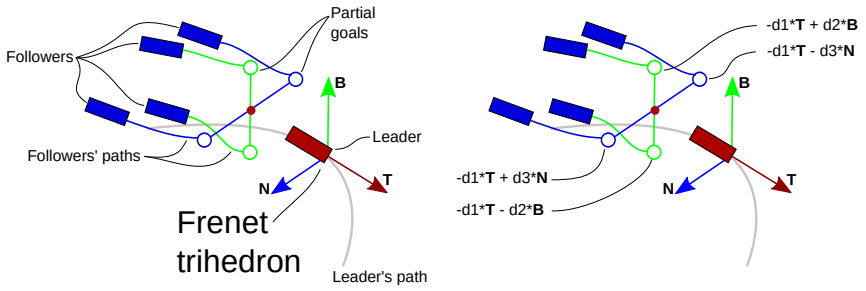

Fig. 6: Schemas of the definition of the geometry in 3D based on the Frenet trihedron. Robots are represented by rectangles, red for the leader and blue for the followers. Attached to the leader, the Frenet trihedron is represented by 3-axes frame in which the tangential vector is drawn in red, the normal vector in blue and the binormal vector in green. The leader's path is represented by a grey line. The follower's paths start from the robot position and go until circles that represent their partial positions. These paths shown as blue and green lines, the difference in colour is only for helping in the visualization. Partial goals for the follower robots are computed with respect to the Frenet trihedron, as indicated in the right part of the figure.

3.2 Grey-level-based shape deformation

As explained before, although the formation is predefined by geometric equations, this shape cannot be exactly the same along all the path because this could lead to collisions with obstacles. Therefore, when the formation gets close to obstacles, the positions of the robots in the formation should be modified in order to adapt the initial shape to the obstacles in the environment. The proposed method achieves this by modifying the distances of the default geometry. The manner these distances are

modified will determine the behaviour of the formation. Figure 7 shows simple deformation rules than can be used. The value *max* represents the distances of the original shape and *min* the minimum distances the formation will deform into. In this case, for the distances in directions parallel to the Frenet trihedron vectors **B** and **N**, namely d_B and d_N, the behaviour is the same: since the less grey level means the closer to obstacles, the distances in these directions have to decrease towards the minimum (which is never reached since it would mean a collision among the followers). This behaviour provokes the follower robots to get closer to the leader's path, which is collision free.

In very narrow environments, this deformation rule can make the robots to suffer an excessive contraction towards the leader's path. Although this contraction is limited, it can still be very dangerous for the robots to move too close to each other. Besides, if the environment is narrow enough, this limit in the contraction could lead to a collision of the followers with the environment. In this case, priorities are introduced in the formation by modifying the distance in the direction parallel to vector **T**, d_T. This behaviour is achieved with a different variation of the distance for each robot. According to figure 7, follower 4 will not modify its distance, while follower 1 will be the one which gets closest to the leader. Therefore, highest priority is given by a higher slope in this function. This deformation rule allows the formation to contract as much as needed so that the obstacles are avoided. The behaviour of the followers can be modified by setting different functions to define the shape changes.

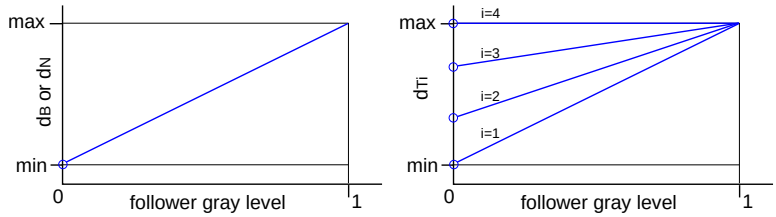

Fig. 7: Followers' partial goals modification based on linear functions.

3.3 Mobile obstacles modelling based on grey level map

In the work presented in [21], only static environments were considered, however, this is not always the case in the real world. In order to take into account mobile obstacles, a grey-level-based model of the obstacles has been developed. This technique deals, not only with the position of the obstacles, but also with their uncertainty which could exist due to, for example, sensor noise.

Let us consider a mobile obstacle O_i with position $P_i = [x_i, y_i, z_i]$ and uncertainty U_i, where i is the number of obstacles and U_i is $0 \leq U_i \leq 1$, being $U_i = 1$ the maximum uncertainty. This uncertainty level models the noise and inaccuracies of the sensors used for obstacle detection and modelling, since it is used to create areas around the objects in which we cannot assure whether there is an obstacle or not.

Note that the aim is not to create an accurate error model for a given sensor, but to create a sensor-independent algorithm. This uncertainty could be understood as a *safety* parameter. Besides, the obstacles can be modelled in different ways, depending on the desired degree of accuracy. In the most simple case, a bounding box [35,36] containing the obstacle is created. Other possibility can be modelling the objects using one or several superquadric functions [37,38]. In any of these cases, a local 3D grid map around each mobile obstacle is created. In this map, we model the obstacle by filling with ones the voxels that correspond to the shape of the chosen model of the obstacle. The accuracy of this model is limited by resolution of the grid. For this example, we will consider a punctual obstacle as can be seen in figure 8, top-left. The rest of the voxels of the local map will have zero values, meaning that they are free locations. Now, if we compute the FMM over this map, we will get the distance transform function, as we can see in figure 8, top-right, in which the locations around the obstacle have a greater grey-scale value than those further from it. This drop of the grey level value can be interpreted as the uncertainty, in which higher values mean a higher likelihood of the voxel to be occupied by the obstacle. Now, this map can be saturated with the value given in U_i to control the level of uncertainty. Then, the values in the map are scaled between 0 and 1. Figure 8, bottom, shows the final step of the local map computation for two different uncertainty values. It can be seen how the uncertainty value U_i allows us to control the area of influence of the obstacle. Finally, the obstacle must be introduced in the environment so that it is taken into account by the leader and the followers in the formation. This is done by applying equation 8 among the voxels of the local map of the obstacle, and the corresponding voxels around its position P_i in the velocities map of the environment. This results into areas around the obstacles that have a velocity value close to zero, which are avoided with the use of the FM^2 path planning method.

$$W_j = \min(W_j, obstacle) \tag{8}$$

where j corresponds to the voxels where the function is applied.

3.4 Formation planning algorithm

In previous sections the low-level modules of the proposed framework have been detailed. Next, these steps are put together into a high-level algorithm, depicted in figure 9. This algorithm assumes that the following information is available at the beginning: a voxelised binary model of the environment; the initial and the goal positions of the leader; the default shape of the formation; the size, uncertainty value, initial position and velocity along the simulation (or the full path) of the mobile obstacles in the environment. It is important to note that a static goal is being considered. A simple replanning approach would not be enough since goal modifications could require the formation do carry out complex maneuvers. Once all the data is provided, it is processed following the steps shown in figure 9, which are explained next:

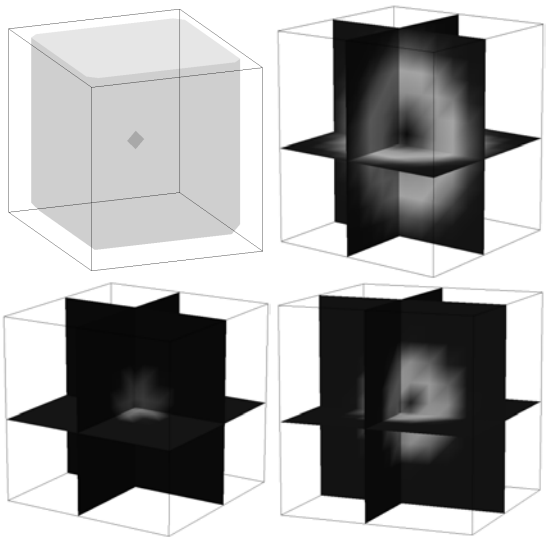

Fig. 8: Process of the computation of the grey-level based model of an obstacle with different uncertainty value. Top left - Local 3D map with a punctual obstacle. Top right - Central slices of the 3D grid representing the distance transform function of the obstacle in the local map. Bottom images: final result after saturation and re-scaling, left $U = 0.3$ and right $U = 0.6$. It is important to note that the furthest positions in the grey scale images should be white, but they have been plotted in black to help the visualization.

(a) The environment map $\mathbf{W_0}$ is read as a binary map, in which 1 (white) means that the position is occupied by an obstacle and 0 (black) means it is free space. This map is common for all the robots in the formation (both the leader and followers).
(b) 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. This potential is also shared by all the robots.
(c) The second potential $\mathbf{D}$ is calculated applying the FMM on the potential $\mathbf{W}$ from the goal point to the leader's initial position.
(d) A path for the leader is calculated applying gradient descent on the potential $\mathbf{D}$, according to the FM^2 method. In the case of static environments, this path never changes.
(e) Once we have a path for the leader of the formation, a loop covering this path is executed. In this loop, each cycle represents a step of the robots' movement. It consists of the next steps:

    I Since we have mobile obstacles, for each cycle t the position of every obstacle is updated both in the binary map $\mathbf{W_0}$, for visualization purposes, and in the first potential $\mathbf{W}$, so that they are taken into account by the robots in the formation, as explained in section 3.3.

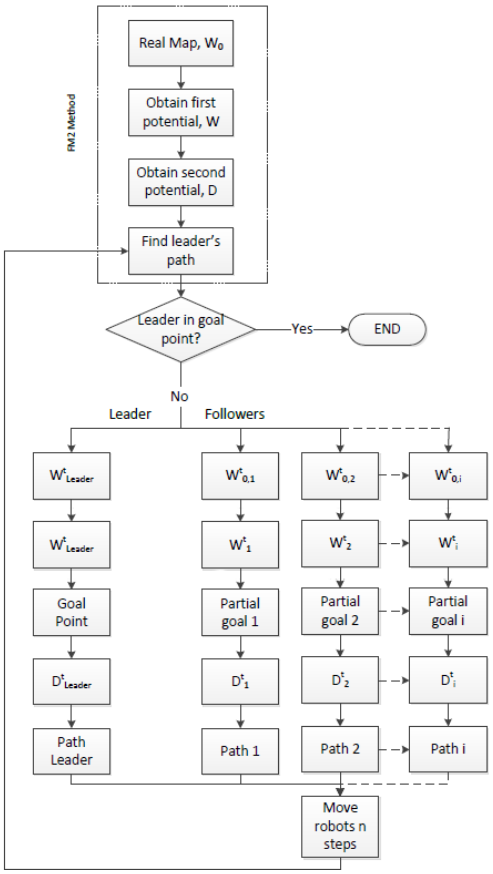

Fig. 9: Robot formation path planning algorithm using FM^2.

II The second potential **D** is updated, so that the changes in **W** are taken into account.

III A new path for the leader is calculated applying gradient descent on the new potential **D**, according to the FM^2 method. Since the environment is dynamic, the previous path that was computed for the leader of the formation may not be safe any more, and so it needs to be recomputed every time any of the obstacles move.

IV According to the new pose of the leader and the default formation geometry, the partial goal (x_{gf}, y_{gf}, z_{gf}) is calculated for each follower f (where f represents every follower in the formation). Initial partial goals are computed with the predefined geometry referenced to the Frenet's trihedron, as explained in section 3.2.

V The grey level, in $\mathbf{W}$, of the partial goal position of each follower f is used in order to recompute this goal, as detailed in section 3.1. Therefore, the shape of the formation is deformed so that the robots move further from obstacles.
VI The second potential $\mathbf{D}_i^t$ is locally calculated for each follower f, applying the FMM to velocities map $\mathbf{W}$. The goal point each follower uses is their partial goal computed on the previous step. The low computational cost of FM^2 allows us to do this without compromising the refresh rate.
VII The path is calculated for each follower f. 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 $\mathbf{D}_i^t$.
VIII All the robots move forward following their paths until a new iteration is completed.

4 Results

According to the previously introduced algorithm, different simulations have been carried out in order to prove the validity of the proposed method. All of them are based on Matlab® implementations of the algorithm running on a Linux-based operating system in a 3.2 GHz dual-core PC (only one core was used in the simulations). Simulations perform only kinematic simulation, meaning that we do not consider any specific dynamic model of the UAVs, and that a perfect path following algorithm is assumed for all the UAVs. In all the cases, the map of the environment is known in advance. Besides, both the initial and the final points of the trajectory are given, and the paths are calculated with the FM^2 algorithm. To calculate the partial goals of the followers, a default shape is previously set. In the simulations carried out, the original formation is a pyramid with quadrangular base, as previously defined in Figure 3. This shape evolves by modifying height of the pyramid and the size of the base. In very narrow situations in which making the size of the base really small could lead to dangerous situations, the followers will evolve into a convoy-like formation increasing the safety. These changes are produced as a function of the grey level of the position of the leader and the followers, as explained in 3.2. In the case of the moving obstacles, both the initial positions and their velocity are also known, although some uncertainty is introduced as explained in 3.3.

While in previous works, only visual results of the simulation are given, in this research a quantitative analysis of the performance of the formation planning algorithm is shown. Two different criteria have been proposed: first, the distance of each robot to the closest obstacle along all the path is evaluated. This is a common criteria used in mobile robotics planning and gives us an idea of how safe is a path with respect to the distance to the obstacles in the environment. In our simulations, it is important to study the safety of the results since the algorithms are tested in very narrow environments and with dynamic obstacles. The second metric measures how much the shape of the formation changes with respect to the initial geometry. It consists on evaluating the average displacement of each robot in the formation from their default location, to their final pose determined by the shape deformation algorithm introduced in this paper. It is computed using equation 9:

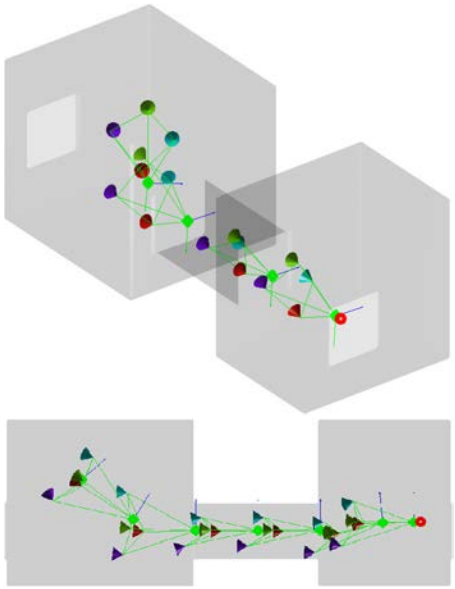

Fig. 10: Example of a motion sequence in a narrow corridor, leveraging the priorities introduced in the algorithm. Top: 3D view. Bottom: top-view which allows to see how the robots change their relative position in the direction of the motion.

$$E = \frac{1}{N} \sum_{i=1}^{N} dist(p_i, d\_p_i) \tag{9}$$

where N is the number of follower robots, *dist* corresponds to the euclidean distance in 3D, p_i is the computed position for follower i and d_p_i is the predefined position of the same robot.

The first simulation environment is shown in figure 10. It consists of two rooms connected by means of a very narrow passage in which the formation does not fit without some amount of shrinking. In this example, we can see how the concept of priorities is inserted in the algorithm since the formation evolves towards a convoy-like geometry in which the followers are assigned with a different priority value. This is done by modifying the height of the pyramid (distance in the tangential direction, d_T) with a different value for each robot, so that it is possible to get extra space between followers. At the same time, the size of the base changes in a way that makes the followers to almost converge into a straight line.

Besides, quantitative analysis according to the metrics aforementioned is shown in figure 11. The top image shows the distance to the closest obstacle for each robot. Although there is a legend in the image, the colour of each line is the same as the colour assigned for the robot in figure 10, and the leader's information is plotted in blue. Horizontal axis corresponds to the steps of the path, being 0 and 64 the start and final pose respectively. Vertical axis shows the distance to the closest obstacle.

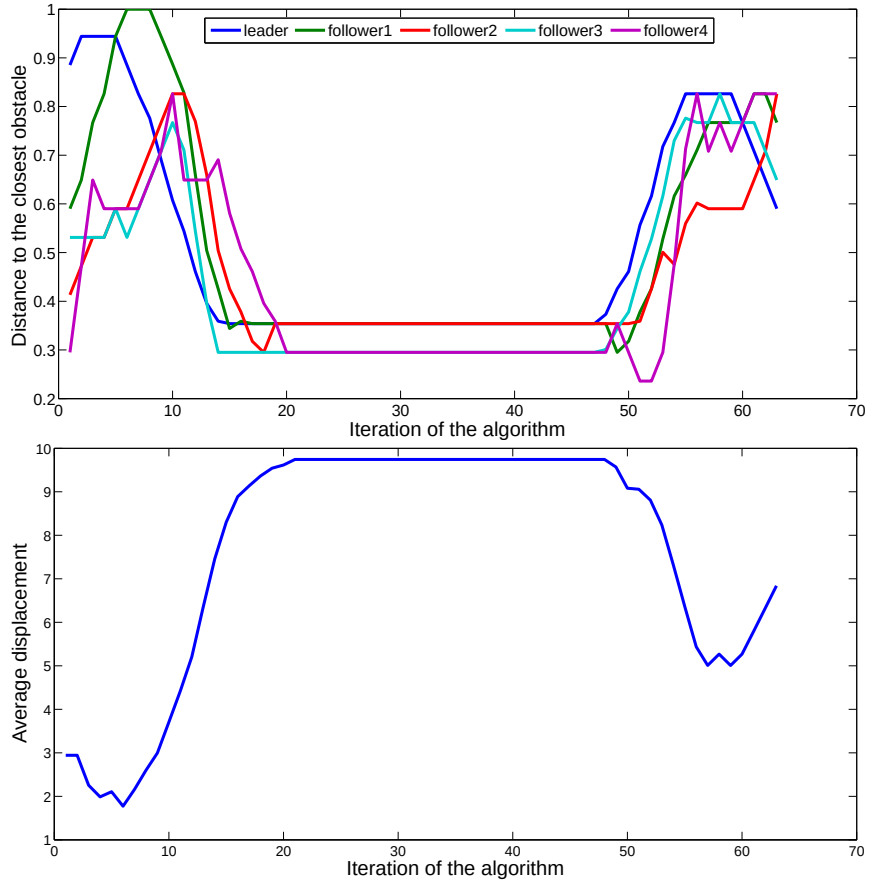

Fig. 11: Quantitative analysis of scenario 1. Top: distance to the closest obstacle. Bottom: average displacement of the formation with respect to the default shape of the formation.

This value is normalised, meaning that a 0 value corresponds to a collision with an obstacle, and a 1 value means the maximum clearance a robot can have in the whole environment. We can observe that, at the beginning, the robots move to a more open space (clearance grows) which corresponds to the movement towards the centre of the first room. Then, the distance gets smaller while the robots approach to the passage, and reaches a minimum value while moving through it. Finally, it grows again while entering in the second room, and becomes smaller when approaching to the goal, which is close to the wall. In terms of deformation (figure 11 bottom), we can see how its evolution is inversely proportional to the distance to the obstacles. This means that when the space in narrower, the shape needs to be deformed in order to fit in the space, which is the expected behaviour.

Figure 12 depicts the results of the same algorithm run in a map of one of the laboratories of our university, in order to show that the proposed algorithm performs

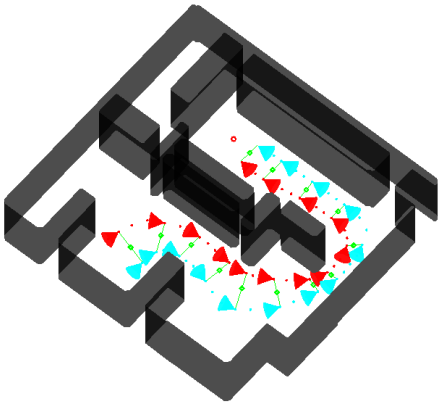

Fig. 12: Example of a motion sequence in a map of one of our laboratories. Two robots (blue and red) move in parallel to a virtual leader (green).

well in realistic environments. In this case, the formation is composed by two followers placed in parallel along the normal vector to the leader and with the tangential distance $t1 = 0$. Therefore, the formation is actually the 2 robots navigating with the leader placed in the straight line that connects the followers. In this case, the leader is considered virtual in order to show the flexibility of the approach. The only deformation allowed is along the normal vector.

The corresponding quantitative results are shown in figure 13. The distances between robots and obstacles are safe and there is not any collision. This performance is remarkable given how cluttered the scenario is.

The next scenario is shown in figure 14. In this case a sequence of positions along the path can be seen. The environment is composed by two big empty rooms connected by a narrow window. Moreover, in the second room there are two mobile obstacles, represented by black balls. These obstacles are located in the same point of the XY plane (floor plane), but at different heights. They move parallel to the floor towards the wall between the rooms. As we can see in the sequence, as explained in section 3.3, the path is different for each step since a new one is calculated at the beginning of it. One can appreciate how during the firsts steps, the path surrounds the obstacles using the space on the left side (from the robots point of view), however, after the robots have passed through the window (at which the formation shrinks in the same way as in the passage in the previous environment) the obstacles move forward and come closer to the wall. This implies that there is more free space on the right side of the obstacles, so the path of the leader changes the side the robots use to avoid the obstacles. Nevertheless, even though the leader eludes the obstacles because of this change, the followers still have to skip the collision. As one can see from the 5^{th} to the 8^{th} images in figure 14, the follower plotted in light blue can avoid the moving obstacles just by following the same shape deformation algorithm used for static environments. This happens because the obstacles are included in the velocities map as explained in section 3.3, so that they cause the same behaviour as

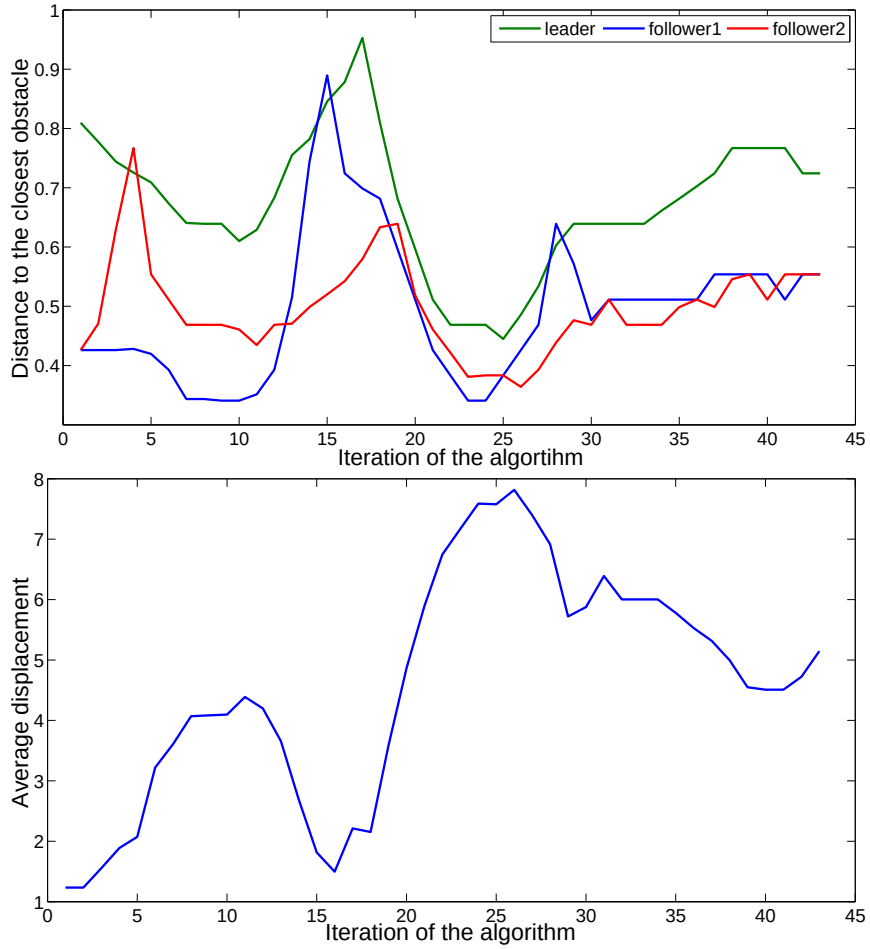

Fig. 13: Quantitative analysis of scenario 2. Top: distance to the closest obstacle. Bottom: average displacement of the formation with respect to the default shape of the formation.

static obstacles. It is important to note, that in the 7^{th} image, although the light blue follower seems to collide with the obstacle, it is just a matter of the perspective used in the visualization.

Moreover, the same quantitative analysis has been applied to this scenario, it is shown in figure 15. As before, the colours in the figure correspond to the same colour of the robots in the simulations. In the case of the distance to obstacles, it can be seen that the value of all the robots decreases towards 0 when they go through the window, and increases afterward. The obstacles do not affect significantly to this values because of the change in the path and the shape deformation movement. In the case of the shape deformation result, it can be seen that there is a very big change in the shape when the robots go through the window and a slight one when they

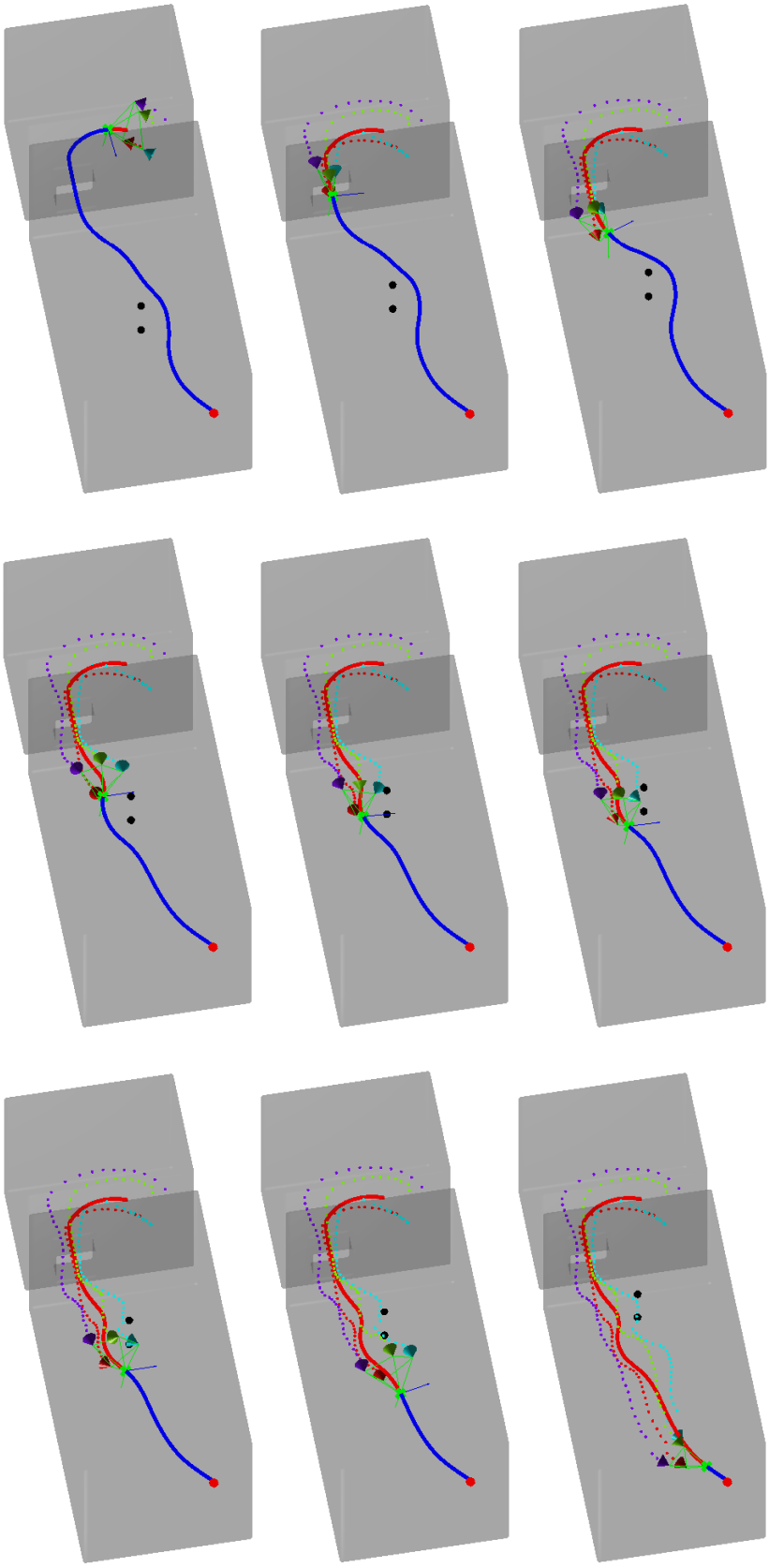

Fig. 14: Environment with mobile obstacles. The temporal evolution starts in the top-left image and ends in the bottom-right image.

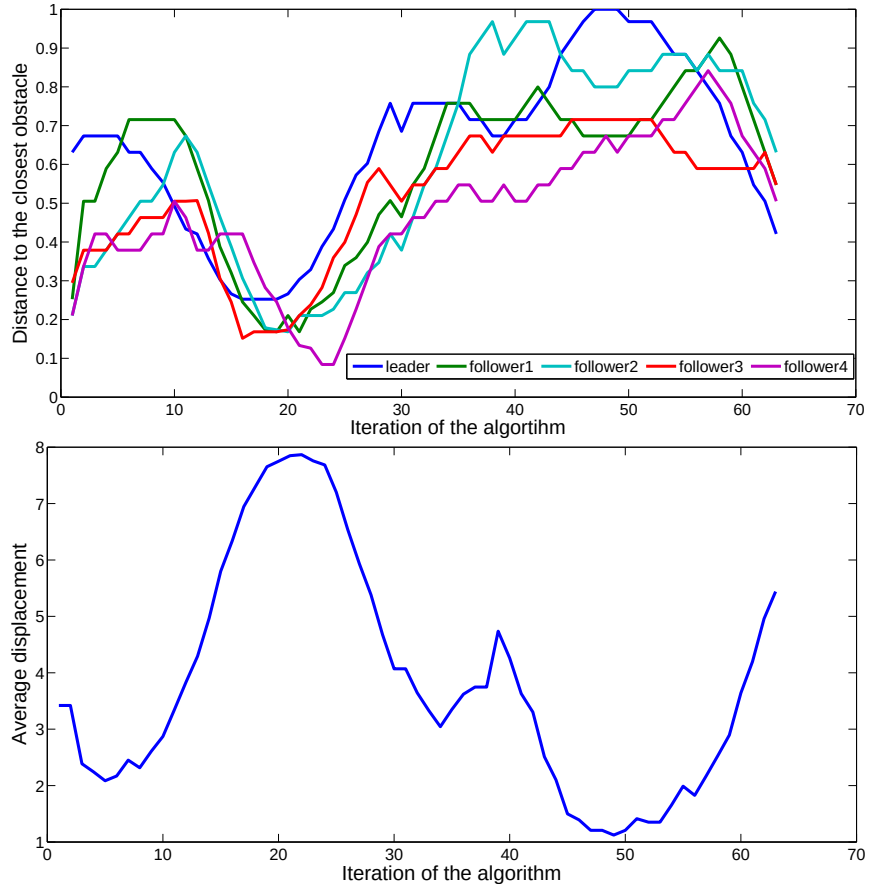

Fig. 15: Quantitative analysis of scenario 3. Top: distance to the closest obstacle. Bottom: average displacement of the formation with respect to the default shape of the formation.

need to avoid the obstacles and arrive to the goal, which is close to the walls of the environment.

Finally, table 1 shows the mean computation times μ and their standard deviation σ for the three previous experiments. For all the experiments the average and standard deviation have been computed over ten different runs of the algorithm. The first column shows the time elapsed in leader's path computations. These times really depend on the environment chosen, specially on the amount of obstacles in them. They are the most time consuming part of the algorithm. Note that in the case of experiments 1 and 2, the leader's path is only computed once, at the beginning of the algorith, however, in experiment 3, a replanning phase is required because of the presence of mobile obstacles. The second column includes formation algorithm times per iteration, which are really fast and could be run in real time with frequencies higher than 1000Hz. Third column contains the times elapsed for all the followers to compute

the path to its partial objectives per iteration. Of course, it depends on the number of followers, but one can see that for 4 followers the computation time is stil around 500ms, so it can be computed online. Note that this can be easily paralellised since every follower computes its own path. Be aware the that the first and third columns are expressed in seconds, while the second is in milliseconds. Also note that the algorithm is completely deterministic, which means that the computations times suffer low variation among runs. Another important remark is that iteration times for experiments 1 and 2 are practically the same, despite the fact that experiment 1 includes 4 followers and experiment 2 only 2. However, experiment 3 needs more time per iteration since the mobile obstacles have to be taken into account as explained in section 3.3. Again, this process could be easily parallelised for every follower reducing considerably the required time.

Table 1: Computation times for the leader's paths, formation algorithm iterations and followers' paths.

	Leader's times (s)		Iteration times (ms)		Followers' times (s)	
	μ	σ	μ	σ	μ	σ
Exp. 1	0.47	0.01	0.22	0.09	0.49	0.03
Exp. 2	0.99	0.1	0.25	0.09	0.44	0.04
Exp. 3	0.65	0.04	10	2	0.52	0.02

5 Conclusions

This research broadens the work presented in [21], for 3D robot formation motion planning, including the necessary changes to apply the algorithm to non-static environments. Besides, it introduces a novel approach to model mobile obstacles in the environment in a way that they are, later on, easily treated by the obstacle avoidance algorithm explained. Finally, quantitative analysis of this approach has been carried out. The results explain how the formation evolves avoiding obstacles while covering the planned path. The tests show that the proposed shape deformation method, in combination with the FM^2 path planner, is robust enough to manage autonomous movements through an indoor static and dynamic 3D environment.

It is important to note that the algorithm is both conceptually and mathematically very simple since it relies on basic natural behaviours such as light movement. Furthermore, the same method is applied in the planning phase for both the leader and the followers. Besides, the deformation schema for the geometry of the robot formation is based on basic 3D translations and rotations which can be computed very fast using standard algebraic tools.

Results show that the proposed algorithm is able to manage difficult environments, modifying the formation when it is necessary. In addition, this approach allows us to include any number of robots in the formation, by only setting the desired position with respect to the leader or the other robots. The introduction of function-based geometry deformation is very powerful since it permits setting very complex

behaviours to the followers by simply modifying the functions. As an example of this, the use of priorities in the formation is showed. These functions can be modified dynamically, an important property that is worthy to explore in the future.

Future work in 3D robot formation using FM^2 is also related to testing this method with other type of formations in which, for example, the leader is not always in front of the team. Also, future simulation should include dynamics in order to prove that the computed paths are smooth enough to be applied to real robots. Besides, the possibility of a goal position change should be included in the high-level formation algorithm, so that the formation can have dynamic goals and the replanning is able to compute the necessary maneuvers to achieve them.

Acknowledgements This work is funded by the project number DPI2010-17772, by the Spanish Ministry of Science and Innovation, and also by RoboCity2030-II-CM project (S2009/ DPI-1559), funded by Programas de Actividades I+D en la Comunidad de Madrid and co-funded by Structural Funds of the EU.

References

1. M. Martin, P. Klupar, S. Kilberg and J. Winter, TechSat 21 and Revolutionizing Space Missions Using Microsatellites, AIAA/USU Conference on Small Satellites (2001).
2. Dewan, A., Mahendran, A., Soni, N., Krishna, K.M., Heterogeneous UGV-MAV exploration using integer programming, AIAA/USU Conference on Small Satellites on Intelligent Robots and Systems (2013).
3. S. Hauert, J.C. Zufferey and D. Floreano, Reverse-engineering of Artificially Evolved Controllers for Swarms of Robots, IEEE Congress on Evolutionary Computation (2009).
4. J.J. Acevedo, B.C. Arrue, I. Maza and A. Ollero, Cooperative Large Area Surveillance with a Team of Aerial Mobile Robots for Long Endurance Missions, Journal of Intelligent and Robotic Systems, Volume 70, Issue 1-4, pp. 329-345 (2013).
5. M. Likhachev, J. Keller, V. Kumar, V. Dobrokhodov, K. Jones, J. Wurz, I. Kaminer, Planning for Opportunistic Surveillance with Multiple Robots, IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pp. 5750-5757, Tokyo (2013).
6. S. Bouabdallah, Design and Control of Quadrotors with Applicationto Autonomous Flying, Theses 3727, cole polytechnique fdrale de Lausanne (2007).
7. S. Hrabar, Reactive Obstacle Avoidance for Rotorcraft UAVs, IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pp. 4967-4974, San Francisco (2011).
8. S. Shen and N. Michael and V. Kumar, 3D Estimation and Control for Autonomous Flight with Constrained Computation, IEEE Intl. Conference of Robotics and Automation, Shanghai (2011).
9. T. Hino, Simple Formation Control Scheme Tolerant to Communication Failures for Small Unmanned Air Vehicles, Intl. Congress of the Aeronautical Sciences, Nice (2010).
10. T. Balch and R. C. Arkin, Behaviour-based Formation Control for Multi-robot Teams, IEEE Transactions on Robotics and Automation, Volume 14, no. 6, pp. 926-939 (1998).
11. D. Naffin and G. Sukhatme, G. Negotiated Formations. Proc. of the Intl. Conference on Intelligent Autonomous Systems, pp- 181-190, Amsterdam (2004).
12. J. Fredslund and M. J. Matari, A general algorithm for robot formations using local sensing and minimal communication, IEEE Transactions on Robotics and Automation, Volume 18, no. 5, pp. 837-846 (2002).
13. M. Lemay, F. Michaud, D. Ltourneau and J.-M. Valin, Autonomous Initialization of Robot Formations, Proc. of the IEEE International Conference on Robotics and Automation, Volume 3, pp. 3018-3023, New Orleans (2004).
14. P. Ogren, M. Egerstedt, X. Hu, A control Lyapunov function approach to multiagent coordination, IEEE Transactions on Robotics and Automation, Volume 18, no. 5, pp. 847–851 (2002).
15. M. Zhang, Y. Shen, Q. Wang, Y. Wang, Dynamic artificial potential field based multi-robot formation control, IEEE Instrumentation and Measurement Technology Conference, pp. 1530-1534, Austin (2010).

16. Z. Cao, L. Xie, B. Zhang, S. Wang, M. Tan, Formation constrained multi-robot system in unknown environments, Proc. Conference on Robotics and Automation, Volume 1, pp. 735-740, (2003).
17. K.H. Tan, M.A. Lewis, Virtual Structures for High-Precision Cooperative Mobile Robotic Control, IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Volume 1, pp. 132-139 (1996).
18. W. Ren and R.W. Beard, Decentralized Scheme for Spacecraft Formation Flying via the Virtual Structure Approach, AIAA Journal of Guidance, Control and Dynamics, Volume 1, no. 1, pp. 73-82 (2004).
19. A. Ahmad, T. Nascimento, A. G. S. Conceiao, A. P. Moreira, P. Lima, Perception-Driven Multi-Robot Formation Control, IEEE Intl. Conference on Robotics and Automation, pp. 1851-1856, Karlsruhe (2013).
20. K. Kanjanawanishkul and A. Zell, A model-predictive approach to formation control of omnidirectional mobile robots, IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pp. 2771-2776, Nice (2008).
21. D. Álvarez, J.V. Gómez, S. Garrido and L. Moreno, 3D Robot Formations Planning with Fast Marching Square, IEEE Intl. Conference on Autonomous Robot Systems and Competitions, Espinho (2014).
22. W. Yu, G. Chen, and M. Cao, Distributed leaderfollower flocking control for multi-agent dynamical systems with time-varying velocities, Systems and Control Letters, Volume 59, no. 9, pp. 543-552 (2010).
23. S. Garrido, L. Moreno and P. Lima, Robot Formation Motion Planning Using Fast Marching, Journal of Robotics and Autonomous Systems, Volume 59, no. 9, pp. 675-683 (2011).
24. D. Álvarez, A. Lumbier, J.V. Gómez, S. Garrido and L. Moreno, Precision Grasp Planning with Gifu Hand III based on Fast Marching Square, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4549-4554, Tokyo (2013).
25. A.Valero; J. V.Gómez; S.Garrido; L.Moreno. The Path to Efficiency. IEEE Robotics and Automation Magazine, Volume 20, no. 4, pp. 111-120 (2013).
26. J. V. Gómez, A. Lumbier, S. Garrido and L. Moreno, Planning Robot Formations with Fast Marching Square Including Uncertainty Conditions, Journal Robotics and Autonomous Systems, Volume 61, no. 2, pp. 137-152 (2013).
27. J. A. Sethian, A Fast Marching Level Set Method for Monotonically Advancing Fronts, Proc. National Academy of Science, Volume 93, no. 4, pp. 1591-1595 (1996).
28. S. Osher and J. A. Sethian, Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations, Journal of Computational Physics, no. 79, pp. 12-49 (1988).
29. S. Garrido, L. Moreno, M. Abderrahim and D. Blanco, FM2: A Real-time Sensor-based Feedback Controller for Mobile Robots, Intl. Journal on Robotics and Automation, Volume 24, no. 1, pp. 3169-3192 (2009).
30. L. Yatziv, A. Bartesaghi and G. Sapiro, A Fast O(n) Implementation of the Fast Marching Algorithm, Journal of Computational Physics, Volume 212, no. 1, pp. 393-399 (2005).
31. S. Garrido, L. Moreno, J.V. Gómez and P.U. Lima, General Path Planning Methodology for Leader-Followers based Robot Formations, Intl. Journal of Advanced Robotic Systems, Volume 10, no. 64, pp. 1-10 (2013).
32. J.V. Gómez, S. Garrido and L. Moreno, Adaptive Robot Formations using Fast Marching Square working under Uncertainty Conditions, IEEE Workshop on Advanced Robotics and Its Social Impacts, pp. 68-71, California (2011).
33. F. Frenet, Sur les Courbes à Double Courbure, Journal de Mathématiques Pures et Appliquées, Volume 1, no. 17, pp. 437-447 (1852).
34. J. A. Serret, Sur Quelques Formules Relatives à la Théorie des Courbes à Double Courbure, Journal de Mathématiques Pures et Appliquées, Volume 1, no. 16, pp. 193-207 (1851).
35. N. Dalal and B. Triggs, Histograms of oriented gradients for human detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Volume 1, pp. 886-893, San Diego (2005).
36. P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 32, Issue 9, pp. 1627-1645 (2009).
37. R. Volpe and P. Khosla, Manipulator Control with Superquadric Artificial Potential Functions: Theory and Experiments, IEEE Transactions on Systems, Man, and Cybernetics, vOLUME 20, PP. 1423-1436 (1990) .
38. S. El-Khoury and A. Sahbani, A new strategy combining empirical and analytical approaches for grasping unknown 3d objects, Journal of Robotics and Autonomous Systems, Volume 58, Issue 5, pp. 497-507 (2010).