Path Planning for Autonomous Underwater Vehicles

Clément Pêtrès, Yan Pailhas, Pedro Patrón, Yvan Petillot, Jonathan Evans and Dave Lane
Ocean Systems Laboratory, Heriot-Watt University, Edinburgh, EH14 4AS, UK

***Abstract*— Efficient path planning algorithms are a crucial issue for modern autonomous underwater vehicles. Classical path planning algorithms in artificial intelligence are not designed to deal with wide continuous environments prone to currents. We present a novel Fast Marching based approach to address the following issues. First, we develop an algorithm we call FM* to efficiently extract a continuous path from a discrete representation of the environment. Secondly we take underwater currents into account thanks to an anisotropic extension of the original Fast Marching algorithm. Thirdly, the vehicle turning radius is introduced as a constraint on the optimal path curvature for both isotropic and anisotropic medias. Finally, a multiresolution method is introduced to speed up the overall path planning process.**

***Index Terms*— path planning, Fast Marching, FM* algorithm, autonomous underwater vehicle, turning radius, currents, multiresolution method.**

I. Introduction

A. Underwater environment and autonomous underwater vehicle

In mobile robotics, path planning research has focussed on wheeled robots moving on 2D surfaces equipped with high rate communication modules. The underwater environment is much more demanding: it is difficult to communicate because of low bandwidth channels undersea; it is prone to currents; and the workspace may be worldwide. Moreover, torpedo-like vehicles are strongly nonholonomic.

The current state of technology allows many laboratories to move forward in the development of autonomous underwater vehicles (AUV). The need of a reliable cognition process for finding a feasible underwater trajectory solution is important.

B. Contributions

The main contribution of the authors is to present a Fast Marching based method as an advanced tool for underwater path planning. With a similar complexity to classical techniques in artificial intelligence, Fast Marching algorithm (FM) converges to a smoothed continuous solution when implemented on a sampled environment. This specificity is crucial to understand the other contributions of our method:

- An heuristically guided version FM* of the Fast Marching algorithm is developed. FM* combines the efficiency of the A* algorithm (described in the next section) with the accuracy of the Fast Marching algorithm.
- FM* allows the curvature of the final path to be constrained, which enables us to take the turning radius of any mobile robot into account. This property is formally proven for both isotropic and anisotropic medias.
- We show that ordered upwind methods based path planners enable the addition of directional constraints. We propose an anisotropic Fast Marching algorithm able to deal with smooth fields of force like underwater currents, but this concept can be generalized for any kind of directional constraint.
- We propose a multiresolution scheme to speed up the overall method in order to cope with real-time constraints of autonomous underwater path planning. This is achieved by implementing the Fast Marching algorithm on adaptive unstructured meshes.

C. Related work

A FM based path planner is proposed in [1] that allows dynamic replanning. No heuristic is introduced in this method to speed up the exploration process. The replanning ability counterbalances this lack of efficiency in the case of a-priori unknown terrain. The Field D* algorithm [2] is another approach, which is close to the FM* algorithm in the sense that it is an interpolated version of the D* algorithm (described in [3]) to the continuous domain. In practice the FM* algorithm is easier to implement and does not present the pathologic cases of the Field D* algorithm when the path is extracted using a gradient descent.

There are few literatures of path planning for autonomous underwater vehicles. Carroll et al. [4] used the A* algorithm. Warren [5] used a potential field method. This technique is fast but it may be affected by local minima. Techniques such as sequential quadratic programming [6], case-based reasoning [7] and genetic algorithms [8], [9], have also been applied to the motion planning of underwater robotic vehicles.

The issue of path planning under directional constraints has been addressed by Alvarez et al. In [10] the authors used genetic algorithms, which is an off-line technique even if they optimize it using dynamic programming. In [11] the A* algorithm is adapted to take current influence into account. This method is close in spirit to our anisotropic Fast Marching algorithm apart from the fact that it is based on a discrete motion model for the AUV.

D. Overview of the paper

The plan of the paper is as follows. Section II reviews the framework of our Fast Marching based planning technique. We rigorously present it as a part of the broader class of sampling based path planning methods. In this section it is shown that the Fast Marching algorithm is an extension of classical artificial intelligence algorithms to the continuous domain. Section III briefly details the Fast Marching algorithm and presents results using the FM* algorithm introduced in this paper. Section IV presents an anisotropic version of the Fast Marching algorithm to deal with directional constraints

such as currents. In section V the turning radius of the vehicle is introduced as a constraint to be respected along the path. Finally, section VI presents a multiresolution method to quickly extract sub-optimal trajectories.

II. Path planning framework

A. Environment representation

The uniform framework to study the path planning problem amongst static obstacles is the *configuration space* (*C-space*). The main idea of the C-space is to represent the robot as a point, called a *configuration*.

A robot configuration is a vector of parameters specifying position, orientation and all the characteristics of the robot in the environment. The C-space is the set of all possible configurations. We call *C-free* the regions of C-space which are free of static obstacles. Obstacles in the workspace become *C-obstacles* in the C-space. Usually a simple rigid body transformation [12] is used to map the real environment into the C-space.

B. Problem statement

Given a C-space Ω, the path planning problem is to find a curve

$$\begin{array}{rccl} C: & [0,1] & \rightarrow & \textit{C-free}, \\ & s & \mapsto & C(s) = (x(s), y(s)) \end{array}$$

An optimal path is a curve C that minimizes a set of internal and external constraints (time, fuel consumption or danger for instance). We assume in this paper that the complete set of constraints is described in a cost function τ, which can be isotropic or anisotropic.

- Isotropic case: the cost function τ depends only on the configuration x:

$$\begin{array}{rccl} \tau: & \Omega & \rightarrow & \mathbb{R}_+, \\ & x & \mapsto & \tau(x),\ \tau(x) > 0 \end{array}$$

- Anisotropic case: τ depends on the configuration x and a vector $\vec{F}$ of a field of force $\mathcal{F}$:

$$\begin{array}{rccl} \tau: & \Omega \times \mathcal{F} & \rightarrow & \mathbb{R}_+, \\ & (x, \vec{F}) & \mapsto & \tau(x, \vec{F}),\ \tau(x, \vec{F}) > 0. \end{array}$$

In sections II and III the C-space Ω is assumed to be isotropic[1]. Directional constraints are introduced in section IV.

C. Sampling based path planning methods

The main idea in sampling based path planning methods is to avoid a tedious construction of C-obstacles by sampling the C-space (see [13] for an exhaustive study of sampling based methods). The sampling scheme may be probabilistic (see [14] for a survey on probabilistic path planning) or may be deterministic (see [15] for a study of the relationship between the deterministic and the probabilistic approaches). We focus in this paper on the uniform Cartesian deterministic sampling scheme although we propose a non-uniform deterministic sampling scheme further in section VI. Cartesian sampling based methods are widely used in mobile robotics because they are suitable for sensor images mapped into a grid of pixels. The key issue is then to use an efficient grid search algorithm to find an optimal path in the sense of a metric.

1) Metric space: A metric ρ defines the distance between two configurations in the C-space, which becomes a metric space (see [13] for a rigorous definition of a metric space).

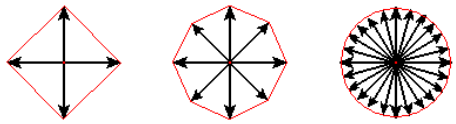
In this paper the metric ρ we refer to is defined as:

$$\rho(x, x') = \int_{[0,1]} \tau(C_{x,x'}(s))ds \tag{1}$$

where $C_{x,x'}$ is a path between two configurations x and x', and τ is a strictly positive cost function.

This metric can be seen as the "cost-to-go" for a specific robot to reach x' from x. At a configuration x, $\tau(x)$ can be interpreted as the cost of one step from x to its neighbors. If a C-obstacle in some region S is impenetrable, then $\tau(S)$ will be infinite. τ is supposed to be strictly positive for an obvious physical reason: $\tau(x) = 0$ would mean that free transportation from some configuration x is possible.

2) Motion models: Metric ρ is defined for a C-space assuming a continuous motion model. However, since the C-space is partitioned into a Cartesian grid, grid search algorithms commonly use a 4-connexity or a 8-connexity discrete motion model for the robot, see figure 1.



a. b. c.

Fig. 1. Three examples of motion models. a) 4-connexity, b) 8-connexity and c) continuous motion models. Cost-to-go level sets corresponding to these three motion models are respectively squares, octagons and circles.

A discrete approximation ρ_d of metric ρ is defined:

$$\rho_d(x, x') = \sum_{i=1}^{n} \tau(x_i) \tag{2}$$

where $x_1 = x$ and $x_n = x'$, and transitions between x_i and x_{i+1} are governed by a discrete motion model.

3) Grid search principle: A grid search algorithm is an optimization technique that performs successively an *exploration* and an *exploitation* process. The exploration process builds a "minimum cost-to-go" map, called *distance function*, from the start to the goal configuration. The exploitation process is a *backtracking* conversely from the goal to the start configuration.

The distance function $u : \Omega^2 \rightarrow R_+$ is the solution of the functional minimization problem defined as follows:

$$\begin{array}{rcl} u(x_{start}, x) & = & \inf_{\{C(x_{start},x)\}} \rho(x_{start}, x) \\ u(x_{start}) & = & 0 \end{array} \tag{3}$$

where $\{C(x_{start}, x)\}$ is the set of all curves between the source x_{start} and the current configuration x. For the sake

[1] We focus on 2D C-spaces in this paper; nonetheless this framework holds for C-spaces of any dimensions.

of notational simplicity, and assuming that the source of exploration x_{start} is fixed, we note $u(x_{start}, x) = u(x)$.

The distance function u can be related to the *value function* concept in reinforcement learning. The difference lies only on the fact that value functions are refined in an iterative process (called learning), whereas the distance function is built from scratch. In the path planning literature one can find other names for the distance function, such as navigation function [13], multi-valued distance map [16] or convex-map [17].

Once the distance function is found up through the goal configuration, the optimal path is the one which follows the steepest descent over the distance function from the goal to the start configuration. This backtracking technique is reliable as no local minima are exhibited during the exploration process.

4) Grid search algorithms: Grid search algorithms rely on a partitioning of the C-space in three sets: *Accepted* configurations for which the distance function u has been computed, *Considered* configurations for which u has been estimated and the remaining *Far* configurations for which u is unknown.

The set of *Considered* configurations is stored in a priority queue. On top of this queue the configuration with the highest priority is called *trial*. At each iteration of the exploration process the *trial* configuration is moved from *Considered* to *Accepted* and its *Far* neighbors are updated and moved from *Far* to *Considered*. The exploration process expands from the start configuration and ends when the goal configuration is eventually set to *Accepted*.

According to the priority assignment of the *Considered* set we distinguish two classes of grid search algorithms (see [18] for a survey of grid search algorithms).

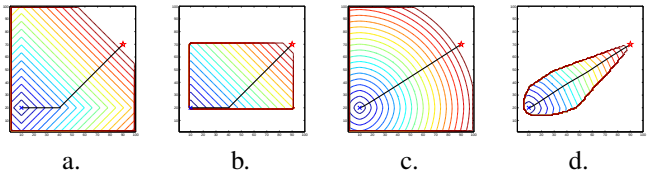
a) Breadth-first search: breadth-first algorithms (BF) give the highest priority to the *Considered* configuration x with the lowest estimate of the distance function u. $u(x)$ does not depend on the goal configuration, that is why the distance function is built symmetrically around the start configuration (see figure 2a).

b) Hybrid search: hybrid search algorithms (HS) gives the highest priority to the *Considered* configuration x with the lowest $f(x)$ defined as $f(x) = (1-\alpha)u(x) + \alpha h(x, x_{goal})$. Here $h(x, x_{goal})$ is an heuristic that estimates the residual distance between the configuration x and the goal configuration x_{goal}, and α is a constant ($0 < \alpha \leq 1$). If $\alpha = 0.5$, this popular HS algorithm is called A* algorithm. Instead of exploring around the start configuration HS algorithms focus the search towards the goal (see figure 2b).

5) Heuristic: When only sparse convex C-obstacles are present, heuristically guided searches are the most efficient techniques. These are the most commonly used heuristics. Let $x_{i,j}$ be the point of coordinates (i, j) on a grid:

- 4-connexity heuristics:

$$\begin{aligned} h_4(x_{i,j}, x'_{i',j'}) &= m(|i'-i| + |j'-j|) \\ h_{4_{\max}}(x_{i,j}, x'_{i',j'}) &= m(\max\{|i'-i|, |j'-j|\}) \end{aligned}$$



a. b. c. d.

Fig. 2. Examples of distance functions and paths using a 4-connexity: a) BF, b) A* using the "Manhattan distance" h_4 as an heuristic, c) FM and d) FM* algorithms. Paths computed using FM based algorithms are continuous. FM* exploration is well focused towards the goal point because of the natural choice of the Euclidean heuristic h_e in this case.

- Euclidean heuristic:

$$h_e(x_{i,j}, x'_{i',j'}) = m\left((i'-i)^2 + (j'-j)^2\right)^{\frac{1}{2}} \quad (4)$$

where $m = \min_\Omega\{\tau\}$.

6) Computational complexity: When a Far configuration is updated and moved to *Considered* a routine is called to sort the *Considered* set in such a way that the configuration with the highest priority remains the first of the queue. For optimal efficiency the priority queue is stored in a heap structure [19].

Assume that the *Considered* list is ordered. When a new element is inserted the price of reordering the list is $O(\log n)$, where n is the number of elements in the list. At each iteration of the exploration process the *Considered* list is sorted three times at the maximum in 2D (the four *trial*'s neighbors excluding at least one neighboring *Accepted* configuration). Considering that the exploration process is iterated N times at the maximum, where N is the total number of configurations in the grid, the computational complexity of grid search algorithms is $O(N \log(N))$.

7) Conclusions: First, resulting paths from grid search algorithms are discontinuous and not unique because the robot is supposed to follow a discrete motion model. Further, even when these paths are post optimized (see [20] for a survey of trajectory optimization techniques), they can still cause the vehicle to execute expensive trajectories.

Secondly, BF and HS algorithms are indeed efficient ($O(N \log(N))$ complexity) but they suffer from biases. Results from these discrete search algorithms can be improved by taking a larger neighborhood for the motion model, like a 8 or a 16-connexity motion model, giving better approximations ρ_d of ρ in diagonal directions. However, there will always be an error in some directions that will be invariant to the grid resolution. These limitations can be overcome using the Fast Marching algorithm presented in the next section.

III. Fast Marching based path planning

The Fast Marching algorithm was introduced in image processing by Sethian [21]. In path planning the main interest (developed in [22], [23]) of the Fast Marching algorithm relies on the fact that, with the same complexity as classical grid search algorithms, it extracts an accurate solution u of the functional minimization problem of expression 3.

A. Eikonal equation

Before introducing the Fast Marching algorithm itself, we start from the observation that the functional minimization problem of equation 3 is equivalent to solving the Eikonal equation:

$$\|\nabla u\| = \tau \tag{5}$$

We give here a geometrical intuition of how to convert equation 3 into equation 5. It is inspired by a level set formulation of the Eikonal equation in [24], nonetheless a formal proof can be found in [25].

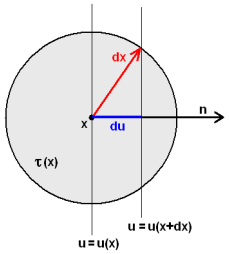


Fig. 3. On a small surface $d\Omega$ around a point x with a radius dx, one can approximate the distance function u as a plane wave, for which the level sets are parallel and perpendicular to the gradient ∇u of u. These observations allow the translation of the functional minimization problem of equation 3 into the Eikonal equation 5.

We start from the fact that the gradient ∇u of u is normal to its level sets. Let $\vec{n} = \frac{\nabla u}{\|\nabla u\|}$ be the outwards unit normal vector to level sets of u located in x. Express a variation du of u according to a variation $\vec{dx}$ of the position x (figure 3):

$$u(x + \vec{dx}) = u(x) + \langle \nabla u, \vec{dx} \rangle \Rightarrow du(x) = \langle \nabla u, \vec{dx} \rangle \tag{6}$$

where $\langle , \rangle$ is the standard dot product in $\mathbb{R}^2$.

Within the small region $d\Omega$ of Ω centered on x with a radius dx, we can assimilate τ as a constant, $\forall p \in d\Omega$, $\tau(p) = \tau(x) = \tau$. Within $d\Omega$ level sets of u are seen as straight lines:

$$du(x) = \tau \langle \vec{n}, \vec{dx} \rangle \tag{7}$$

From equations 6 and 7 we get $\langle \nabla u, \vec{dx} \rangle = \tau \frac{\langle \nabla u, \vec{dx} \rangle}{\|\nabla u\|}$, which leads to the Eikonal equation 5.

B. Upwind schemes and numerical approximations

The Fast Marching algorithm uses a first order numerical approximation of the Eikonal equation 5 based on the following operators. Suppose a function u is given with values $u_{i,j} = u(x_{i,j})$ on a Cartesian grid with grid spacing h.

- Forward operator (direction x): $D^{+x}_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{h}$
- Backward operator (direction x): $D^{-x}_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{h}$.

Forward and backward operators in direction y are similar.

The following upwind scheme (due to Godunov [26]) is used to estimate the gradient ∇u in two dimensions:

$$\|\nabla u_{ij}\|^2 \approx \begin{bmatrix} \max\{D^{-x}_{i,j}u, -D^{+x}_{i,j}u, 0\}^2 + \\ \max\{D^{-y}_{i,j}u, -D^{+y}_{i,j}u, 0\}^2 \end{bmatrix} = \tau^2_{i,j} \tag{8}$$

It is proven in [21] that this numerical scheme converges to a correct continuous solution[2].

C. Fast Marching algorithm

The Fast Marching algorithm belongs to the class of breadth-first algorithms. At each iteration the *trial* configuration x to be moved from *Considered* to *Accepted* is the one with the lowest estimate of the distance function u. Contrarily to classical breadth-first algorithms the *trial*'s neighbors $\{x_{i,j}\}$ may be updated more than once using the numerical scheme of equation 8 (see figure 4).

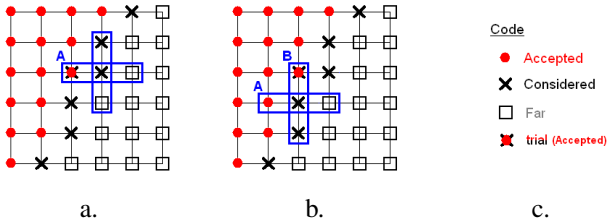


Fig. 4. To update a neighbor $x_{i,j}$ (which is *Far* or *Considered*, but not *Accepted*) of the *trial* point, we examine its 4-connexity neighborhood. a) Only one *Accepted* points around $x_{i,j}$, then we apply case 1. b) Two non-opposite *Accepted* points around $x_{i,j}$, then we apply case 2.

1) Algorithm for 2D isotropic Fast Marching: We give here an insight of the update procedure of the *trial*'s neighbors because we slightly modify it in our anisotropic version in section IV. For more implementation details see [21] and [24]. One or two *Accepted* points are used to solve equation 8:

- Case 1: only one *Accepted* or one pair A of opposite *Accepted* points around $x_{i,j}$ (figure 4a). Note $u_A = \min(u(A))$. In that case equation 8 is equivalent to

  $$(u_{i,j} - u_A)^2 = \tau^2_{i,j}$$

  which leads to

  $$u_{i,j} = u_A + \tau_{i,j}$$

- Case 2: at least two non-opposite *Accepted* points around x, they belong to two crossing pairs A and B (figure 4b). Note $u_A = \min(u(A))$ and $u_B = \min(u(B))$. In that case equation 8 is equivalent to

  $$(u_{i,j} - u_A)^2 + (u_{i,j} - u_B)^2 = \tau^2_{i,j}. \tag{9}$$

  Based on the discriminant test of equation 9 one or two *Accepted* points are used to solve it:

  - if $\tau_{i,j} > |u_A - u_B|$

    $$u_{i,j} = \frac{1}{2}\left(u_A + u_B + \sqrt{2\tau^2_{i,j} - (u_A - u_B)^2}\right)$$

  - else

    $$u_{i,j} = \min(u_A, u_B) + \tau_{i,j}$$

[2]This numerical scheme actually converges to the viscosity solution in the sense of Crandall and Lions [27].

2) FM:* The original Fast Marching algorithm uses the same priority assignment than breadth-first algorithms. We can introduce the Euclidean heuristic defined in expression 4 to speed up the exploration process. We call FM* this heuristically guided Fast Marching because it can be seen as the equivalent to the A* algorithm in the continuous domain.

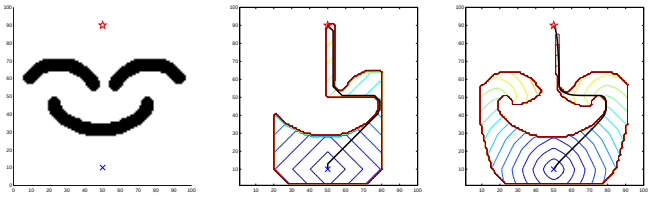
Note that if the cost function $\tau(x)$ is the inverse of the AUV velocity than the distance function is equivalent to the first arrival travel time t of the vehicle. Using a time dependent cost function $\tau(x,t)$ makes the FM* easy to extend for planning paths amongst moving obstacles.

3) Backtracking: The Fast Marching algorithm computes a first order estimate of the gradient of the distance function. The optimal path is naturally extracted from the goal to the start configuration by performing a gradient descent backtracking.

D. Isotropic Fast Marching applied to Underwater Path Planning

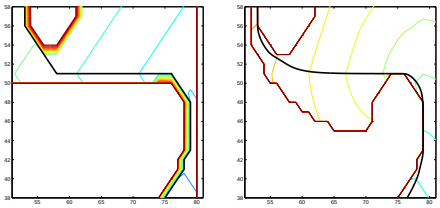
The isotropic Fast Marching and the FM* algorithms are validated here using Matlab and C++ simulations run with simulated and real underwater environments.

1) Non-convex obstacles: Non-convex obstacles are a problem for embedded path planning algorithms using potential field methods [12]. Figure 5 shows that the Fast Marching algorithm associated with a gradient descent naturally deals with the concavity since the start configuration is the only global minimum of the distance function u.



a. b. c.

Fig. 5. a) A binary cost function. b) Exploration and optimal path found using A* algorithm. c) Exploration and optimal path found using FM* algorithm.



a. b.

Fig. 6. A close-up on figures 5b and 5c. One can see that FM* algorithm gives a smoother solution (b) than A* algorithm (a). This is thanks to the better FM* estimate of the distance function in the continuous domain.

Note that to increase the safety distance between the path and the C-obstacles, a simple dilatation of the cost function is required.

2) Complex C-spaces: The computational complexity of Fast Marching based path planning is ($O(N \log N)$ where N is the number of pixels in the image. This efficiency is not affected by the complexity (i.e. number and shapes of C-obctacles) of the C-space. Figure 7 shows a harbor for which the main entrance is obstructed by a net. Fast Marching exploration naturally finds the little back entrance contrarily to probabilistic path planning methods which may not return any solution to this "narrow passage" problem [28].

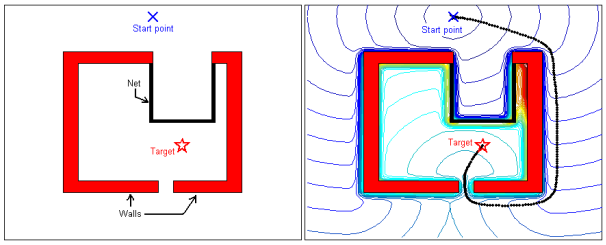


Fig. 7. A complex simulated cost map of a harbor obstructed by a net (on the left) and the optimal path found using the FM algorithm (on the right).

3) Real environment: A C++ implementation of the 2D Fast Marching based path planning method has been applied to real sonar images, see figure 8. These images have been processed *in-line* in the simulator of the Ocean Systems Laboratory.

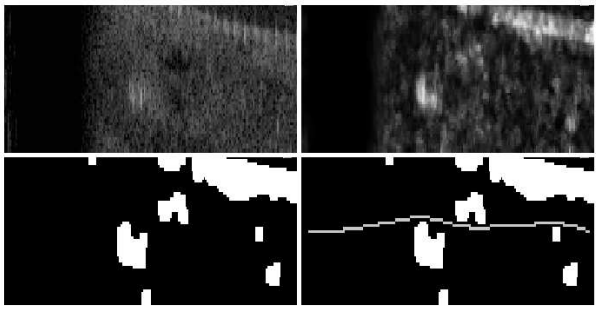


Fig. 8. From left to right and up to down: original sonar image, processed images (using a salient filter) and the optimal path. Sizes of these images are 200x50 points. Computation time is approximately 10 ms for the Fast Marching algorithm and 1 ms for the gradient descent algorithm.

IV. Directional constrained path planning

The Fast Marching method is a particular case of *ordered upwind methods* (OUM), which can be applied to isotropic or anisotropic medias. It is shown in [29] that the computational complexity of OUM methods is $O(\Upsilon N \log N)$, where N is the number of configurations and Υ is the anisotropic coefficient which can be much greater than one. Then OUMs are not fast any more.

The idea behind our anisotropic *Fast* Marching method is to simplify OUMs by considering the gradient ∇u as a good estimate of the wavefront propagation of the distance function u. This is equivalent to assuming that the field of force $\mathcal{F}$ is smooth. This way the $O(N \log N)$ complexity of the original Fast Marching algorithm is preserved in the anisotropic version. One can note a similar attempt in [30], but the authors do not analyze the complexity of their method.

A. Anisotropic cost function

The original Fast Marching algorithm is fast thanks to the resolution of the simple quadratic equation 8 for u. Since τ appears as a square in this equation our idea is to build a new cost function $\tilde{\tau}$ linearly dependent on u. We split the cost function in two parts: $\tilde{\tau} = \tau_{obst} + \tau_{vect}$. τ_{obst} is linked to the obstacles as previously and τ_{vect} is defined as follows:

$$\forall x_{i,j} \in \Omega,\ \tau_{vect}(i,j) = \alpha \left(1 - \frac{\langle \nabla u_{i,j} \cdot \vec{F_{i,j}} \rangle}{Q_{i,j}}\right) \geq 0 \quad (10)$$

where $Q_{i,j} = (\tau_{obst}(i,j) + 2\alpha) \sup_\Omega \left\{ \|\vec{F}\| \right\}$ is a normalization term so that $\left| \frac{\langle \nabla u \cdot \vec{F} \rangle}{Q} \right| \leq 1$, and α is a positive gain.

τ_{vect} models the external current forces applied to the vehicle. It is equivalent to say that a force favors the vehicle when both force and vehicle are pointing in the same direction.

Note that the concept can be generalized to path planning in any field of forces, like path planning for sailing applications, where $\mathcal{F}$ is a wind field for instance.

B. Anisotropic Fast Marching algorithm

The anisotropic version of the Fast Marching algorithm is similar to the isotropic version. A new quadratic equation is solved:

$$\|\nabla u_{ij}\|^2 \approx \left[\begin{array}{l} \max\{D_{i,j}^{-x}u, -D_{i,j}^{+x}u, 0\}^2 + \\ \max\{D_{i,j}^{-y}u, -D_{i,j}^{+y}u, 0\}^2 \end{array} \right] = \tilde{\tau}_{i,j}^2 \quad (11)$$

The *trial*'s neighbors $\{x_{i,j}\}$ are updated in 4-connexity (see figure 4). Let $\{A_i\}$ be the *Accepted* points around $x_{i,j}$ and let $\vec{F_{i,j}} = F_x \vec{u_x} + F_y \vec{u_y}$ be the vectorial force in $x_{i,j}$.

- Case 1: one *Accepted* point
  First, suppose that $x_{i,j}$ is surrounded only by one *Accepted* point, i.e. $\{A_i\} = \{A\}$.

  Equation 11 becomes $(u_{i,j} - u(A))^2 = \tilde{\tau}_{i,j}^2$.

  In this case $\nabla u_{i,j} = \pm (u_{i,j} - u(A)) \overrightarrow{u_{x\ or\ y}}$, then

  $$u_{i,j} = u(A) + \frac{\tau_{obst}(i,j) + \alpha}{1 + \frac{\alpha}{Q_{i,j}} F_{x\ or\ y}}$$

- Case 2: two *Accepted* points
  Suppose that $x_{i,j}$ is surrounded by two neighboring *Accepted* points, i.e. $\{A_i\} = \{A_1, A_2\}$. The case where A_1 and A_2 are opposite is farther considered.

  Equation 11 becomes:

  $$(u_{i,j} - u(A_1))^2 + (u_{i,j} - u(A_2))^2 = \tilde{\tau}_{i,j}^2 \quad (12)$$

  which leads to the following conditions:
  - if $|(u(A_2) - Lb) - (u(A_1) - La)| \leq L\sqrt{U}$

    $$u_{i,j} = \frac{-V + \sqrt{\Delta}}{2U}$$

  - else $u_{i,j} = \infty$

  where

  $$\left\{ \begin{array}{rcl} a & = & \frac{\alpha}{Q} F_x \\ b & = & \frac{\alpha}{Q} F_y \\ \lambda & = & \tau_{obst}(i,j) + \alpha \\ L & = & \frac{\lambda}{1-(a^2+b^2)} > 0 \\ \Delta & = & \frac{4\lambda}{L} \left[UL^2 - \left[(u(A_2) - Lb) - (u(A_1) - La) \right]^2 \right] \\ U & = & 2 - (a+b)^2 > 0 \\ V & = & 2u(A_1)(a^2 + ab - 1) + 2u(A_2)(b^2 + ab - 1) \\ & & +2\lambda(a+b) \end{array} \right.$$

  This is a generalized formulation of the isotropic Fast Marching equations. Conditions about $u(A_1)$ and $u(A_2)$ are respectively translated of La and Lb (related to F_x and F_y).

- Computation of u
  In the case of two neighboring *Accepted* points $\{A_i\} = \{A_1, A_2\}$ the computation of u needs three steps:
  - First we suppose that $x_{i,j}$ is surrounded only by one *Accepted* point, $\{A_i\} = \{A_1\}$. We apply the case 1 and the value u_1 is stored.
  - Secondly we suppose that $x_{i,j}$ is surrounded only by one *Accepted* point $\{A_i\} = \{A_2\}$. We apply a second time the case 1 and the value u_2 is stored.
  - Thirdly we suppose that $x_{i,j}$ is surrounded by two neighboring *Accepted* points, $\{A_i\} = \{A_1, A_2\}$. We apply the case 2 and the value u_3 is stored.

  Finally $u_{i,j} = \min\{u_1, u_2, u_3\}$ gives the correct viscosity solution u for the distance function. For the case where A_1 and A_2 are opposite we just skip the third step.

- Three or four *Accepted* points
  The method for three or four *Accepted* points around $x_{i,j}$ is exactly the same as for two *Accepted* points.
  - We apply the case 1 for each *Accepted* points and we store values of u.
  - We apply the case 2 for each couple of two consecutive *Accepted* points and we store values of u.

  Finally $u_{i,j} = \min\{u_k\}$.

C. Limitation and results

Note that we define $\tau_{vect} = \alpha \left(1 - \frac{\langle \nabla u \cdot \vec{F} \rangle}{Q}\right)$ for τ_{vect} to be linear for u. This choice allows small runtime but most of underwater vehicles have a more complex behavior than a linear reaction to currents. In that case an acceptable first order approximation of the AUV reaction to currents needs to be found.

Figures 9 and 10 show the influence of fields of force in underwater path planning ($\alpha = 1$, $\sup_\Omega \left\{ \|\vec{F}\| \right\} = 1$). The gain on the total cost of the paths computed using anisotropic FM algorithm is about 10% compared to the paths computed using isotropic FM for both figures. Runtime on these 100x100 figures is less than 1 second.

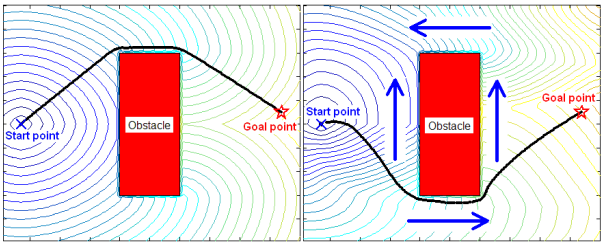


Fig. 9. On the left an isotropic Fast Marching, on the right our anisotropic version with currents symbolized with arrows. The path computed using the anisotropic FM leads to a 10% gain over the total cost of the trajectory.

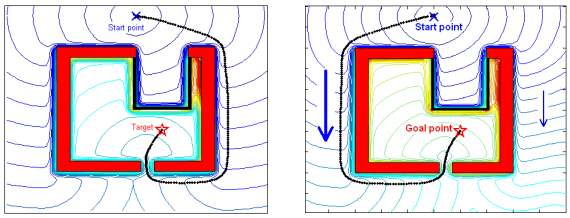


Fig. 10. Path found in absence of currents on the left, in presence of currents on the right. The path computed using the anisotropic FM leads to a 8% gain over the total cost of the trajectory in this case.

V. Curvature constrained path planning

In this section the influence of the cost function τ on the smoothness of a path C is demonstrated. Here C is the solution of the functional minimization problem:

$$\begin{array}{ccc} \Omega^2 & \rightarrow & \mathcal{C}(\Omega) \\ (x,x') & \mapsto & C = argmin_{\{C(x,x')\}}\rho(x,x') \end{array} \quad (13)$$

where $\mathcal{C}(\Omega)$ is the set of all curves in Ω and ρ is the continuous metric:

$$\rho(x,x') = \int_{[0,1]} \tau(C_{x,x'}(s))ds \quad (14)$$

Both isotropic and anisotropic Fast Marching methods compute a continuous solution C associated to the continuous metric ρ. Therefore, tools from differential geometry can be used to examine the curvature properties of C.

A. Problem statement

Let us define the kinematic constraints considered here.

- Curvature magnitude of a curve C: $k(C) = \frac{\partial^2 C}{\partial^2 s}$
- Curvature radius of a curve C: $R(C) = \frac{1}{|k(C)|}$
- lower bound of the curvature radius along a curve C: $R_{min}(C) = \inf_{s\in[0,1]} R(C(s))$.
- Turning radius of a vehicle v: $r(v)$

Our goal in this section is to express a formal link between the cost function τ and the lower bound of R_{min}. In other words we want to find a lower bound $R_{min}(\tau)$ before computing the distance function u (and before extracting any path).

B. Lower bound of the curvature radius

1) Isotropic case: Using the differential geometry framework, it is shown in [31] that the Euler-Lagrange equation associated to functional minimization of equation 13 is:

$$\tau k\vec{N} - < \nabla\tau, \vec{N} > \vec{N} = 0. \quad (15)$$

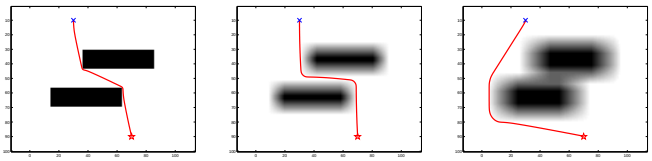
where $\vec{N}$ is the normal unit vector to a curve C and $\langle , \rangle$ is the standard dot product in $\mathbb{R}^2$.

From equation 15, it is deduced in [24] that the curvature magnitude k is bounded along any path C minimizing ρ. The lower bound for R_{min} is then:

$$R_{min} \geq \frac{\inf_\Omega \tau}{\sup_\Omega \{\|\nabla\tau\|\}} \quad (16)$$

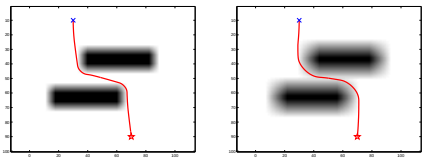
The conclusion is that to increase the lower bound of the curvature radius $R_{min}(C)$ of an optimal path C, two choices are possible (see figures 11 and 12 for illustrations):

1) smoothing the cost function to decrease $\sup_\Omega \{\|\nabla\tau\|\}$,
2) adding an offset to the cost function to increase the numerator $\inf_\Omega \tau$ without affecting the denominator.



a. b. c.

Fig. 11. Influence of smoothing the cost function.
a) A binary cost function τ (τ(C-free)=1 and τ(C-obst)=11) and the relative optimal path C_a, $R_{min}(C_a) = 332$ (in arbitrary unit).
b) τ after smoothing using a 11x11 average filter, $R_{min}(C_b) = 1216$.
c) τ after smoothing using a 21x21 average filter, $R_{min}(C_c) = 1377$.



a. b.

Fig. 12. Influence of both smoothing and adding an offset. The cost function τ is similar to the one in figure 11a.
a) Offset = 5, average filter 7x7, $R_{min}(C_a) = 1977$.
b) Offset = 5, average filter 15x15, $R_{min}(C_b) = 2787$.

At this stage experiments have to be carried out to study the applicability of our curvature constrained path planning technique to real AUV missions.

2) Anisotropic case: With our anisotropic cost function $\tilde{\tau}$ we show in the appendix that the Euler-Lagrange equation becomes:

$$\langle \nabla\tau_{obst}, \vec{N}\rangle\vec{N} + \frac{\alpha}{Q}\left(\frac{\partial F_x}{\partial y} - \frac{\partial F_y}{\partial x}\right)\vec{N} - \tilde{\tau}k\vec{N} = 0$$

From this equation, we derive the following lower bound of the path curvature radius:

$$R_{min} \geq \frac{\inf_\Omega\{\tau_{obst}\}}{\sup_\Omega \{\|\nabla\tau_{obst}\|\} + \frac{2\alpha}{\inf_\Omega\{Q\}}\|J_F\|_\infty} \quad (17)$$

where J_F is the Jacobian of $\vec{F}$ on Ω and $\|\cdot\|_\infty$ is the L_∞ norm.

Similarly to the isotropic case, there are three parameters to tune to increase the curvature radius $R_{min}(C)$ of an optimal path C:

1) smoothing τ_{obst},
2) adding an offset to $\tilde{\tau}$,
3) or smoothing the field of force $\vec{F}$ to decrease $\|J_F\|_\infty$.

VI. Multiresolution path planning

Multiresolution methods start with the idea that it is not necessary to represent the C-space in an uniform way. Some regions may be of more interest than others from a path planning point of view. For example a vast empty region may not need to be described as precisely as a region which contains many C-obstacles. This leads us to consider the partitioning of the environment as a crucial issue to optimize a path planning method.

A. Unstructured mesh framework

The method proposed by the authors is to couple a quadtree decomposition of the C-space with a Delaunay triangulation.

1) Quadtree decomposition: The quadtree decomposition (or octree decomposition in more than two dimensions) is based on a recursive decomposition of a uniform grid into blocks. The size of blocks can depend either on the information into them [32] or on their distance from the robot [33].

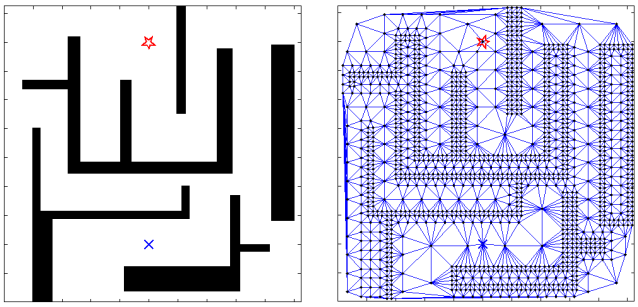
Quadtrees allow efficient partitioning of the environment since single blocks can be used to encode large empty regions. However, two main drawbacks remain. First, paths generated by quadtrees are suboptimal because they are constrained to segments between centers of blocks. The framed-quadtree technique [34] improves the situation but it is only applicable for sparse environments. Secondly, since the initial C-space is transformed in a tree data structure it is not easy to define the spatial neighborhood of each block. A simple solution is to test all the neighbors of each block [35].

2) Delaunay triangulation: The method proposed by the authors is to couple the quadtree decomposition with an adaptive mesh generation. The Delaunay triangulation is a good candidate as fast and robust implementations exist.

The input of the Delaunay mesh generation is the set of nodes q with their cost $\tau(q)$ given by the quadtree decomposition. The output is a net of vertices linked to their neighbors by edges, see figure 13. Versions of breadth-first and Fast Marching algorithms on this kind of unstructured mesh [36] are implemented in the following section.

B. Fast Marching algorithm on unstructured meshes

The original Fast Marching algorithm computes a distance function over a C-space sampled in a rectangular orthogonal mesh Ω in $O(N_{points} \log N_{points})$ steps, where N_{points} is the total number of grid points. The idea in this section is to perform the Fast Marching algorithm over the same C-space partitioned into a mesh using much less nodes. It is shown

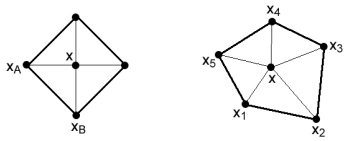


a. b.

Fig. 13. a) The original 1000x1000 image. b) The mesh computed from the quadtree decomposition using a Delaunay triangulation (1400 vertices). The links between the vertices of this mesh and their neighbors are well defined so that the propagation of the distance function from one vertex to its neighborhood is straightforward.

in [36] that an extension of the Fast Marching algorithm on unstructured meshes with N_{nodes} nodes has the same $O(N_{nodes} \log N_{nodes})$ complexity.

Because there is no natural choice of the coordinate system for an unstructured mesh (see figure 14), for each simplex $xx_1 \dots x_n$ we compute the gradient ∇u as a linear combination of the n directional derivatives along the unit vectors $T_i = \frac{x - x_i}{\|x - x_i\|}$, for $i = 1 \dots n$.



a. b.

Fig. 14. The original FM algorithm is implemented on a Cartesian grid (a), the update for the point x is computed from the simplex xx_Ax_B with the smallest values u_A and u_B. When implemented on an unstructured mesh (b) the update for the point x is computed from a pair of adjacent neighbors.

Let T be the nonsingular matrix having vectors T_i as its columns. Define $a = \frac{1}{\|x - x_i\|}$, $b = -\frac{u_i}{\|x - x_i\|}$ and $Q = (T^{\mathrm{T}}T)^{-1}$. Eikonal equation 5 becomes the following quadratic equation to solve for $u(x)$ (see [36] for further details):

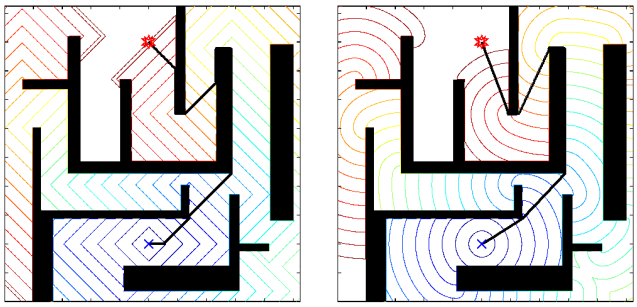
$$(a^T Q a)u^2 + (2a^T Q b)u + (b^T Q b) = \tau^2 \qquad (18)$$

C. Results

Figure 15 depicts the solution computed with a BF and a FM algorithm on the grid of figure 13a. Figure 16 depicts the solution computed with a BF and a FM algorithm on the mesh of figure 13b.

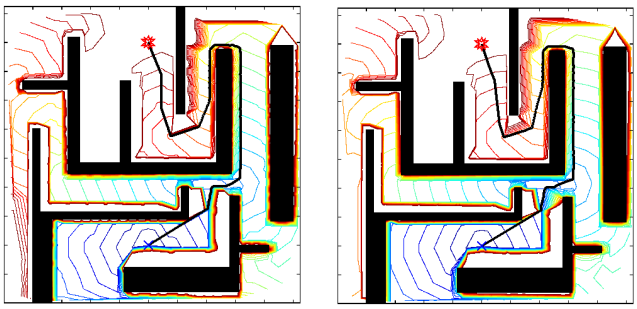
The multiresolution method gives only suboptimal paths. We partially overcome the problem of suboptimal paths by interpolating the distance function (computed on mesh vertices) on the entire underlying grid and by performing a gradient descent backtracking on this grid.

However, an interesting feature of both BF and FM on adaptive meshes compared to their implementation on uniform grids is their computation load.It is approximately divided by 1000 (on a Matlab platform), including the mesh generation.



a. b.

Fig. 15. Paths computed on the regular 1000x1000 grid on the original image (runtime is about 100 seconds): a) using a 4-connexity BF b) using a 4-connexity FM.



a. b.

Fig. 16. Paths computed on the 1400 vertices mesh: a) using BF b) using FM algorithms adapted on meshes. Both paths are suboptimal, but they are computed in 0.1 second.

Surprisingly BF gives "similar looking" paths than FM on the mesh. One could expect a better behavior of FM on the mesh thanks to its first order estimate of the distance function over the mesh. At this stage a tool has to be designed in order to compare the "quality" of a path. To the knowledge of the authors, no general criteria exist in the literature to objectively estimate the acceptability of a path for a given vehicle.

VII. Conclusion and Future Work

The underwater world is a very demanding environment for path planning algorithms. Great efforts are currently being made to develop autonomous systems as underwater technology becomes more mature. Several key issues for underwater path planning are addressed in this paper.

Reliability of path planners is improved by developing an efficient algorithm called FM* which is a continuous version of A* based on the Fast Marching algorithm. First, this algorithm is proved to find a continuous path when it is implemented on a discretized perception of the world. Secondly, a practical implementation of anisotropic Fast Marching is proposed to adapt our path planning method to underwater currents. Thirdly, a novel work is presented to take the vehicle kinematics into account for both isotropic and anisotropic environments. It is shown that smoothing the map of the environment leads to greater path curvature radius. Finally, a mesh conversion of the input data is used to drastically reduce the computation load by reducing the input data set of the path search algorithm. However, this reduction produces a loss of information that can affect the optimality of the resulting path. This highlights the need for future work to find an analytical tool to measure the path acceptability.

In all this work we assume that the world is static and described a-priori in a cost function. The authors are currently developing a dynamic version of the FM* in order to improve its replanning capacity in a-priori unknown real environments.

Appendix

Proof of the upper bound of the curvature magnitude for the anisotropic case

Given two points x and x' in a domain $\Omega \subset \mathbb{R}^2$ we consider the functional

$$E(C) = \int_{[0,1]} \tilde{\tau}(C(s))ds \tag{19}$$

to minimize over the following set of curves

$$C \in \mathcal{C}^2 : \begin{array}{ccc} [0,1] & \rightarrow & \Omega \\ s & \mapsto & [x(s), y(s)] \end{array}$$

where $\tilde{\tau} = \tau_{obst} + \tau_{vect} = \tau_{obst} + \alpha\left(1 - \frac{\langle \nabla C \cdot \vec{F} \rangle}{Q}\right)$ is the anisotropic strictly positive cost function[3]. We name $\mathcal{C}^2$ the set of curves derivable twice and s the arc-length parameter of C between x and x' ($C(0) = x$ and $C(1) = x'$).

We want to show that the Euler-Lagrange equation of this functional minimization problem is

$$\langle \nabla \tau_{obst}, \vec{N} \rangle \vec{N} + \frac{\alpha}{Q}\left(\frac{\partial F_x}{\partial y} - \frac{\partial F_y}{\partial x}\right)\vec{N} - \tilde{\tau} k \vec{N} = 0$$

where $\vec{N}$ is the normal unit vector to a curve C and $\langle , \rangle$ is the standard dot product in $\mathbb{R}^2$.

One can rewrite $E(C) = \int_{[0,1]} \tilde{\tau}(C(s))\|C'(s)\|ds$. Note that it is always possible to parameterize a curve $C \in \mathcal{C}^2$ so that $\forall s \in [0,1]$, $\|C'(s)\| = \|\frac{dC}{ds}(s)\| = 1$.

We consider the curve C which minimizes E. Let $\eta \in \mathcal{C}^2$ be a small deformation of C with the limit conditions $\eta(x) = \eta(x') = 0$ (figure 17).

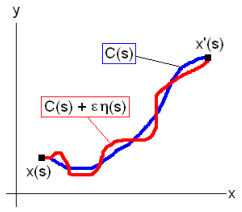


Fig. 17. The curve η is a small deformation of the curve C minimizing E.

We define $\phi(\epsilon) = E(C + \epsilon\eta)$, where $\epsilon > 0$. Then $\frac{d\phi}{d\epsilon} = 0$ when ϵ tends to 0. We suppose τ derivable on Ω, then:

$$\frac{d\phi}{d\epsilon} = \int_{[0,1]} \frac{d}{d\epsilon}\left[\tilde{\tau}\left(C + \epsilon\eta\right)\|C' + \epsilon\eta'\|\right] ds$$

First,

$$\begin{array}{rl} & \frac{d}{d\epsilon}\left[\tau_{obst}\left(C + \epsilon\eta\right) + \alpha\left(1 - \frac{\langle (C' + \epsilon\eta') \cdot \vec{F}(C + \epsilon\eta) \rangle}{Q}\right)\right] \\ = & \langle \nabla \tau_{obst}, \vec{\eta} \rangle - \frac{\alpha}{Q}\frac{d}{d\epsilon}\langle C' + \epsilon\eta', \vec{F}(C + \epsilon\eta) \rangle \end{array}$$

Secondly,

$$\begin{array}{rl} & \lim_{\epsilon \to 0}\left\{\frac{d}{d\epsilon}\langle C' + \epsilon\eta', \vec{F}(C + \epsilon\eta) \rangle\right\} \\ = & \langle \vec{F}, \vec{\eta'} \rangle + (\vec{T}^{\mathrm{T}} \cdot J_F) \cdot \vec{\eta} \end{array}$$

[3]Note that we replaced ∇u of expression 10 by $\nabla C = \vec{T}$ because C is the curve found after a gradient descent on u.

where $\vec{T}^{\mathrm{T}}$ is the transpose of the tangent vector to C and J_F is the Jacobian of $\vec{F}$ along the x and y coordinates.

Then,

$$\begin{aligned} & \lim_{\epsilon\to 0}\left\{\frac{d\phi}{d\epsilon}\right\} \\ = \; & \int_{[0,1]}\langle\nabla\tau_{obst},\vec{\eta}\rangle - \frac{\alpha}{Q}\langle\vec{F},\vec{\eta'}\rangle - \frac{\alpha}{Q}(\vec{T}^{\mathrm{T}}\cdot J_F)\cdot\vec{\eta} + \tilde{\tau}\langle\vec{T},\vec{\eta'}\rangle ds \end{aligned}$$

We integrate by part to get:

$$\int_{[0,1]}\tilde{\tau}\langle\vec{T},\vec{\eta'}\rangle ds = -\int_{[0,1]}\tilde{\tau}k\langle\vec{N},\vec{\eta}\rangle ds$$

and to get

$$\int_{[0,1]}\langle\vec{F},\vec{\eta'}\rangle ds = -\int_{[0,1]}\left[(J_F\cdot\vec{T})^{\mathrm{T}}\cdot\vec{\eta}\right] ds$$

Hence

$$\begin{aligned} & \lim_{\epsilon\to 0}\left\{\frac{d\phi}{d\epsilon}\right\} \\ = \; & \int_{[0,1]}\left\langle\left(\nabla\tau_{obst} + \frac{\alpha}{Q}\left(J_F^{\mathrm{T}} - J_F\right)^{\mathrm{T}}\cdot\vec{T} - \tilde{\tau}k\vec{N}\right),\vec{\eta}\right\rangle ds \end{aligned}$$

Noticing that $\forall\vec{\eta},\ \frac{d\phi}{d\epsilon} = 0$
and that

$$\left(J_F^{\mathrm{T}} - J_F\right)^{\mathrm{T}}\cdot\vec{T} = \left(\frac{\partial F_x}{\partial y} - \frac{\partial F_y}{\partial x}\right)\vec{N}$$

we get the Euler-Lagrange equation

$$\langle\nabla\tau_{obst},\vec{N}\rangle\vec{N} + \frac{\alpha}{Q}\left(\frac{\partial F_x}{\partial y} - \frac{\partial F_y}{\partial x}\right)\vec{N} - \tilde{\tau}k\vec{N} = 0 \tag{20}$$

Finally we derive an upper bound of the curvature magnitude:

$$|k| \leq \frac{\sup_\Omega\{\|\nabla\tau_{obst}\|\} + \frac{2\alpha}{\inf_\Omega\{Q\}}\|J_F\|_\infty}{\inf_\Omega\{\tau_{obst}\}} \tag{21}$$

REFERENCES

[1] R. Philippsen and R. Siegwart, "An interpolated dynamic navigation function," in *Proceedings IEEE Conference on Robotics and Automation (ICRA 2005)*, 2005.

[2] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The Field D* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79–101, 2006.

[3] A. Stentz, "Optimal and efficient path planning for partially-known environment," in *IEEE International Conference on Robotics and Automation*, May 1994.

[4] K. P. Carroll, S. R. McClaran, E. L. Nelson, D. M. Barnett, D. K. Friesen, and G. N. Williams, "AUV path planning: An A* approach," in *Proceedings of the Symposium on AUV Technology (AUV '92)*, 1992, pp. 3–8.

[5] C. W. Warren, "A technique for autonomous underwater vehicle route planning," in *IEEE Journal of Oceanic Engineering*, vol. 15, 1990, pp. 199–204.

[6] Y. Petillot, I. T. Ruiz, and D. Lane, "Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar," in *IEEE Journal of Oceanic Engineering*, vol. 26, no. 2, April 2001, pp. 240–251.

[7] C. Vasudevan and K. Ganesan, "Case-based path planning for autonomous underwater vehicles," *Autonomous Robots*, pp. 79–89, 1996.

[8] K. Sugihara and J. Yuh, "Ga-based motion planning for underwater robotic vehicle," in *Proceedings of the 10th International Symposium on Unmanned Untethered Submersible Technology*, 1997, pp. 406–415.

[9] R. Fox, A. Garcia, and M. L. Nelson, "A three dimensional path planning algorithm for autonomous vehicles," in *Proceedings of the 11th International Symposium on Unmanned Untethered Submersible Technology*, 1999, pp. 546–556.

[10] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary path planning for autonomous underwater vehicles in a variable ocean," in *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, April 2004, pp. 418–429.

[11] B. Garau, A. Alvarez, and G. Oliver, "Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach," in *Proceedings of the IEEE Conference on Robotics and Automation (ICRA 2005)*, Barcelona, Spain, April 2005, pp. 546–556.

[12] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publisher, 1991.

[13] S. M. LaValle, *Planning Algorithms*, University of Illinois, 2006.

[14] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion : Theory, Algorithms, and Implementations*. The MIT Press, 2005.

[15] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *Internationnal Journal of Robotic Research*, vol. 23, no. 7, pp. 673–692, July-August 2004.

[16] R. Kimmel, N. Kiryati, and A. Bruckstein, "Multi-valued distance maps in finding shortest paths between moving obstacles," *IEEE Trans. on Robotics & Automation*, vol. 3, no. 14, pp. 427–436, 1998.

[17] P. Melchior, B. Orsoni, O. Lavialle, A. Poty, and A. Oustaloup, "Consideration of obstacle danger level in path planning using A* and fast-marching optimization: comparative study," *Signal Processing*, no. 83, pp. 2387–2396, 2003.

[18] R. E. Korf, "Artificial intelligence search algorithms," *CRC Handbook of Algorithms and Theory of Computation*, pp. 36–1–36–20, 1998.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[20] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, March-April 1998.

[21] J. A. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge, Massachusetts: Cambridge University Press, 1999.

[22] C. Petres, Y. Pailhas, J. Evans, Y. Petillot, and D. Lane, "Underwater path planning using fast marching algorithms," in *Proceedings of Ocean 2005 - Europe*, vol. 2, Brest, France, June 2005, pp. 814–819.

[23] C. Petres and P. Patron, "Path planning for unmanned underwater vehicles," in *Proceedings of the IJCAI 2005 Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains*, Edinburgh, UK, August 2005.

[24] L. Cohen and R. Kimmel, "Global minimum for active contour models: A minimal path approach," *Internationnal Journal of Computer Vision*, vol. 24, no. 1, pp. 57–78, 1997.

[25] A. M. Bruckstein, "On shape from shading," in *Computer Vision, Graphics and Image processing*, vol. 44, 1988, pp. 139–154.

[26] E. Rouy and A. Tourin, "A viscosity solution approach to shape-from-shading," *SIAM Journal of Numerical Analyzis*, vol. 29, pp. 867–884, 1992.

[27] M. G. Crandall, L. C. Evans, and P. L. Lions, "Some properties of viscosity solutions of hamilton-jacobi equations," *Trans. Amer. Math. Soc.*, no. 282, pp. 487–502, 1984.

[28] P. Svestka and M. Overmars, "Probabilistic path planning," in *Lecture Notes in Control and Information Sciences 229*. Springer, 1998.

[29] A. Vladimirsky, "Ordered upwind methods for static hamilton-jacobi equation: Theory and algorithms," *SIAM Journal of Numerical Analysis*, vol. 41, no. 1, pp. 325–363, 2003.

[30] Q. Lin, *Enhancement, Extraction, and Visualization of 3D Volume Data*. Linkopings Universitet, Sweden: PhD thesis, 2003.

[31] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," in *Proceedings of 5th IEEE International Conference on Computer Vision (ICCV'95)*, Cambridge, USA, 1995.

[32] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE journal of Robotics and Automation*, vol. RA-2, no. 3, pp. 135–145, September 1986.

[33] S. Behnke, "Local multiresolution path planning," in *Proceedings of 7th RoboCup International Symposium*, Padua, Italy, 2004.

[34] A. Yahja, A. Stentz, S. Singh, and B. Brumitt, "Framed-quadtree path planning for mobile robots operating in sparse environments," in *Proceedings IEEE Conference on Robotics and Automation (ICRA '98)*, Leuven, Belgium, May 1998.

[35] D. Ferguson and A. Stentz, "Multi-resolution field D*," in *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, 2006.

[36] J. A. Sethian and A. Vladimirsky, "Fast methods for the eikonal and related hamilton-jacobi equations on ustructured meshes," *Applied Mathematics*, vol. 97, no. 11, pp. 5699–5703, May 23 2000.