Improving Sampling-based Path Planning Methods with Fast Marching

Javier V. Gómez, David Álvarez, Santiago Garrido, and Luis Moreno

RoboticsLab, Carlos III University of Madrid, Avda. de la Universidad 30, 28911, Leganés, Madrid, Spain.
jvgomez, dasanche, sgarrido, moreno@ing.uc3m.es,
WWW home page: http://roboticslab.uc3m.es/

Abstract. Sampling-based path planning algorithms are well-known because they are able to find a path in a very short period of time, even in high-dimensional spaces. However, they are non-smooth, random paths far away from the optimum. In this paper we introduce a novel improving technique based on the Fast Marching Method which improves in a deterministic, non-iterative way the initial path provided by a sampling-based methods. Simulation results show that the computation time of the proposed method is low and that path length and smoothness are improved.

Keywords: Fast Marching, Path Planning, Path Improvement

1 INTRODUCTION

Sampling-based path planning algorithms are nowadays one of the most powerful tools to solve planning problems, specially in high-dimensional spaces. Since the first versions of these algorithms appeared [1, 2], they have been applied to many different problems and many different versions have appeared, improving more and more their performance [3].

However, the drawbacks of these methods are well-known. They are based on random (or pseudo-random) space sampling. This leads to non-optimal, stochastic paths which are far away from the optimal one (in terms of distance, obstacle clearance, etc). Many optimization techniques have been already proposed [4–6]. These optimization techniques, based on iterative processes, achieve the optimal path in terms of distance. Nevertheless, these optimal paths are still non-smooth and the obstacle clearance is minimum, which can be dangerous in real applications.

In this paper we introduce a method for improve the sampling-based paths by increasing the smoothness and decreasing the path length. Also, a *good enough* obstacles clearance is ensured. The proposed algorithm tries to imitate the Fast Marching Square (FM^2) algorithm [7] but applied over a triangular mesh. Modeling a continuous space as a triangular mesh and computing the paths as geodesics is already proposed in [8]. However, the requirements for the triangular mesh and

the method used for computing the distances map makes this method computationally slow. In our case, we restrict the triangular mesh to a *region of interest* around an initial sampling-based path and compute the geodesic of the mesh with the Fast Marching Method (FMM) [9] speeding up the computation time and obtaining high-quality paths.

This paper is organised as follows. Next section explains the path initialization algorithm. Section 3 outlines the FMM applied to irregular triangle meshes. Following, section 4 details the proposed algorithm. The results are given in section 5 and finally the conclusions of the work are extracted in section 6.

2 SAMPLING-BASED PATH INITIALIZATION

The reason we choose to use a sampling-based algorithm as a path initialization is because these algorithms are known to be the fastest to provide a reliable path between two points when no previous experience is taken into account.

In this case, we will use the Rapidly-exploring Random Trees (RRT) algorithm which is one of the basic algorithms within the sampling-based group [2]. However, since most of the sampling-based methods share the same properties (fast response, non-smoothness, low obstacles clearance...) [3] any kind of sampling-based algorithm is suitable for the purpose of this paper.

The RRT algorithm is detailed in algorithm 1. Formalizing, the RRT creates a tree T, which is a set of N vertices V and $N-1$ edges connecting those vertices E, $T(V, E)$. Following, the main components of the algorithm are described:

- *Sample(i)* Provides uniformily random samples from the obstacles-free space.
- *Nearest*(V, x_{new}) Returns the closest vertex $v \in V$ to the point given in the argument x_{new}.
- *Steer*$(x_{new}, x_{nearest})$ Creates a path σ between the two points given. Usually, it is a straight line between those two points.
- *CollisionFree*(σ) Returns true if the path given σ is collision free.

The finishing condition in this case is to sample N times. However, this can cause that the algorithm ends and it does not find any possible path. Then, the finishing condition can be easily changed for any other, e.g. one of the vertices of the tree is close enough to the goal point. Figure 1 shows a tree expanding using as ending condition the number of vertices to expand N. Figure 2 plots the paths obtained with RRT when the ending condition is to reach a vertex close to the goal point.

3 FAST MARCHING METHOD

The FMM are a set of algorithms for computing consistent distance maps. The first approach was based on regular orthogonal grids [10, 11]. Later, these algorithms were extended to general triangular meshes [12]. Since triangular meshes are more flexible when describin shapes, we will focus on this version of the FMM.

Algorithm 1 The RRT Algorithm

Input: Initial point x_0.
Output: Tree T.

$V \leftarrow \{x_0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ to N **do**
  $x_{new} \leftarrow \textsc{Sample}(i)$;
  $x_{nearest} \leftarrow \textsc{Nearest}(V, x_{new})$;
  $\sigma \leftarrow \textsc{Steer}(x_{new}, x_{nearest})$;
  if $\textsc{CollisionFree}(\sigma)$ **then**
    $V.\textsc{add}(x_{new})$;
    $E.\textsc{add}(x_{new}, x_{nearest})$;
  end if
end for
return $T = (V, E)$.

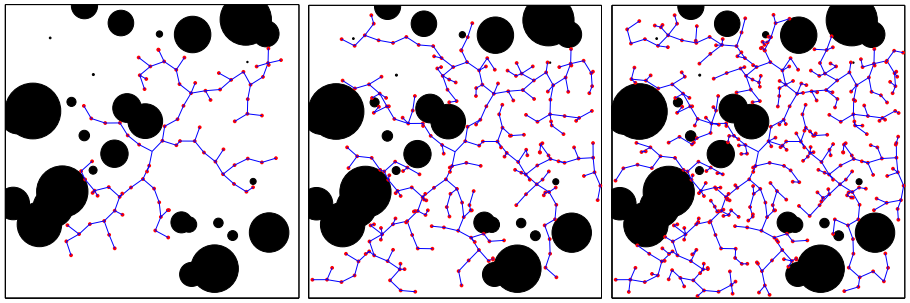


Fig. 1: Examples of a tree created with the RRT algorithm. From left to right: $N = 100, 300, 500$.

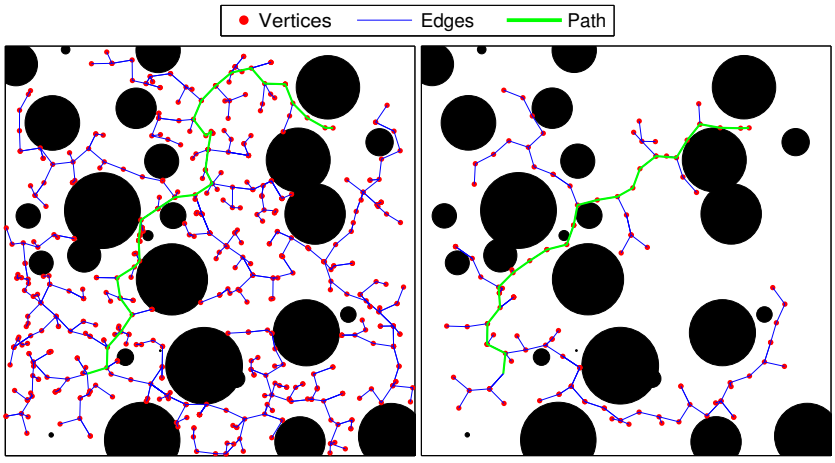


Fig. 2: Paths obtained when planning with the RRT algorithm.

In this section we outline the FMM applied in the following sections of this paper. A detailed analysis of the FMM can be found in [9], whose notation we will use. Let X be the surface defined by a triangle mesh and x a coordinate parametrization of X, $x : U \to X$. FMM is a procedure to build a distance map $d(x) = d_X(x_0, x)$ by numerically solving the Eikonal equation:

$$||\nabla_X d(x)||_2 = 1 \tag{1}$$

where ∇_X represents the intrinsic gradient with the boundary condition $d(x_0) = 0$.

Intuitively, FMM simulates a wavefront propagation computing the time of arrival $d(x)$ for every point of the space when the wave propagates with constant, non-negative velocity. Let us suppose that a wave starts propagating at x_0 with $d(x_0) = 0$. This point of the mesh is already *frozen* (its value will never change). By *open* points we denote those points of the mesh which have not been visited yet by the wave, therefore $d(x) = \infty$. Finally, points in the *narrow band* are those belonging to the wave front, acting as an interface between frozen and open points. Algorithm 2 describes the FMM algorithm to compute the distance map $d(x)$.

Algorithm 2 Fast Marching Method

Input: non-obtuse triangular mesh (X, T), source point x_0.
Output: distance map $d : X \to \mathbb{R}$ from the source point.

Initialization.
1: **for all** $x \in X$ **do**
2: $d(x) \leftarrow \infty$;
3: **end for**
4: $d(x_0) \leftarrow 0$;
5: $frozen \leftarrow x_0$;
6: $narrow \leftarrow \mathcal{N}(x_0)$; ▷ Neighbours of x_0.
7: $open \leftarrow X \backslash (frozen \cup narrow)$;
Iteration.
8: **while** $frozen \neq X$ **do**
9: $x_1 \leftarrow \underset{x \in narrow}{\arg\min}\, d(x)$;
10: **for all** $t(x_1, x_2, x_3) \in \{(x_1, x_2, x_3) \in T : x_2 \in frozen \cup narrow, x_3 \in narrow \cup open\}$ **do**
11: $narrow \leftarrow narrow \cup \{x_3\}$;
12: UPDATE(x_1, x_2, x_3);
13: **end for**
14: $narrow \leftarrow narrow \backslash \{x_1\}$; ▷ Updating sets.
15: $frozen \leftarrow frozen \cup \{x_1\}$;
16: **end while**

The main difference between FMM and Dijsktra's algorithms [13] is the *Update* step in the line 12 of algorithm 2. This update step, detailed in algorithm 3

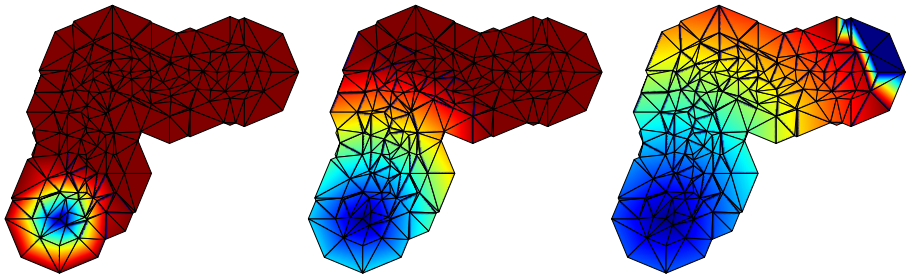


Fig. 3: Front wave propagation in a triangular mesh. The source point is at the bottom of the mesh.

requires two vertices of the same triangle in order to compute the time of arrival of the third vertex. This allows the geodesics to go for any place within the triangle mesh following the gradient of the distance map $d(x)$, which is approximated by the wave propagation direction n.

Algorithm 3 FMM Update Step

Input: non-obtuse triangle with vertices x_1, x_2, x_3 and the corresponding arrival times d_1, d_2, d_3.
Output: Updated d_3.

$V = (x_1 - x_3, x_2 - x_3)$;
$d = (d_1, d_2)^T$;
$Q = (V^T V)^{-1}$;
$p = \frac{1_{2\times1}^T Q d + \sqrt{(1_{2\times1}^T Q d)^2 - 1_{2\times1}^T Q 1_{2\times1} \cdot (d^T Q d - 1)}}{1_{2\times1}^T Q 1_{2\times1}}$;
$n = V^{-T}(d - p \cdot 1_{2\times1})$;
if $QV^T n < 0$ **then** ▷ Monotonicity condition.
    $d_3 \leftarrow \min\{d_3, p\}$;
else
    $d_3 \leftarrow \min\{d_3, d_1 + ||x_1||, d_2 + ||x_2||\}$;
end if

The aforementioned algorithm works for non-obtuse triangular meshes. In case there is any non-obstuse triangle in the mesh, it can be solved by connecting the vertex x_3 to another point on the mesh [12]. Figure 3 depicts the front wave propagation following the FMM. Also, Figure 4 includes the geodesic computed with FMM. In fact, it is not the optimal geodesic, since FMM carries out some approximations when computing $d(x)$. If necessary, it is possible to improve the accuracy of the geodesic by: 1) increasing the number of triangles of the mesh 2) applying other, more advanced FMM methods [14] ot increasing the order of the finite differences Eikonal solver [15]. In any case, the method would become more complex and slower.

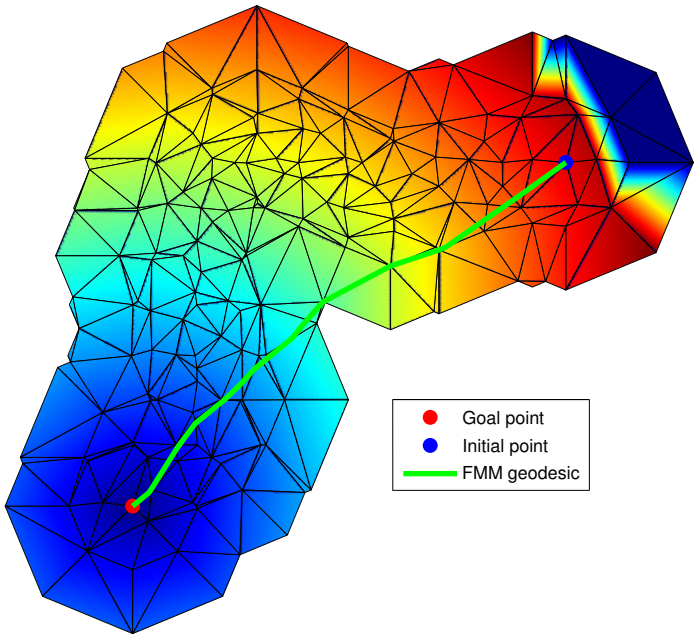


Fig. 4: Geodesic computed with FMM on a triangle mesh. Plotted together with the distances map $d(x)$.

4 IMPROVING PATH QUALITY WITH FAST MARCHING

In this section, we describe a novel technique for improving paths. We want to note that it is not an optimization technique since our method is not looking for an optimal solution. However, we propose a fast technique which improves the quality of a given path. In terms of path length, obstacle clearance and smoothness the final path will be closer to the optimal one. Although we cannot guarantee that the final solution is optimal, we provide a deterministic, non-iterative algorithm which will improve the initial path.

The proposed method is based on creating a tube around the initial path. Next, a triangular mesh is created within the tube and the FMM method is applied. However, in this case the velocity of the propagating wave will not be constant, but directly proportional to the distance to the closest obstacle. This is very close to the Fast Marching Square (FM^2) algorithm deeply studied in the recent years [7, 16]. Computing geodesics in the distance map generated will provide collision-free, near-optimal paths in terms of path length (since FMM approximates the geodesics). The proposed algorithm, summarized in algorithm 4, is detailed in the following lines.

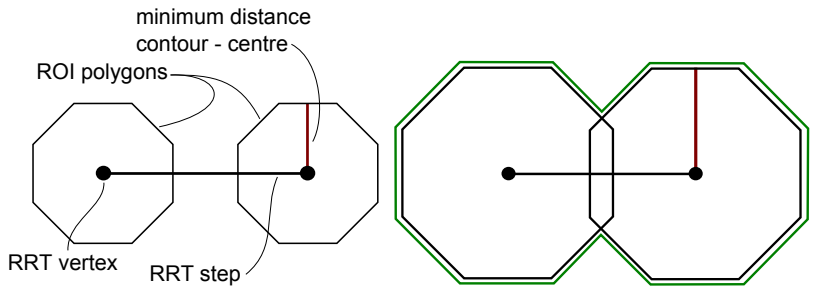
Algorithm 4 Proposed Path Improving Algorithm

Input: Initial path $p(V_p, E_p)$.
Output: Final, improved path q.

for all $v \in V_P$ **do**
  $(V_{ROI,v}, V_{M,v}) \leftarrow$ ComputeVertices(v);
  $V_{ROI} \leftarrow$ UpdateROI$(V_{ROI}, V_{ROI,v})$;
  $V_M \leftarrow (V_M \cup V_{M,v})$;
end for
$X \leftarrow (V_p \cup V_{ROI} \cup V_M)$;
$(X, T) \leftarrow \mathrm{CDT}(X, V_{ROI})$;
$d(x) \leftarrow \mathrm{FMM}(V_{P,0}, V_{P,g})$;
$q \leftarrow$ ComputeGeodesic$(d(x), V_{P,g})$;

4.1 Mesh generation

Once the RRT (or any other sampling-based algorithm) has been applied, an initial path p is obtained, expressed as a subset of the tree T: $p(V_p, E_p) \subset T(V, E)$. A *region of interest* (ROI) has to be defined, so the mesh is confined around the initial path. This region of interest is created by placing a regular polygon with the center at every vertex V_p. It is important that the polygons cover the area needed to ensure connectivity of the polygons of adjacent vertices. This requirement is easy to overcome: when planning with RRT (or similar), the main parameter is the length of the step given in the *Steer* step. This parameter is the maximum possible distance between two vertices of the tree connected by an edge. Therefore, connectivity of the polygons will be ensured as long as the shortest line between the polygon contour and its center is larger than half of the RRT step parameter, as shown in figure 5.



(a) No connectivity. **(b)** Connectivity ensured.

Fig. 5: ROI connectivity according algorithms parameters.

Once all the polygons have been created, the external contour, expressed as a set of vertices V_{ROI}, is computed by computing the union of all the polygons, as shown in figure 5 b).

With the ROI computed, the next step is to create a triangular mesh inside this ROI. Before creating the mesh, it is mandatory to define the set of vertices V_M that will be used when triangulating. Although a uniform random sampling could be a good option for this case, it could be complex and time consuming to ensure that all the points created randomly are in the ROI. For this reason, we have decided to create a structured sampling pattern.

When calculating the vertices of the polygons for the ROI, also the middle points between the center and vertices are computed and included in the set V_M. Therefore, the vertices to be used in the triangulation will be: $X = (V_p \cup V_{ROI} \cup V_M)$ (figure 6). Note that we are not restricting these vertices to be in the free space. Since the obstacles will be taken into account in the next step, it is not necessary to spend time checking the vertices generated.

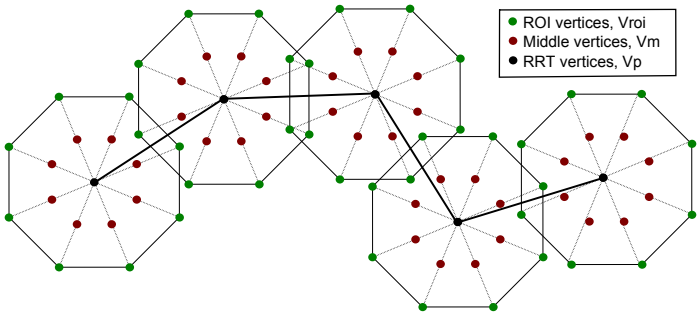


Fig. 6: Set of all the vertices X to be used in the triangulation.

Finally, the Constrained Delaunay Triangulation algorithm (CDT) [17] is applied to the set of vertices X and the ROI contour defined by V_{ROI}, obtaining a triangular mesh (X, T). An example is given in figure 7. Any other triangulation algorithm could be chosen here. However, CDT maximizes the minimum angles of the triangles, so it tries to avoid skinny triangles, which is a desirable property for a FMM input mesh. Also, CDT is easily extensible to more dimensions.

4.2 Application of FMM

The next step is to compute the distances map $d(x)$ for the generated mesh (X, T). The FMM is applied from the goal point of the path $V_{p,g}$ until the front wave reaches the initial point of the path $V_{p,0}$. However, the propagation velocity used is not constant. The geodesics, when computing using FMM tend to go closer to the places where the wave can expand faster, following the *least action principle* [9]. Therefore, $d(x)$ is translated into a *times-of-arrival* map.

When updating vertex x_3 from (x_1, x_2), the distance to the initial point d_3 is updated as mentioned in section 3. However, when the propagation velocity is not uniform, after calculating d_3 is is necessary to update it, to compute the time of arrival approximated as:

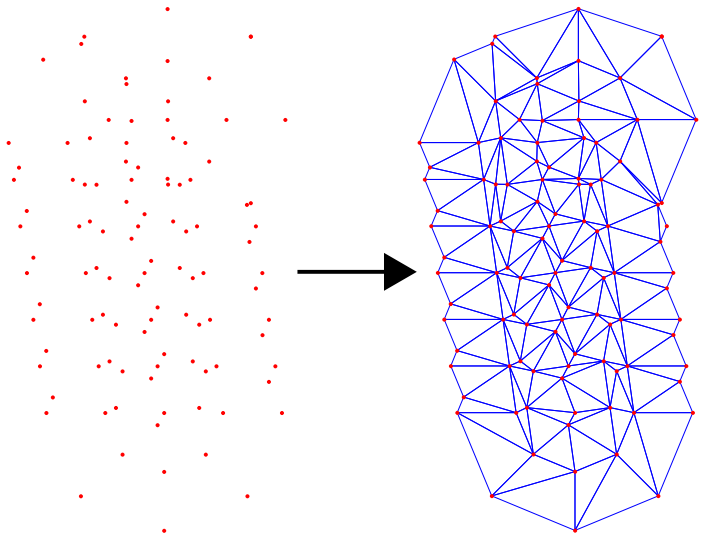


Fig. 7: Example of the CDT algorithm applied to a set of vertices X.

$$d_3 = \min(d_1 + v_3 \cdot ||x_3 - x_1||, d_2 + v_3 \cdot ||x_2 - x_1||) \tag{2}$$

Where v_3 is the wave propagation velocity towards x_3. In figure 8 the velocities for every point of the mesh are shown. The closer a vertex is to an obstacle, the slower the wave propagates towards that vertex. The resulting wave propagation with this velocities map is shown in figure 9. Comparing this figure with figure 3 it is possible to appreciate the influence of the obstacles when computing the time of arrival value for each vertex.

Finally, in order to obtain the final, improved path gradient descent is applied from the point $V_{p,0}$. Since only one wave was expanded from the goal point $V_{p,g}$, gradient descent will always converge to this point since it is the only minimum in $d(x)$. The final path in comparison with the initial RRT path is shown in figure 10 (octagons were used in the mesh generation).

5 RESULTS

The results will be expressed as a set of metrics and their variation for the initial RRT path and the improved path after applying FMM. The experimental setup comprises 3 different, random environments, 5 random path queries for every enviroment, and every query computed 10 times. Regarding the algorithms, the workspace size was 100×100 (the units are not important, since we are always using the same scale). 100 random obstacles were generated, with a maximum radius of 2.5. The RRT maximum step size was set to 2. In FMM, octogons were used to create the mesh with a radius of 5.

The metrics computed are the following:

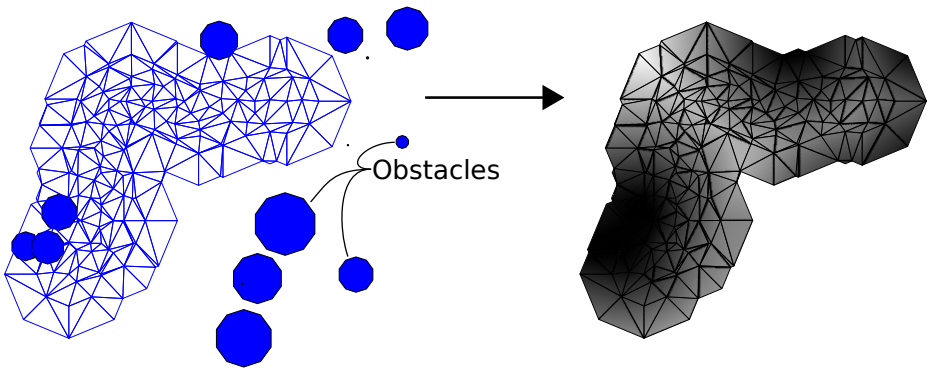


Fig. 8: Velocities map used when propagating the wave. The darker the slower the wave propagates.

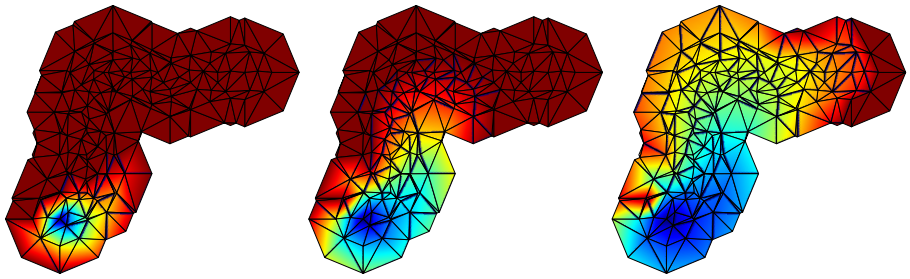


Fig. 9: Front wave propagation in a triangular mesh with varying velocity. The source point is at the bottom of the mesh.

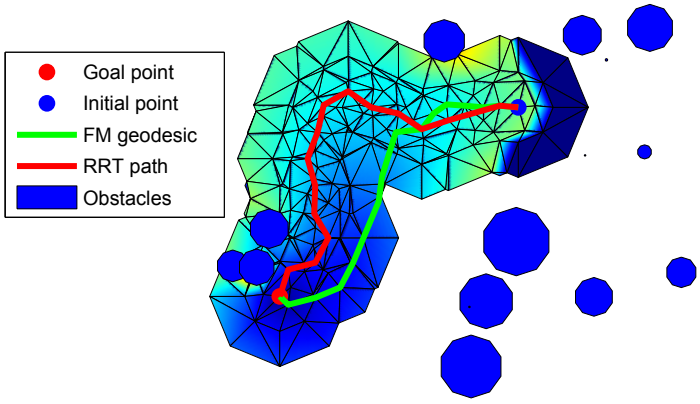


Fig. 10: Comparison of the initial RRT path the geodesic computed with the proposed method.

- **RRT Computation time** - The time t_{RRT} (in s) the RRT algorithm took to compute the initial path.
- **FM Computation time** - The time T_{FMM} (in s) the proposed algorithm took to improve the path.
- **Deviation in path smoothness** - The smoothness κ' can be measured in many different ways. We will use the smoothness metric given in [18], which represents the standard deviation of the angles along the path. Let α_i be the angle between two consecutive segments of a path divided into m segments. Therefore, $\kappa' = \sqrt{\frac{1}{m-1}\sum_{i=2}^{m}\alpha_i^2}$. The angle taken into account is illustrated in Figure 11. Since we are comparing the deviation between the FMM and RRT paths, we compute the smoothness ratio $\kappa' = \kappa'_{FMM}/\kappa'_{RRT}$.

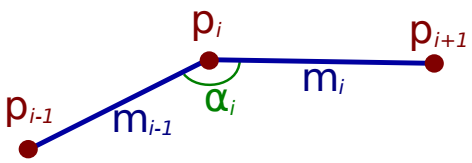


Fig. 11: The angle between two consecutive segments of a path.

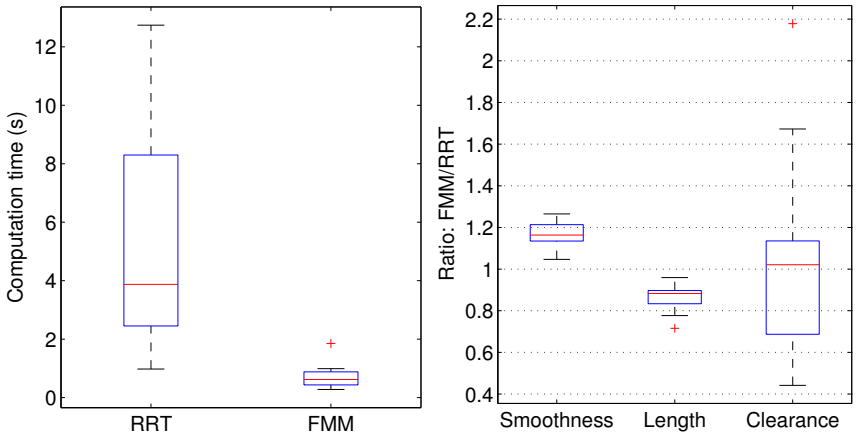
- **Deviation in path length** - The path length l is approximated by dividing the path into n points $P = \langle p_1, p_2, ..., p_n \rangle$ and computing $l = \sum_{i=1}^{n-1} d_E(p_i, p_{i+1})$, where d_E stands for the Euclidean distance. Therefore, we compute the deviation as the ration $l = l_{FMM}/l_{RRT}$.
- **Deviation in Minimum Obstacles clearance** - The metric d_n contains the deviation of the minimum distance of the points along the path to the closest obstacles of the environment. It is computed as the ratio $d_n = d_{n,FMM}/d_{n,RRT}$.

Figure 12 summarises the results obtained for the metrics aforementioned. The graph shows the distribution of results for the 15 experiments done (3 environemts, 5 queries per environment) using the means obtained per every query (computed 10 times). Figure 12 a) shows the computation times of both initialisation path obtained with RRT and the proposed, FMM-based algorithm. Although the absolute values are not meaningful (because of implementation issues), the conclusion from this graph is that the novel method computation time does not vary that much as RRT. The longer the initial RRT path is the longer the optimzation takes, but in a restricted period of time.

From figure 12 b) it is possible to evaluate the proposed algorithm. The smoothness ratio is close to 1.2, which means that the average smoothness improvement is around 20%. Also, note that for all the experiments, this ratio is higher than 1. Analogously, the length ratio is always lower than 1 (average around 0.85) which supposes an average improvement of the path length of 15%. Lastly, the clearance ratio requires a deeper analysis. The RRT algorithm does

not take into account the distance to obstacles when planning. Therefore, the minimum clearance is totally random (it could be, in fact, a safe trajectory). Therefore, it is not a significant result.

However, it is possible to ensure that the FMM method provides a good enough minimum clearance (where *good enough* depends on the application). In our case, the wave propagation velocity was proportional to the distance to the closest obstacle. If this proportion is made stronger (let us say, proportional to the square of the distance to the closest obstacle) the geodesic will be longer in terms of length but with a better minimum clearance.



(a) Computation times. **(b)** Metrics ratios

Fig. 12: Results of the proposed FMM-based algorithm.

Finally, figure 13 includes the visualisation of the application of the algorithm.

6 CONCLUSIONS

Along this paper we have proposed and analysed a novel method for improving initial paths computed with fast, non-optimal algorithms. The results show that the computation time is acceptable, but there is a huge margin for improvement since the implementation is not optimised.

The key properties of the proposed method are that it is deterministic and non-iterative. For a given initial path, it will always produce the same output and in one iteration (although the underlying FMM method is iterative, we are applying it as a *black box*). In some, very specific cases it is possible to have different outputs because of the CDT, which is unique for most of the meshes.

As this work is still under development, there are many ways of improving the results. Testing different, more efficent velocities map can improve the quality of

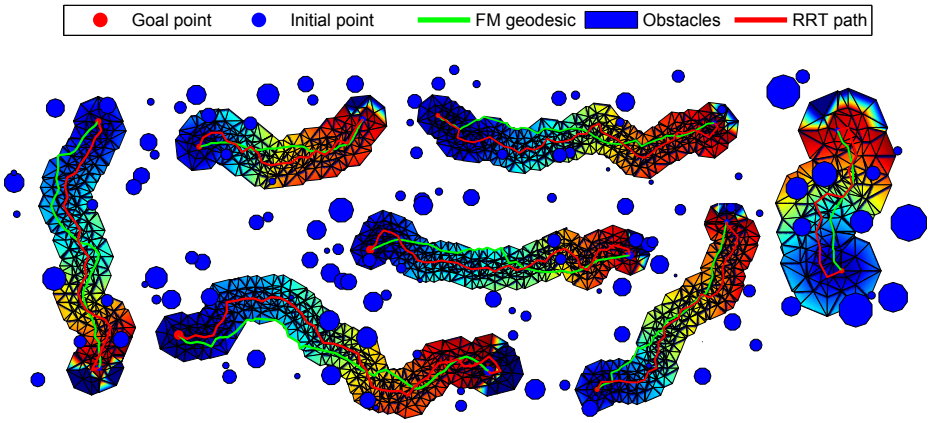


Fig. 13: Different executions of the algorithm and their results.

the final path. Also, the addition of other triangulation methods (or a different sampling for the mesh vertices) could improve the computation time.

Apart of the aforementioned, future work will focus on see how the parameters of both RRT and FMM affect the improvement and to try to apply it to more generic scenarios, where obstacles are not represented by circles.

Finally, this paper has focused on a 2D implementation. However, all the components of the algorithm are defined for n-dimensional space.

ACKNOWLEDGMENT This work was supported by the projects number DPI2010-17772 and CSD2009-00067 HYPER-CONSOLIDER INGENIO 2010 of the spanish MICYT.

References

1. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. on Robotics and Automation, 12, 566–580 (1996).
2. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Intl. J. of Robotics Research, 20, 378–400 (2001).
3. LaValle, S.M.: Planning Algorithms, Cambridge University Press (2006).
4. Keraman, S., Farazolli, E.: Sampling-based algorithms for optimal motion planning. Intl. J. of Robotics Research, 30, 846–894 (2010).
5. Perez, A., Karaman, S., Shkolnik, A.C., Frazzoli, E., Teller, S.J., Walter, M.R.: Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. In: IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (2011).
6. Islam, F., Nasir, J., Malik, U., Ayax, Y., Hasan, O.: RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution. In: Intl. Conf. Mechatronics and Automation (2012).

7. Valero, A., Gómez, J.V., Garrido S., Moreno, L.: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories. IEEE Robotics and Automation Magazine, in press.
8. Yershov, D.S., LaValle, S.M.: Simplicial Dijkstra and A* algorithms: from graphs to continuous spaces. Advanced Robotics, 26, 2065–2085 (2012).
9. Bronstein, A.M., Bronstein, M.M., Kimmel, R.: Numerical geometry of non-rigid shapes. pringer Monographs in Computer Science (2008).
10. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. IEEE Trans. Automatic Control, 40, 1528–1538 (1995).
11. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proc. National Academy of Sciences, 93, 1591–1595 (1996).
12. Kimmel, R., Sethian, J.A.: Computing geodesic paths on manifolds. National Academy of Sciences, 95, 8431–8435 (1998).
13. Dijkstra, E.W.: A Note on Two Problems in Connexion With Graphs. Numerische Mathematik, 1, 269–271 (1959).
14. Hassouna, M.S., Farag, A.A.: Multistencils Fast Marching methods: a highly accurate solution to the Eikonal equation on cartesian domains. IEEE Trans. Pattern Analysis and Machine Intelligence, 29, 1563–1574 (2007).
15. Ahmed, S., Bak, S., McLaughlim, J., Renzi, D.: A third order accurate fast marching method for the Eikonal equation in two dimensions. J. on Scientific Computing, 33, 2402–2420 (2011).
16. Garrido, S., Moreno, L., Abderrahimm M., Blanco, D.: FM^2: A Real-time sensor-based feedback controller for mobile robots. Intl. J. of Robotics and Automation, 24, 3169–3192 (2009).
17. Chew, L.P.: Constrained Delaunay Triangulations. In: Proc. of the Third Annual Symposium on Computational Geometry. pp. 215–222 (1987).
18. Cohen, B., Şucan, I.A., Chitta, S.: Generic Infrastructure for Benchmarking Motion Planners. In: IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (2012).