We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

21

Trajectory Planning for Autonomous Underwater Vehicles

Clement Petres[1], Yan Pailhas[2], Pedro Patron[2], Jonathan Evans[2], Yvan Petillot[2] and Dave Lane[2]
[1]*CEA-LIST,*
[2]*Heriot-Watt University, Ocean Systems Laboratory,*
[1]*France*
[2]*Scotland*

1. Introduction

1.1 Trajectory planning

This chapter is a contribution to the field of Artificial Intelligence. Artificial Intelligence can be defined as the study of methods by which a computer can simulate aspects of human intelligence (Moravec, 2003). Among many mental capabilities, a human being is able to find his own path in a given environment and to optimize it according to the situation requirements. For an autonomous mobile robot, the computation of a safe trajectory is crucial for the success of a mission. Here is the ultimate goal of the trajectory planning issue for autonomous robots:

given a set of internal and external constraints from the robot capabilities and from the environment what is the best trajectory solution to reach a given target?

This is the problem we want to solve in this chapter. For this purpose, a novel approach is developed which is inspired from a level set method that originally emerged within the image processing community. This method, called Fast Marching (FM) algorithm, is analyzed and extended to improve the trajectory planning process for mobile robots. Theory and algorithms hold for any kind of autonomous mobile robot. Nonetheless, since this research work has been supported by the Oceans Systems Laboratory, the trajectory planning methods are applied to the underwater environment. Simulations and results are given assuming the use of an autonomous underwater vehicle (AUV).

1.2 Underwater environment and autonomous underwater vehicles

In mobile robotics, trajectory planning research has focussed on wheeled robots moving on 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 three dimensional workspace may be worldwide. Moreover, torpedo-like vehicles are strongly nonholonomic.
The current state of technology allows many laboratories such as the Oceans Systems Laboratory to move forward in the development of AUVs. The need for a reliable cognition process for finding a feasible trajectory derived from underwater imagery is important.

Source: Underwater Vehicles, Book edited by: Alexander V. Inzartsev,
ISBN 978-953-7619-49-7, pp. 582, December 2008, I-Tech, Vienna, Austria

1.3 Contributions

The main contribution of the authors is to present a Fast Marching based method as an advanced tool for underwater trajectory planning (Petres et al., 2007). With a similar complexity to classical graph-search techniques in Artificial Intelligence, the Fast Marching algorithm converges to a smooth solution in the continuous domain even when it is implemented on a sampled environment. This specificity is crucial to the understanding of the other contributions of our method:

- FM* algorithm: we develop a new algorithm called FM* that is a heuristically guided version of the Fast Marching algorithm. The FM* algorithm combines the efficiency of the A* algorithm (Hart, 1968) with the accuracy of the Fast Marching algorithm (Sethian, 1999).
- Curvature constrained trajectory planning: the FM* algorithm allows the curvature of the trajectory solution to be constrained, which enables us to take the turning radius of any mobile robot into account.
- Dynamic and partially-known domains: a dynamic version of the Fast Marching algorithm, called DFM, is proposed to deal with dynamic environments. DFM algorithm is then proved to be very efficient to recompute trajectories after minor changes in the robot perception of the world.
- Simulations and open water trials: a complete architecture has been designed, developed and tested for simulated and real AUV missions. In-water experiments are compared to simulation results to demonstrate the performance and usefulness of the DFM-based trajectory planning approach in the real world.

2. Trajectory planning framework

2.1 Environment representation

The usual framework to study the trajectory planning problem among static or dynamic 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. Its dimension is the number of parameters that defines a configuration. *C-free* is the set of configurations that are free of obstacles. Obstacles in the workspace become *C-obstacles* in the C-space.

Usually a simple rigid body transformation (Latombe, 1991) is used to map the real environment into the C-space. We focus on 2D and 3D C-spaces in this chapter, nonetheless this framework holds for C-spaces of any dimensions.

2.2 Problem statement

Given a C-space Ω, planning a trajectory is finding a curve

$$C : \begin{matrix} [0,1] \to C - \text{free} \\ s \mapsto C(s) \end{matrix} \tag{1}$$

where [0,1] is the parameterization interval and s is the arc-length parameter of C. If x_{start} and x_{goal} are the start and the goal configurations respectively, then $C(0) = x_{start}$ and $C(1) = x_{goal}$.

An optimal trajectory is a curve C that minimizes a set of internal and external constraints (time, fuel consumption or danger for instance). It is assumed in this chapter that the complete set of constraints is described in a cost function τ:

$$\tau : \begin{matrix} \Omega \rightarrow \Re \\ x \mapsto \tau(x) \end{matrix} \tag{2}$$

2.3 Metric space

In this chapter the metric space Ω we refer to is the usual C-space equipped with the metric ρ defined as:

$$\rho(x_1, x_2) = \int_{[0,1]} \tau\left(C_{x_1, x_2}(s) \right) ds \tag{3}$$

where C_{x_1, x_2} is a trajectory between two configurations x_1 and x_2, and τ is the cost function. This metric can be seen as the "cost-to-go" for a specific robot to reach x_2 from x_1. At a configuration x, τ(x) can be interpreted as the cost of one step from x to its neighbours. If a C-obstacle in some region S is impenetrable, then τ(S) will be infinite. The function τ is supposed to be strictly positive for an obvious physical reason: τ(x) = 0 would mean that free transportation from some configuration x is possible.

2.4 Distance function concept

A grid-search algorithm aims at building a distance function $u : \Omega^2 \rightarrow \Re$, which is solution of the functional minimization problem defined as follows:

$$u(x_{start}, x) = \inf_{\{C_{x_{start}, x}\}} \rho(x_{start}, x) \tag{4}$$

where $\{C_{x_{start}, x}\}$ is the set of all the possible curves between the source x_{start} and the current configuration x within Ω. For the sake 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 may be related to the *value function* concept in reinforcement learning. The difference lies only in 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 (LaValle, 2006), convex-map (Melchior et al., 2003) or multi-valued distance map (Kimmel et al., 1998).

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

3. Fast marching based trajectory planning

3.1 Related previous work

A method for computing consistent distance functions in the continuous domain was first proposed in (Tsitsiklis, 1995) but the method of the author is less efficient than the Fast

Marching method (Sethian, 1999). A FM based trajectory planning method among moving obstacles has been proposed in (Kimmel et al., 1998). The Fast Marching algorithm has also been applied in trajectory planning in (Melchior et al., 2003), where the authors compare A* and FM efficiencies among static obstacles. In (Philippsen & Siegwart, 2005), the authors develop a FM based trajectory planning method that allows dynamic replanning and improves Fast Marching efficiency in the case of a-priori unknown or dynamic domains. All these works are close in spirit to what we describe in this chapter except for the fact that we introduce a heuristic in a novel FM* algorithm to speed up the exploration process.

3.2 Eikonal equation

Before introducing the Fast Marching algorithm itself, we start from the observation that the functional minimization problem (4) is equivalent to solving the Eikonal equation:

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

We give here a geometrical intuition in two dimensions of how to convert equation (4) into equation (5). It is inspired by a level set formulation of the Eikonal equation in (Cohen & Kimmel, 1997) and a formal proof can be found in (Bruckstein, 1988).

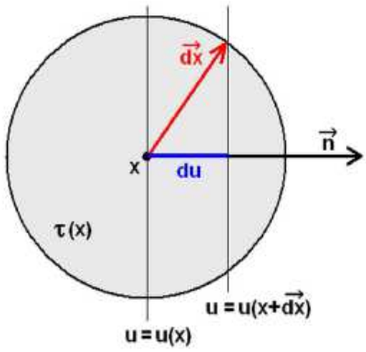

Fig. 1. On a small surface dΩ around a configuration x with a radius dx, one can approximate the distance function u as a plane wave, for which the level sets are parallel between them and perpendicular to the gradient ∇u of u.

We start from the fact that the gradient ∇u of u is normal to its level sets. Let $\vec{n} = \nabla u / \|\nabla u\|$, where $\|.\|$ is the Euclidean norm, be the outwards unit normal vector to level sets of u located in x (see figure 1). Express a variation du of u according to a variation $\overrightarrow{dx}$ of the position x:

$$u\left(x + \overrightarrow{dx}\right) = u(x) + \left\langle \nabla u, \overrightarrow{dx} \right\rangle$$

$$du(x) = \left\langle \nabla u, \overrightarrow{dx} \right\rangle \tag{6}$$

where $\langle .,. \rangle$ is the standard dot product in $\Re^2$.

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:

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

From equations (6) and (7) we get $\left\langle \nabla u, \overrightarrow{dx} \right\rangle = \tau\left\langle \nabla u, \overrightarrow{dx} \right\rangle / \|\nabla u\|$, which leads to the Eikonal equation (5).

3.3 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 $u_{i,j,k} = u(x_{i,j,k})$ on a 3D Cartesian grid with grid spacing h.

- Forward operator (direction i): $D^{+i}_{i,j,k}(u) = \left(u_{i+1,j,k} - u_{i,j,k}\right)/h$
- Backward operator (direction i): $D^{-i}_{i,j,k}(u) = \left(u_{i,j,k} - u_{i-1,j,k}\right)/h$

Forward and backward operators in directions j and k are similar.
The following upwind scheme, originally due to Godunov (Godunov, 1969) and well explained in (Rouy & Tourin, 1992) and in (Sethian, 1999), is used to estimate the gradient ∇u in three dimensions:

$$\begin{aligned} &\max\left(D^{-i}_{i,j,k}(u), D^{+i}_{i,j,k}(u), 0\right)^2 \\ &+ \max\left(D^{-j}_{i,j,k}(u), D^{+j}_{i,j,k}(u), 0\right)^2 = \tau^2_{i,j,k} \\ &+ \max\left(D^{-k}_{i,j,k}(u), D^{+k}_{i,j,k}(u), 0\right)^2 \end{aligned} \tag{8}$$

where $\tau_{i,j,k} = \tau\left(x_{i,j,k}\right)$.

3.4 Fast Marching algorithm

3.4.1 Pseudo-code

The pseudo code of the Fast Marching algorithm is given in table 1. The FM algorithm relies on a partitioning of the C-space in three sets: *Accepted* configurations for which the distance function u has been computed and frozen, *Current* configurations for which an estimate v of u has been estimated (and not frozen), and the remaining *Unvisited* configurations for which u is unknown.

Definitions

Start is the set of start configurations;
Goal is the set of goal configurations;
Neigh(S) is the set of neighbours of a set of configurations S;
x_{top} is the configuration in priority queue Current with the highest priority.

Procedure **Initialization()**

{01} Accepted = Start, u(Accepted) = 0;
{02} Unvisited = $\Omega \setminus$ Accepted, u(Unvisited) = v(Unvisited) = ∞;
{03} Current = Neigh(Start), v(Current) = τ(Current);

Procedure **Main()**

{04} Loop : while Goal $\not\subset$ Accepted
{05} Remove x_{top} from Current and insert it in Accepted with $u(x_{top}) = v(x_{top})$;
{06} FMComputeV(Neigh(x_{top}));

Table 1. Pseudo code of the Fast Marching algorithm

The set of Current configurations is stored in a priority queue. On top of this queue the configuration with the highest priority is called x_{top}. At each iteration of the exploration process, x_{top} is moved from Current to Accepted and its Unvisited neighbours are updated and moved from Unvisited to Current. The exploration process expands from the start configuration and ends when the goal configuration is eventually set to Accepted.

3.4.2 Computation procedure

The computation procedure for the 3D Fast Marching algorithm described in table 2 can be found in (Deschamps & Cohen, 2001). We give here additional calculation details to update the distance function estimate v_k of an x_{top}'s neighbour x_k with a cost τ_k.

Procedure **FMComputeV(Neigh(x_{top}))**
01} Loop : for all configurations x_k in Neigh(x_{top}) {02} If x_k is Unvisited, then remove it from Unvisited and insert it in Current with $v_k = \infty$ {03} If x_k is Current then apply case 1 or case 2 for the computation of v_k. {04} Sort Current list according to the priority assignment.

Table 2. Pseudo code of the FM procedure for updating Neigh(x_{top})

One, two or three Accepted configurations are used to solve equation (8). We note $\{A_1, A_2\}$, $\{B_1, B_2\}$ and $\{C_1, C_2\}$ the three couples of opposite neighbours of x_k (in 6-connexity) with the ordering $u(A_1) \le u(A_2)$, $u(B_1) \le u(B_2)$, $u(C_1) \le u(C_2)$ and $u(A_1) \le u(B_1) \le u(C_1)$. Two different cases are to be examined sequentially:

Case 1: considering that $v_k \ge u(C_1) \ge u(B_1) \ge u(A_1)$, the upwind scheme (8) is equivalent to:

$$(v_k - u(A_1))^2 + (v_k - u(B_1))^2 + (v_k - u(C_1))^2 = \tau_k^2 \tag{9}$$

Computing the discriminant of equation (9) there are two possibilities:

- if $\tau_k^2 > u(A_1)^2 + u(B_1)^2 + 2u(C_1)(u(C_1) - u(B_1) - u(A_1))$ and $u(B_1) < \infty$

$$v_k = \frac{1}{3}(u(A_1) + u(B_1) + u(C_1)) + \frac{1}{3}\sqrt{(3\tau_k^2 - 2(u(A_1)^2 + u(B_1)^2 + u(C_1)^2) - u(A_1)u(B_1) - u(A_1)u(C_1) - u(B_1)u(C_1))} \tag{10}$$

- else
  Go to case 2

Case 2: considering that $v_k \ge u(B_1) \ge u(A_1)$ and $v_k < u(C_1)$, the upwind scheme (8) is equivalent to:

$$(v_k - u(A_1))^2 + (v_k - u(B_1))^2 = \tau_k^2 \tag{11}$$

Computing the discriminant of equation (11) there are two possibilities:

- if $\tau_k > u(B_1) - u(A_1)$

$$v_k = \frac{1}{2}(u(A_1) + u(B_1)) + \frac{1}{2}\sqrt{2\tau^2 - (u(B_1) - u(A_1))^2} \tag{12}$$

- else

$$v_k = u(A_1) + \tau_k \tag{13}$$

Note that case 2 is similar to the update procedure of the 2D Fast Marching (Sethian, 1999).

3.5 FM* algorithm

In the Fast Marching algorithm the highest priority is assigned to the Current configuration x_{top} with the lowest estimate $e_{top} = v(x_{top})$, see table 3.

x_{top} (e_{top})	x_1 (e_1)	x_2 (e_2)	…	x_N (e_N)

Table 3. List of Current configurations stored in a priority queue. The highest priority is given the to lowest estimate e: $e_{top} < e_1 < e_2 < \ldots < e_N$.

Since u(x) does not depend on the goal configuration, the distance function u is built symmetrically around the start configuration, see figure 2.a. In this figure, distance maps and trajectories have been computed over a constant cost map. We use cool colours for small distances and hot colours for high distances (in arbitrary units).

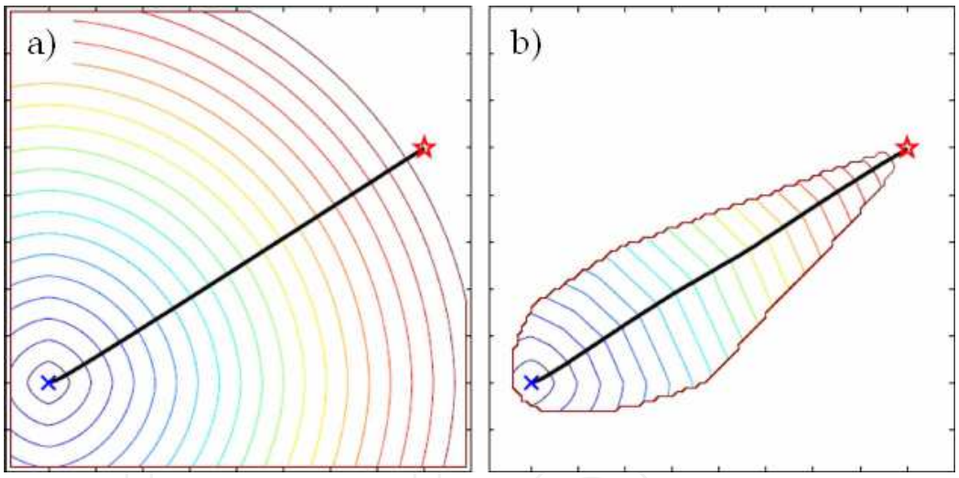

Fig. 2. Examples of distance maps and trajectories computed over a constant 100x100 cost map ($\tau = 1$) using a 4-connexity: a) FM algorithm and b) FM* algorithm (using the Euclidean distance H_e as a heuristic).

In the FM* algorithm the highest priority is assigned to the Current configuration x_{top} with the lowest estimate $e_{top} = \frac{1}{2} v(x_{top}) + \frac{1}{2} H_e(x_{top}, x_{goal})$. Here $H_e(x_{top}, x_{goal})$ is the *heuristic* that estimates the residual distance between the Current configuration x_{top} and the goal configuration x_{goal}. Similarly to the A* algorithm, instead of exploring around the start configuration, the FM* algorithm focuses the search towards the goal configuration, see figure 2.b.

Bi-directional versions of these grid-search algorithms can also be implemented. We just have to launch the grid-search algorithm simultaneously from the start and the goal configurations. We stop it when the two sets of Accepted configurations are merging.

4. Curvature constrained trajectory planning

In this section, differential constraints are reduced to curvature constraints. A Fast Marching based fully coupled approach (Petres et al., 2007) is proposed that ensures the trajectory solution to be smooth enough for an AUV with a given turning radius.

4.1 Problem statement

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

$$\begin{array}{c} \Omega^2 \to \widetilde{\Omega} \\ (x_1, x_2) \mapsto C = \operatorname{argmin}_{\{C_{x_1,x_2}\}} \rho(x_1, x_2) \end{array} \tag{14}$$

where $\widetilde{\Omega}$ is the set of all the possible curves in Ω, $\{C_{x_1,x_2}\}$ is the set of all the possible curves in Ω between x_1 and x_2 and ρ is the continuous metric:

$$\rho(x_1, x_2) = \int_{[0,1]} \tau\left(C_{x_1,x_2}(s)\right) ds \tag{15}$$

The Fast Marching method computes a derivable solution C associated with the continuous metric ρ. Therefore, tools from differential geometry can be used to examine the curvature properties of C.

Let us define the curvature parameters considered here.

- Curvature magnitude of a curve C: $k(C) = \dfrac{\partial^2 C}{\partial^2 s}$
- Curvature radius of a curve C: $R(C) = \dfrac{1}{|k(C)|}$
- Lower bound on 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)

4.2 Lower bound on the curvature radius

Given a cost function τ, our goal is to insure the feasibility of any trajectory C for an AUV v before computing the distance function u. Mathematically speaking, we want $\forall (x_1, x_2) \in \Omega^2, R_{min}(C) > r(v)$ knowing that $C = \operatorname{argmin}_{\{C_{x_1,x_2}\}} \rho(x_1, x_2)$. For this purpose we will express a formal link between the cost function τ and the lower bound $R_{min}(C)$ for any curve C minimizing the metric ρ between two configurations.

Using the differential geometry framework, it is shown in (Caselles et al., 1997) that the Euler-Lagrange equation associated with the functional minimization (14) is:

$$\tau k \vec{N} - \left\langle \nabla \tau, \vec{N} \right\rangle \vec{N} = 0 \tag{16}$$

where $\vec{N}$ is the normal unit vector to a curve C.

From equation (16), it is deduced in (Cohen & Kimmel, 1997) that the curvature magnitude k is bounded along any curve C minimizing ρ. The lower bound R_{min} is then:

$$R_{min} \geq \frac{\inf_{\Omega} \tau}{\sup_{\Omega} \{\|\nabla \tau\|\}} \tag{17}$$

The conclusion is that to increase the lower bound on the curvature radius $R_{min}(C)$ of an optimal trajectory C, two choices are possible:

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

The following illustrations depict some trajectories computed using the FM* algorithm after smoothing the cost map (figure 3) and after smoothing the cost map and adding an offset (figure 4).

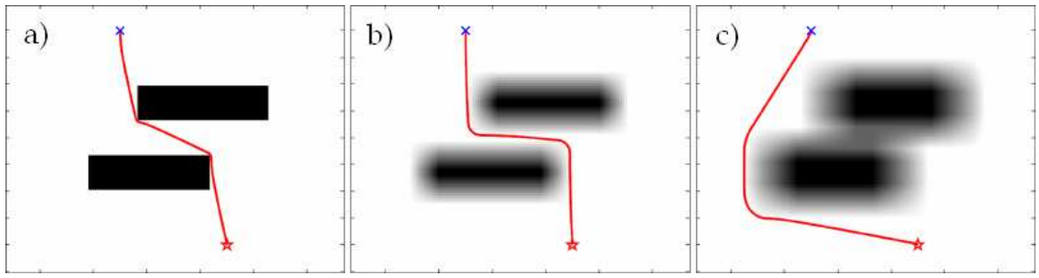

Fig. 3. Influence of smoothing the cost function. a) A binary 100x100 cost function τ, τ(C-free) = 1 and τ(C-obstacles) = 11 and the related optimal trajectory C_a, $R_{min}(C_a)$ = 332 (in arbitrary units). b) τ after smoothing using a 11x11 average filter, $R_{min}(C_b)$ = 1216. c) τ after smoothing using a 21x21 average filter, $R_{min}(C_c)$ = 1377.

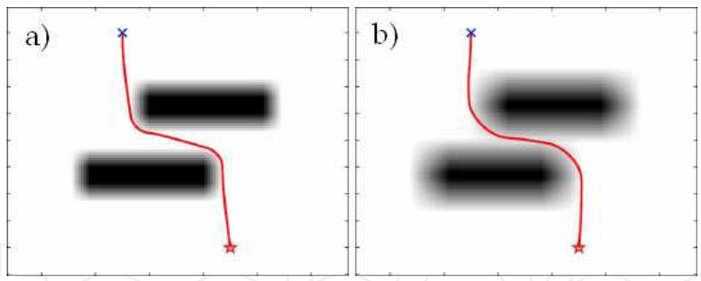

Fig. 4. Influence of both smoothing and adding an offset. The original cost function τ is similar to the one in figure 3.a. a) Offset = 5, average filter 7x7, $R_{min}(C_a)$ = 1977 (in arbitrary units). b) Offset = 5, average filter 15x15, $R_{min}(C_b)$ = 2787.

5. Trajectory planning in dynamic and partially-known domains

The two problems of planning trajectories in unpredictable dynamic environment and in partially-known environments are equivalent. In both cases the robot has to adapt its plans continuously to changes in (its knowledge of) the world. In this section we present a dynamic version of the Fast Marching algorithm called DFM and we compare it to A*, FM, FM* and D* Lite algorithms in simulated 2D environments. The DFM based trajectory planning method is eventually tested in a real open water environment using the AUV prototype of the Ocean Systems Laboratory.

5.1 DFM algorithm

The DFM algorithm is inspired from the LPA* and D* Lite algorithms described in (Koenig et al., 2004). It is similar to the E* algorithm developed by Philippsen in (Philippsen & Siegwart, 2005) but we prefer to name this algorithm DFM instead of E* because the asterisk usually refers to heuristically guided search algorithms (such as A* and D* algorithms). Since no heuristic has been integrated yet in any dynamic version of the Fast Marching algorithm, we propose to use the abbreviation DFM for Dynamic Fast Marching.

According to the principle of optimality it is not necessary to recompute an entire trajectory from A to B when a change appears in C somewhere between A and B. An efficient algorithm may only update the trajectory from C to B and leave the sub-trajectory from A to C unchanged.

5.1.1 Local consistency concept

Since changes appear dynamically in the cost function, any configuration $x \in \Omega$ may be updated more than once. The computation process of the distance function needs to be dynamic and the previous division between Accepted, Current and Unvisited sets of configurations is not compliant any more with a refresh of an Accepted configuration. Recall that an Accepted configuration in the static FM algorithm is frozen. Several updates of the estimate v(x) of the distance function u for a configuration x in Current is possible but the exploration process can only proceed forward from Unvisited to Accepted such as a flame in a landscape. The "engine" of this mechanism is that, at each iteration of the FM algorithm, the configuration x_{top} is moved from Current to Accepted. Then, its neighbours are updated, and the process continues until the goal configuration (initially tagged as Unvisited) is set as Accepted. The "Unvisited-Current-Accepted" scheme is well designed for static problems since a configuration can only proceed one way:

$$\begin{array}{ccccccc} \text{Unvisited} & \rightarrow & \text{Current} & \rightarrow & \text{Accepted} \\ \infty & \rightarrow & v & \rightarrow & u \end{array}$$

In the DFM algorithm, the "tripartite" structure "Unvisited-Current-Accepted" is removed and replaced by a more subtle mechanism between the estimate v and the distance function u. The latter structure is made dynamic by the fact that the relationship between u and v is bilateral. The estimate v, which is affected by changes in the cost function τ, is computed from u, but u itself is computed from v:

$$\tau \rightarrow v \Leftrightarrow u$$

This mechanism, described in detail in the pseudo-code of the next section, stops when v and u match. The "engine" that leads to the "bipartite" agreement between v and u is the processing of a priority queue Q that contains exactly the inconsistent configurations defined as follows (Koenig et al., 2004). A configuration x is called *locally consistent* if $v(x) = u(x)$ and is called *locally inconsistent* if $v(x) \neq u(x)$. In (Philippsen & Siegwart, 2005), the authors reproduce this inequality in their pseudo-code. However, since Fast Marching methods use real numbers for approximating the distance function, a tolerance ε (set empirically at 0.1 in our implementations) must be introduced in the DFM algorithm so that the previous inequality becomes:

$$\left|v(x) - u(x)\right| > \varepsilon \tag{18}$$

5.1.2 Pseudo-code of the 3D DFM algorithm

The pseudo code of the 3D DFM algorithm is given in table 4.

Procedure **CalculateKey(x)**

{01} return [0.5*min(v(x), u(x)) + 0.5*$H_e(x, x_{goal})$; min(v(x), u(x))];

Procedure **Initialize()**

{02} Q = Ø;
{03} for all $x \in \Omega$, $v(x) = u(x) = \infty$;
{04} $v(x_{start}) = 0$
{05} Q.Insert(x_{start}, [0.5*$H_e(x_{start}, x_{goal})$; 0]);

Procedure **FMComputeV(x)**

{06} Select configurations A_1, B_1, C_1 using the computation procedure of table 2;
{07} Apply case 1 or case 2 using the computation procedure 3.4.2.

Procedure **Update(x)**

{08} if $x \neq x_{start}$ then v(x) = FMComputeV(x);
{09} if $x \in Q$ then Q.Remove(x);
{10} if $|v(x) - u(x)| > \varepsilon$ then Q.Insert(x, CalculateKey(x));

Procedure **RunDFM()**

{11} while Q.TopKey() < CalculateKey(x_{goal}) OR $|v(x_{goal}) - u(x_{goal})| > \varepsilon$
{12} x = Q.Pop();
{13} if v(x) < u(x)
{14} u(x) = v(x);
{15} for all $y \in$ Neigh(x) Update(y);
{16} else
{17} $u(x) = \infty$;
{18} for all $y \in$ Neigh(x)\{x} Update(y);

Procedure **Main()**

{19} Initialize();
{20} forever
{21} RunDFM();
{22} Wait for changes in τ;
{23} for all configurations {x} with changed cost
{24} Update τ({x})
{25} Update({x})

Table 4. Pseudo code of the 3D DFM algorithm.

Main functions are:

- Q.Insert(x, key(x)): insert configuration x in the priority queue Q with priority key(x) = CalculateKey(x);
- Q.Remove(x): remove configuration x from the priority queue Q;

- Q.TopKey() = CalculateKey(x_{top})
- Q.Pop(): remove x_{top} from the priority queue Q and return it;

Main procedures are:

- **Initialize()**, lines {02-05}. Estimate v and distance function u are initialized at ∞, except for the start configuration x_{start} for which $v(x_{start}) = 0$. Then, start configuration is inconsistent and it is inserted in the priority queue Q described farther.
- **FMComputeV(x)**, lines {06-07}. The estimate v(x) is computed using the procedure described in table 2 similarly to the static 3D Fast Marching algorithm.
- **Update(x)**, lines {08-10}. First, v(x) is computed using FMComputeV(x). Second, x is removed from Q and, if x is still inconsistent, then it is re-inserted in Q.
- **RunDFM()**, lines {11-18}. The inconsistent configurations in Q are processed until their priorities become higher than the priority of the goal configuration x_{goal} AND x_{goal} becomes consistent (line {11}).

5.2 Application to trajectory planning for AUV in simulated environment

The purpose of this section is to test the DFM algorithm in a realistic simulated environment. First, a dynamic testbed is built, in which obstacles are supposed to be sensed by a sonar device. Second, DFM performance is analyzed and compared to some other dynamic trajectory planning algorithms.

5.2.1 Simulated testbed

We propose to use a simulated 500x500 pixels 2D sonar image as a cost function for testing the DFM algorithm. We want the cost function to be binary. It implies that we need to build a sonar image in which obstacles are supposed to be classified.

Before building the sonar image (SI), a binary map of obstacles (MO) is randomly generated. Three parameters control:

- the number of obstacles to generate ($nb_{Obst} = 50$),
- the number of obstacles to modify ($nb_{ObstMod} = 15$),
- the range of the width w_{Obst} and the length l_{Obst} of obstacles ($10 < w_{Obst}, l_{Obst} < 100$ (in pixels)).

The number of obstacles to be modified refers to the number of obstacles that will be randomly added or deleted from one map to another between the first and the second computation of the DFM algorithm in the tests of the next section.

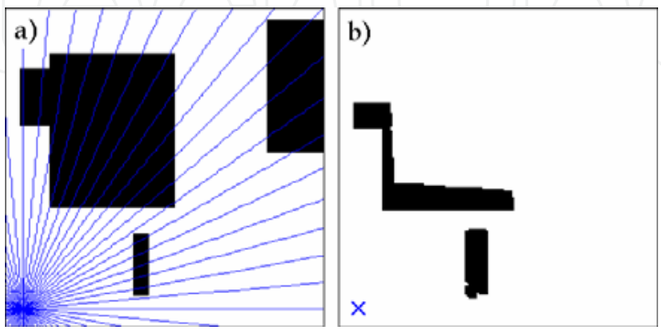

Fig. 5. Close-up on the computation of a simulated sonar image. a) A radial sweep is performed on a binary cost map using a virtual sonar beam (blue lines). b) The sonar echoes are interpreted to build the sonar image.

Once the map of obstacles has been generated, a ray tracing technique is used to build the sonar image. A radial sweep of 360 degrees is performed using a virtual sonar beam with a limited range (sonar-range = 150 pixels). It is assumed that obstacles have the properties of total reflection and homogeneous diffraction at the virtual frequency of the sonar, so that, when the beam meets an obstacle in MO, a spot is generated in SI (see figure 5). The size of the spot, which corresponds to the duration of the sonar pulse, is tuneable (size-spot = 10 pixels)

5.2.2 Dynamic trajectory planning optimization

In practice dynamic replanning algorithms are always launched from the goal to the robot configuration. These are the two reasons. First, it is obvious that dynamic trajectory planning algorithms are more efficient when changes appear close the goal location (Koenig et al., 2004). Since an AUV can only detect the changes that are close to its location (because of the reflection of the sonar beam by the obstacles and because of the limited range of the sonar), it is logic to consider the robot configuration as the goal configuration for the replanning algorithm.

Second, and more importantly, since the robot is continuously moving, its location is necessarily different between two consecutive replanning processes. If the robot configuration was considered as the start configuration of the replanning algorithm, the whole distance function should be permanently recomputed and the computational efficiency of the dynamic planning algorithm would be lost.

5.2.3 Comparative study

In this section a comparative study on a set of deterministic-sampling based dynamic trajectory planners is carried out to analyze the performance of the DFM algorithm. A*, FM, FM*, D* Lite and DFM algorithms are tested using the simulation testbed described in the previous section. The graph of figure 6 depicts the performance of the five trajectory planning algorithms over a range of replanning computations (runs).

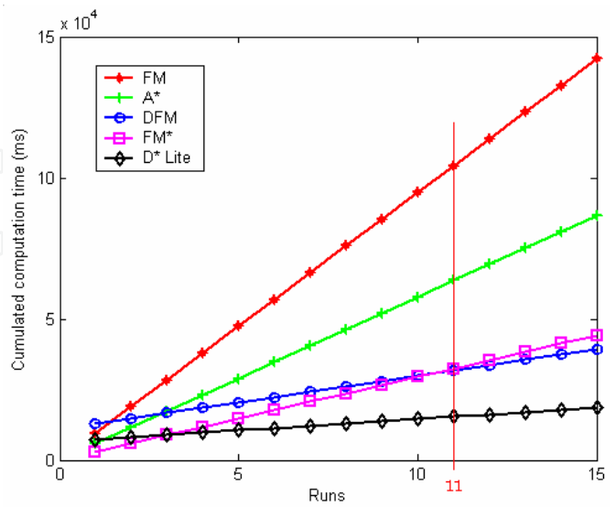

Fig. 6. Performance of A*, FM, FM*, D* Lite and DFM algorithms as a function of the number of runs. Each graph represents the evolution of the cumulated computation time of each algorithm over the runs.

One can see in figure 6 that FM* is the fastest static algorithm. However, from run 11, dynamic replanning algorithms (D* Lite and DFM) give better performance than static planning algorithms (A*, FM and FM*). This is explained by the greater efficiency of dynamic replanning algorithms when changes in the cost function happen close to the goal configuration.

5.2.4 Conclusion and future work

First, the FM* algorithm appears to be the best static trajectory planning algorithm (better than A*) both in terms of computation time and smoothness of the trajectory solutions. Second, dynamic trajectory planning algorithms are faster than static planners after a limited amount of time. Third, the DFM algorithm is slower than the D* Lite algorithm but it allows the curvature of the trajectories to be controllable.

Contrary to D* Lite, DFM algorithm does not include any heuristic to speed up the exploration process. One interesting direction for further research would be to develop what could be called the DFM* algorithm. A novel DFM* algorithm would combine the accuracy of the DFM algorithm with the exploration efficiency of the FM* algorithm.

5.3 Application to trajectory planning for AUV in real environment

In this section a complete AUV architecture designed to operate in unstructured environments is evaluated. Open water missions have been carried out to establish the performance of our FM based trajectory planning approach using the AUV prototype of the Ocean Systems Laboratory (Evans et al., 2008).

5.3.1 AUV architecture

It is important for an AUV to be able to follow complex scenarios and to rapidly respond to emergency situations. The following architecture has been designed to reach these objectives.

In this section both sensor and deliberative layers are described. Literature on actuators and control systems for AUVs may be found in (Hamilton et al., 2007; Fossen, 2002). The trajectory generation is provided by the DFM algorithm.

5.3.1.1 Sensor layer

The primary objective of the sensor layer is the generation of a local map. The output of this map provides an input for the deliberative layer that tries to match the arrangements of targets within the map against known scenarios. Our vehicle was equipped with inexpensive Tritech Sea King mechanically scanning forward looking sonar for obstacle detection. Navigation used an integrated GPS and Doppler Velocity Log solution mixing absolute and dead reckoning modes.

5.3.1.2 Deliberative layer

To provide deliberation in the generation of a safe behavior, a subsumption (Brooks, 1986) deliberative architecture has been chosen. It includes a reactive layer above a scenario layer as depicted in figure 7.

The reactive layer is empowered to take over from the scenario layer in the event of emergency, thus safeguarding the vehicle. It is deigned as a fuzzy system and it is triggered by range to nearest object.

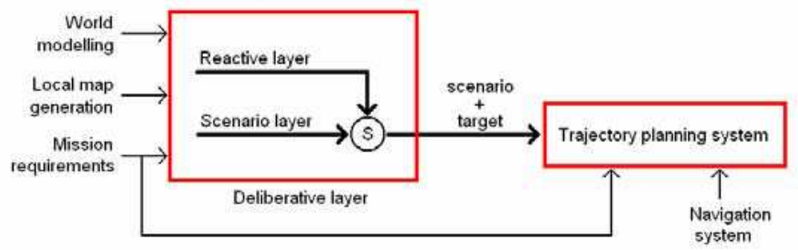

Fig. 7. Subsumption architecture implemented in the deliberative layer. A reactive layer is empowered to take over from the scenario layer in the event of emergency.

In the scenario layer, scripts called *scenarios* are employed. They are selected based on external and internal information along with mission requirements coming from the sensor layer. Ultimately, the deliberative layer sends the selected scenario and the selected target to the trajectory planning system, which generates the waypoints that are applied to the vehicle autopilot.

5.3.1.3 Trajectory generation

The trajectory planning method used in this module is based on the DFM algorithm. Since the local map around the vehicle is regularly updated, the DFM algorithm fulfils its real-time trajectory replanning mission.

5.3.2 Open water trials

A comprehensive set of open water trials have been carried out to validate some of the science reported above. Trials were carried out in Portmore Loch (Scotland) and Vobster Quarry Somerset (England) using HWU RAUVER hover capable AUV, see figure 8.a.

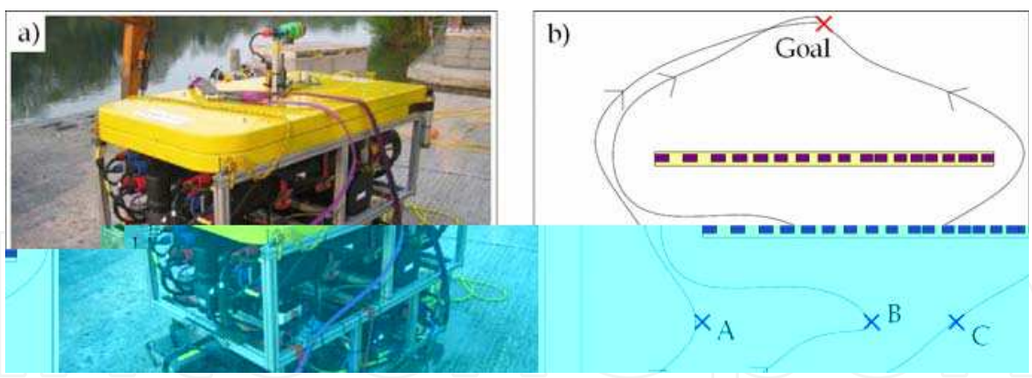

Fig. 8. The prototype of the Ocean Systems Laboratory, RAUVER, a hover capable autonomous underwater vehicle. b) In-water trials: net avoidance from different starting positions.

In this test, a net structure is set in the middle of the scene, see figure 8.b. The vehicle is sent to a waypoint located in the other side of the net. The starting point of the mission is situated in different places to observe the different behaviours. The expected behaviour is for the deliberative layer to plan a parallel course until the extent is detected, then a horizontal diversion. If this fails, the reactive layer should reverse to clear danger.
During the test starting from A the deliberative system found an alternative trajectory to reach the target avoiding the obstacle with a left horizontal diversion. In test B, the system,

that was keeping track of the next extension, found a trajectory on the left of the net and replanned the mission to go back and do the left horizontal diversion. Test C demonstrated that, by moving the starting point a little forward to the right of the net, extent is detected and the system is able to find a horizontal diversion on the right side of the net. In all cases, deliberative behaviours were always successful, without recourse to the reactive layer.

5.3.3 Conclusion

A complete architecture has been designed, developed and tested for real AUV missions. By inserting a scenario layer in the deliberative module, local maps generated in the sensor layer are used to choose and parameterize appropriate behaviours on the fly. A reactive layer has also been implemented and contributes by inhibiting goal points from the scenario layer in extremis. It has fuzzy behaviours to rapidly extricate the vehicle in case of emergency conditions. In practice, the scenario layer is rarely if ever inhibited by this reactive layer because the DFM based trajectory planning module produces safe trajectories.
Tests have been carried out on the real AUV prototype RAUVER of the Ocean Systems Laboratory in open water. Whilst the experiments reached the objectives, an unforeseen problem had to be solved: bottom reverberation. Bottom reverberation appeared as obstacle data and affected computed trajectories particularly at long ranges. In practice bottom reverberation has been detected and segmented into the local map, which successfully prevented these artefacts.

6. Conclusion

6.1. Recapitulative

The underwater world is a very demanding environment for trajectory planning algorithms. Great efforts are currently being made to develop autonomous systems as underwater technology becomes more mature. Several key issues for the three dimensional underwater trajectory planning problem have been addressed in this chapter. Reliability of trajectory planners has been improved by introducing the Fast Marching algorithm as a new basis for sampling based trajectory planning methods in the continuous domain.
First, we have introduced the trajectory planning framework and the basic concepts shared by all the deterministic sampling based planning algorithms. The Fast Marching method, as one of these trajectory planning technique is similar in spirit to classical grid-search algorithms such as the A* algorithm. This led us to develop a new algorithm, called FM*, that combines the exploration efficiency of the A* algorithm with the accuracy of the Fast Marching method. For these reasons, the FM* algorithm opens new possibilities for planning trajectories in wide and continuous underwater environments.
Second, even if they are implemented on a discretized perception of the world, Fast Marching based planning methods have the property to extract derivable trajectories. By applying mathematical tools from differential geometry, it has been proved that smoothing input data results in smoother trajectories. A technique has been proposed that insures the feasibility of a trajectory for a mobile robot with a given turning radius. This technique iteratively smoothes input data until a formal criterion is satisfied. The method is efficient because the Fast Marching algorithm is eventually launched only when input data are compliant with the curvature constraints of the vehicle.
Third, another approach has been developed to speed up the exploration process in the case of partially-known or dynamic environments. A dynamic version of the Fast Marching

algorithm, called DFM, has been presented that is able to reuse information of previous searches. Compared to A*, FM, FM* and D* Lite algorithms, the DFM algorithm is very efficient when changes happen randomly in the vehicle's perception of the world.
Eventually, a complete architecture has been designed, developed and tested for real AUV missions. Performance and usefulness of the DFM based trajectory planning approach in partially-known domains have been demonstrated using the experimental prototype of the Ocean Systems Laboratory.

6.2 Future work

6.2.1 High dimensional state spaces

Even if the O(Nlog N) complexity of the Fast Marching algorithm is similar to the complexity of classical discrete grid-search algorithms, FM based trajectory planners are suitable for C-spaces with only a few numbers of dimensions (at most three in practice). The DFM algorithm improves re-planning efficiency of trajectory planners in unpredictable or a priori unknown environments. Nonetheless, further research would benefit from the addition of a heuristic to the DFM algorithm in order to speed up its exploration capacities.

6.2.2 Planning with uncertainty

In this chapter we have not dealt with uncertainties on the AUV perception of the world. It has been assumed that the vehicle had either an a priori comprehensive knowledge of its environment (chapters 2, 3 and 4) or had a limited visibility (chapter 5). In both cases, precise location of C-obstacles was assumed. This hypothesis is not very realistic as underwater sensors have limited performance. In (Petres, 2007) a dilation of C-obstacles is proposed to improve the safety of the trajectories. This simple method is easy to implement practically but it does not explicitly include uncertainties about sensor specifications. Further work on FM based trajectory planning for AUV using an information space representation would be promising.
On the other hand, uncertainties about AUV configurations have not been considered in the presented trajectory planning methods. This is not very realistic as accurate underwater navigation is still a challenging issue. Nonetheless, concurrent mapping and localization (CML) techniques exist for AUV navigation (Tena Ruiz et al., 2004). This topic is out of the scope of this chapter but further development would benefit from a joint navigation approach for AUV that would couple CML and advanced trajectory planning techniques.

7. References

Brooks, R.A. (1986). A Robust Layered Control System for a Mobile Robot, *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, pp. 14-23, ISSN: 0882-4967

Bruckstein, A.M. (1988). On shape from shading, *Computer Vision, Graphics, and Image Processing*, Vol. 44, No. 2, pp. 139-154, ISSN: 0734-189X

Caselles, V.; Kimmel, R. & Sapiro, G. (1997). Geodesic Active Contours, *International Journal of Computer Vision*, Vol. 22, No. 1, pp. 61-79, ISSN: 0920-5691 (Print), 1573-1405 (Online)

Cohen, L.D. & Kimmel, R. (1997). Global Minimum for Active Contour Models: A Minimal Path Approach, *International Journal of Computer Vision*, Vol. 24, No. 1, pp. 57-78, ISSN: 0920-5691 (Print), 1573-1405 (Online)

Deschamps, T. & Cohen, L.D. (2001). Fast Extraction of Minimal Paths in 3D Images and Applications to Virtual Endoscopy, *Medical Image Analysis*, Vol. 5, No. 4, pp. 281-299

Evans, J.; Patron, P.; Smith, B. & Lane, D.M. (2008). Design and Evaluation of a Reactive and Deliberative Collision Avoidance and Escape Architecture for Autonomous Robots, *Autonomous Robots*, Vol. 24, No. 3, pp. 247-266, ISSN: 0929-5593

Fossen, T.I. (2002). *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs, and Underwater Vehicles*, Marine Cybernetics, ISBN: 82-92356-00-2

Godunov, S.K. (1969). A Difference Scheme for Numerical Solution of Discontinuous Solution of Hydrodynamic Equations, *Sbornik Mathematics*, Vol. 47, pp. 271-306

Hamilton, K.; Lane, D.M.; Brown, K.E.; Evans, J. & Taylor, N.K. (2007). An Integrated Diagnostic Architecture for Autonomous Underwater Vehicles: Research Articles, *Journal of Field Robotics*, Vol. 24, No. 6, pp. 497-526, ISSN: 1556-4959

Hart, P.E.; Nilsson, N.J.; Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100-107, ISSN: 0536-1567

Kimmel, R.; Kiryati, N. & Bruckstein, A.M. (1998). Multi-Valued Distance Maps for Motion Planning on Surfaces with Moving Obstacles, *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 3, pp. 427-436, ISSN: 1042-296X

Koenig, S.; Likhachev, M.; Liu, Y. & Furcy, D. (2004). Incremental Heuristic Search in Artificial Intelligence, *Artificial Intelligence Magazine*, vol. 25, No. 2, pp. 99-112, ISSN: 0738-4602

Latombe, J.-C. (1991). *Robot Motion Planning*, Kluwer Academic Publisher, ISBN: 079239206X, Norwell, MA, USA

LaValle, S.M. (2006), *Planning Algorithms*, Cambridge University Press, ISBN-10: 0521862051, ISBN-13: 978-0521862059

Melchior, P.; Orsoni, B.; Lavialle, O.; Poty, A. & Oustaloup, A. (2003). Consideration of Obstacle Danger Level in Path Planning Using A* and Fast-Marching Optimization: Comparative Study, *Signal Processing*, Vol. 83, No. 11, ISSN: 0165-1684

Moravec, H. (2003). Robots, After All, *Communications of the ACM*, Vol. 46, No. 10, pp. 90-97, ISSN: 0001-0782

Petres, C.; Pailhas, Y.; Patron, P.; Petillot, Y.; Evans, J. & Lane, D.M. (2007). Path Planning for Autonomous Underwater Vehicles, *IEEE Transactions on Robotics*, Vol. 23, No. 2, pp. 331-341, ISSN: 1552-3098

Petres, C. (2007). *Trajectory Planning for Autonomous Underwater Vehicles*, Heriot-Watt University, Ph.D. Thesis

Philippsen, R. & Siegwart, R. (2005). An Interpolated Dynamic Navigation Function, *Proceedings of IEEE Conference on Robotics and Automation (ICRA 2005)*, pp. 3782-3789, ISBN: 0-7803-8914-X

Rouy, E. & Tourin, A. (1992). A Viscosity Solutions Approach to Shape-from-Shading, *SIAM Journal on Numerical Analyzis*, Vol. 29, No. 3, pp. 867-884, ISSN: 0036-1429

Sethian, J.A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, ISBN: 0521645573, 9780521645577, Cambridge, MA, USA

Tena Ruiz, I.; de Raucourt, S.; Petillot, Y. & Lane, D.M. (2004). Concurrent Mapping and Localization Using Sidescan Sonar, *IEEE Journal of Oceanic Engineering*, Vol. 29, No. 2, pp. 442-456, ISSN: 0364-9059

Tsitsiklis, J.N. (1995). Efficient Algorithms for Globally Optimal Trajectories, *IEEE Transactions on Automatic Control*, Vol. 40, No. 9, pp. 1528-1538, ISSN: 0018-9286

Underwater Vehicles

Edited by Alexander V. Inzartsev

ISBN 978-953-7619-49-7
Hard cover, 582 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009

For the latest twenty to thirty years, a significant number of AUVs has been created for the solving of wide spectrum of scientific and applied tasks of ocean development and research. For the short time period the AUVs have shown the efficiency at performance of complex search and inspection works and opened a number of new important applications. Initially the information about AUVs had mainly review-advertising character but now more attention is paid to practical achievements, problems and systems technologies. AUVs are losing their prototype status and have become a fully operational, reliable and effective tool and modern multi-purpose AUVs represent the new class of underwater robotic objects with inherent tasks and practical applications, particular features of technology, systems structure and functional properties.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Clement Petres, Yan Pailhas, Pedro Patron, Jonathan Evans, Yvan Petillot and Dave Lane (2009). Trajectory Planning for Autonomous Underwater Vehicles, Underwater Vehicles, Alexander V. Inzartsev (Ed.), ISBN: 978-953-7619-49-7, InTech, Available from: http://www.intechopen.com/books/underwater_vehicles/trajectory_planning_for_autonomous_underwater_vehicles

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen