Optimized path planning for marine vehicles considering uncertainty

Margarida Correia
Universidade do Porto
FEUP/DEEC
Email: margarida.mrrc@gmail.com

Anbal Matos
INESC-TEC
Universidade do Porto
FEUP/DEEC
Rua Dr. Roberto Frias, n. 378,
4200-465 Porto, Portugal
Email: anibal@fe.up.pt

***Abstract*—The majority of Autonomous Underwater Vehicles (AUVs) spend most of their energy in order to propel themselves. Therefore, a good path planning technique can improve both their autonomy and range, thus their performance. This paper proposes an optimized trajectory planning methodology able to find the best possible path from a starting point to a target position, taking advantage of the water currents. In addition, the possibility of water currents changing throughout the path is contemplated and both the optimal path and currents field are updated based on the detected deviations in a predefined number of checkpoints along the path. Finally, an estimate of the vehicle's real path is performed.**

I. Introduction

Autonomous Underwater Vehicles (AUVs) are self-propelled vehicles which movement is highly dependent on its surrounding environment. The ocean presents a highly dynamic behavior, with energetic currents, wind field effects, flows induced by tides, etc.. The high variability of this environment can jeopardize an entire operation. Hence predicting and updating the currents' map, the AUV's perception of its environment, is fundamental in order to maximize the covered area in a mission, save energy and minimize the travel time, thus increase autonomy and range [1].

Several authors present works in this field, trending towards the incorporation of the currents in the robots motion model [2] [1], increasingly considering the robots environment dynamics [3] and finally minimizing both time and energy spent in a mission [4] [5].

Therefore, in this work an optimized path planning technique was developed in order to find the best possible path from a starting point to a target position, in the minimum possible time and thus energy consumption, by using the water currents knowledge and update.

Optimal control is a wide field of study with a diverse range of applications, such as robotics, aeronautics, etc. According to [6], in an optimization problem one should consider three subproblems: Formulation (translation of the problem at stake into a mathematical optimization problem), Initialization (choice of the optimization algorithm to use and definition of the initial values) and Optimization procedure (implementation and choice of technology).

Following that methodology in section II the engineering problem is mathematically formulated and defined. Afterwards, in sections III and IV, the choice of the algorithm and its implementation are presented.

In section V a proposal of how one could use the previously computed uncertainty to update the currents' map at each check point throughout the path is performed and afterwards, that uncertainty is used to both compute the new path and update the currents' field.

At last the conclusions and future work are presented in chapter VI.

II. Problem definition

This paper is intended to address the problem of determining the best possible path from a start position to a target, in a marine environment. Therefore an optimization algorithm should be used to find the best path. But what does it mean to be the best path? What should one optimize? In the presence of water currents a vehicle should be able to profit from it and avoid counter currents. Also it should take the least possible time to go from point A to B. This time depends on the path distance, the vehicle's speed and the currents' speed, therefore, time will be the objective function, thus what we want to minimize.

In mathematics an optimization problem consists in finding the best solution from a set of possible solutions. Usually an objective function is minimized or maximized within its domain.

In the described problem we want to minimize the time and for that we have some linear constraints as well as nonlinear restrictions. Hence one is before a nonlinear optimization problem with constraints.

According to [7], a nonlinear optimization problem in the standard form is mathematically defined by:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subjet to} \quad & h_i(x) = 0, \quad \text{for } i = 1, \ldots, l. \\ & g_i(x) \leq 0, \quad \text{for } i = 1, \ldots, m. \\ & x \in X \end{aligned} \tag{1}$$

where

f: objective function

978-0-933957-40-4 ©2013 MTS

x: parameter vector

$h_i(x) = 0$: equality constraints

$g_i(x) \leq 0$: inequality constraints

The above problem must be solved for the values of the variables of the vector x and satisfy the restrictions, both equalities and inequalities, as well as minimize the objective function f.

Keeping that in mind, the problem under study can be mathematically characterized as follows:

$$\min_{t} \quad T \tag{2}$$

$$\text{s.t.} \quad \dot{x}(t) = v_x(t) + v_{field_x}(t) \tag{3}$$

$$\dot{y}(t) = v_y(t) + v_{field_y}(t) \tag{4}$$

$$v_x^2 + v_y^2 \leq V_{max}^2 \tag{5}$$

$$(x, y)(0) = P_{X_0} \tag{6}$$

$$(x, y)(t_f) = P_{X_f} \tag{7}$$

where

$T = f$: objective function

$X = (x, y)$: parameter vector

$v_x(t)$: x component of the vehicle's speed

$v_y(t)$: y component of the vehicle's speed

$v_{field_x}(t)$: x component of the water currents' speed

$v_{field_y}(t)$: y component of the water currents' speed

$\dot{x}(t)$: x component of the total speed over time

$\dot{y}(t)$: y component of the total speed over time

V_{max}: maximum vehicle's speed

P_{X_0}: initial point of the path

P_{X_f}: final point of the path

The differential equations 3 and 4 represent the equality constraints whilst equation 5, which limits the vehicle's speed, is the inequality constraint of this problem. Furthermore, equations 6 and 7 represent the lower and upper bounds of vector X.

The problem is now completely defined and in the next section an appropriate solver will be selected as well as the optimization method within the nonlinear optimal control scope.

III. Choice of Optimization Algorithm and Toolbox

Optimal control is a wide field of study with a diverse range of applications such as robotics, aeronautics, economics and so on. But as all fields it has its pros and cons. Although it has a systematic design procedure, it is applicable to nonlinear control problems and captures limitations (constraints), it can be hard to find suitable criteria and to solve the equations that give the optimal controller [8]. Therefore, for this work it was decided to find an optimization solver which would improve the chances of having robust results and save time.

There are several solvers and the following were considered to be used:

1) BOCOP (open source toolbox for optimal control problems) [9]
2) RIOTS(Matlab toolbox for solving optimal control problems) [10]
3) OCP (optimal control problem solver) [11]
4) PSOPT (open source optimal control software package) [12]
5) MATLAB Optimization Toolbox [13].

Some examples of the above solvers/toolboxes were studied. But a demonstration from the Optimization Toolbox of MATLAB stood out. In [14], a solution to a path planning problem for an airplane navigating through a vector field of wind in the least possible time was presented. If we change wind for water current and the airplane for a marine vehicle the problem looks really similar to the one under study. Therefore, it was decided the code available in MATLAB Central would be adapted.

A. Base script: Math Works' "Finding Optimal Path Using Optimization Toolbox"

As the title states, MATLAB's script has the aim of finding the optimal path in a navigation problem through a wind field in the least possible time, using the Optimization Toolbox. Hence time is the objective function and is defined as follows:

$$Time = \frac{Distance\ \ Traveled}{Average\ \ Speed} \tag{8}$$

The vehicle's environment is represented by a vector field: the local wind conditions at every point are characterized by their magnitude and direction. The plane should profit from tailwinds (wind in the direction of travel) while avoiding headwinds (wind in the opposite direction of travel). Crosswinds are assumed to have no effect in the movement.

In figure 1 one can see how the wind vector field is displayed in MATLAB. The arrows (given by the *quiver* command) represent at each point the wind's magnitude and direction.

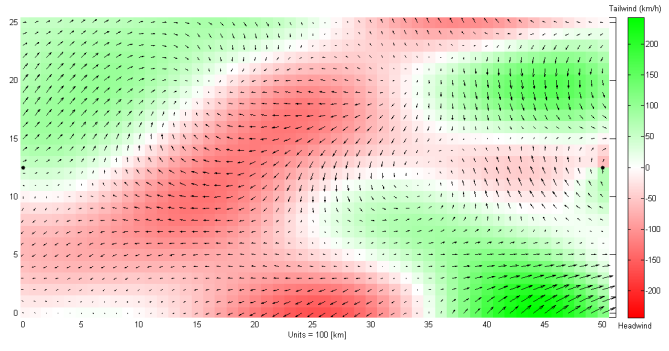

Fig. 1. Environment representation: wind vector field.

The red zones represent the places to avoid, since the vector fields has an opposite direction to the intended motion whilst the green areas represent the spots where the wind is favorable to the vehicle's movement.

Defining equally horizontally distributed points one gets a straight line from the starting point to the target. Although this line, represented in figure 2, would give the shortest path, it might not give the quickest one because of the wind field disposition.

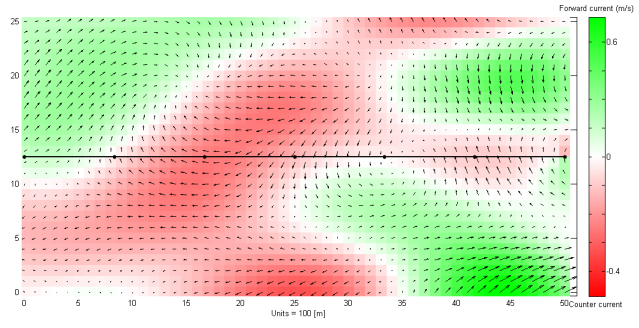

Fig. 2. Straight line path.

The next step will then be to define the best possible path that takes the vehicle from the starting point to the destination. And for that, as it was done for the straight line path, one should set a number of way points which will define this trajectory.

Having these points, the computation of travel time over the path defined by them is given by a line integral: the path from one point to another is broken in smaller subintervals in which one consider the wind field to be approximately constant. Taking that into account, the average velocity over the segment is found as follows:

$$v_{i_x} = v_x + v_{field_{i_x}} \tag{9}$$

$$v_{i_y} = v_y + v_{field_{i_y}} \tag{10}$$

Using the expression 8 and summing up all the small subintervals and then all the segments' time the total travel time over the entire path is found.

All of the above proceedings are performed in the file *getTimeFromPath.m* and this function is used as the objective function of the algorithm.

In the original script, the vehicle's speed is considered to be constant throughout the path and equal to 500 km/h. The initial point is $P_{X_0} = (0, 1250)$ and the target $P_{X_f} = (5000, 1250)$. Hence, the travel distance is 5000 km.

If the vector field was not considered, a straight line would be both the shortest and the quickest path and it would take the vehicle 10 h to complete it. Since in the example a vector field with headwinds and tailwinds is considered, the result for a straight line will be different. By using the expressions 10 and 10 one gets a total travel time of 10 h 58.8 min.

To determine the best path which minimizes the time spent in the journey MATLAB's script uses the function *fmincon* [15], which is available in MATLAB's Optimization Toolbox and finds the minimum of a constrained nonlinear multivariable problem.

This function allows the user to choose between four nonlinear programming methods to solve the optimization problem:

- ***Trust Region Reflective*** [16]: also known as restricted step methods, the trust region algorithms take the following approximation of the main minimization problem:

$$\min_s \{ \; q(s) \; \text{ such that } \; s \in N \; \} \tag{11}$$

for a certain x. $q(s)$ is a quadratic approximation of the objective function $f(x)$ in the neighborhood N of x. This neighborhood is called trust region and represents a subset of the objective function space in which one believes the minimum lies on.
Solving the subproblem one gets the value of s, minimum value of $q(s)$, which will be called step. Afterwards, if $f(x+s) < f(x)$ the current point x is updated to $x+s$ and N, the trusted region is expanded. Otherwise the current point remains the same, N is contracted and one should recompute the step s. These steps are repeated until the method converges.[1]
This is the method used by default in MATLAB's *fmincon* function.

- ***Active Set*** [17] [16]: also known as projection method, it is most effective with small to medium-scale problems and falls within the scope of quadratic programming (optimization problem with a quadratic objective function and linear constraints).
In an optimization problem, a feasible region is the set of all points x where the optimal solution might be. These points are defined by the problem's constraints 1 (equalities and inequalities).
Given an x point in the feasible region, a constraint $g_i(x) \geq 0$ is considered to be active if $g_i(x) = 0$ (all equality constraints are active) and inactive if otherwise. Hence the active set at x is the group of all the active optimization problem's constraints.
The main steps of an Active-Set method are then:

1: Find a feasible starting point x
2: **while** not "optimal enough" **do**
3: Solve $g_i(x) = 0$
4: Compute λ_i of the active set
5: Remove a subset with $\lambda_i < 0$
6: Search for infeasible constraints
7: **end while**

This is the chosen algorithm to solve the problem under study by MATLAB's script.

- ***Interior Point*** [17] [16]: also known as barrier methods, MATLAB's implementation of this method is a variant of Mehrotra's predictor-corrector algorithm, a primal-dual interior-point method [16]. This method uses a barrier function which encodes the convex set. It reaches an optimal solution after crossing the feasible region.

This method can be less accurate than others since the internally computed barrier function keeps inequality constraints away. Therefore, it was not chosen to solve the problem under study.

- ***SQP* (Sequential Quadratic Programming)** [17] [16]: it is an iterative method to solve nonlinear optimization problems with twice continuously differentiable objective functions. These algorithms optimize a quadratic model of the objective function subject to a linearization of the constraints. Using the general definition of the nonlinear programming problem 1, one define the problem's Lagrangian as:

$$\mathcal{L}(x, \lambda, \sigma) = f(x) - \lambda^T g(x) - \sigma^T h(x) \tag{12}$$

where λ and σ are the Lagrange Multipliers.
At each iteration x_k, will try to solve the quadratic programming problem in the direction d_k:

$$\min_d \quad f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k, \sigma_k) d \tag{13}$$

$$\text{s.t.} \quad g(x_k) = \nabla g(x_k)^T d \geq 0 \tag{14}$$

$$h(x_k) = \nabla h(x_k)^T d = 0 \tag{15}$$

The base script uses the algorithm option Active Set, with a maximum iterations number of 2000. The *fmincon* function returns the coordinates of the waypoints which will give the optimal path after being interpolated.

The result of MATLAB's optimization script for five waypoints can be seen in figure 3. The time was improved from the 10 h 58.8 min of the straight line path to 10 h 17.8 min, thus almost one hour.

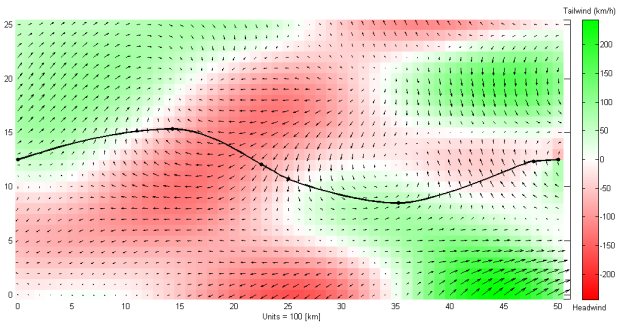

Fig. 3. Original algorithm's results.

IV. Incorporating Uncertainty into the Original Script

In section III, the original script was described: a path planning algorithm determines the best path from a starting point to the destination for an airplane.

Since this work falls within the marine robotics scope, the original algorithm was adapted to fit its aim. Therefore one shall consider the wind field as a water current and instead of an airplane the vehicle under study will be an AUV.

Also, the magnitude of the problem's variables has to be modified. The available OceanSys' AUVs have a range of approximately 40 km with an average speed of 1 m/s and the Glider has a range of 1500 km with an average speed of 0.4 m/s. Therefore, for testing purposes, the total travel distance will be 5 km instead of the original 5000 km, the AUV average speed will be 1 m/s instead of the airplane's 500 km/h and the water currents will have a maximum value of 0.5 m/s, instead of 200 km/h.

But what else should be adapted and improved in MATLAB's implementation? Multiple hypothesis were considered:

- Enable the script responsible for the water currents' definition to be customizable, thus allowing it to be altered to the specific conditions of a mission day. For example, make it able to read a text file with some parameters and convert them into a vectorial field.
- Change the optimization procedure and method (the script uses *fmincon* with the option "active-set" but there are more optimization functions or within *fmincon* other methods).
- The current model optimizes the trajectory without considering any uncertainty. So, it would be a good improvement to incorporate it and recompute the path affected by it.
- Related to the last hypothesis, update the currents' map accordingly to the computed uncertainty.
- Change the script in order to solve a 3D problem, since the water currents are depth dependent.

From the improvements presented above, the priority was to incorporate the uncertainty in the path planning problem. A strong reason for that choice was the fact that the initially studied problem, from MATLAB's example, takes place along hundreds of kilometers. With such distances there is no guarantee the initial conditions will be the same throughout the whole path. Besides the vehicles odometry's errors, the estimated current might not be equal to the real one or change during the traveling, etc.

For those reasons the waypoints, used in the original implementation as the optimization points, which represent the spots where the optimal path should go through, would now have an additional feature. These points would represent check points, where the AUV's estimated optimal position would be compared with its real one, given by, for instance, a GPS signal or the distance to a known beacon. This difference would represent the uncertainty up to that instant.

In order to simulate the uncertainty in the script a circle with unitary radius was drawn around each waypoint and a random point chosen inside that area. In figure 4, the blue circle represents the uncertainty area delimiter while the red point is the aforementioned random point.

The next step is to compute the new optimal path from the new starting point (the red point in the simulation), since by being in a different region of the water current the optimal path might be different from the previously computed.

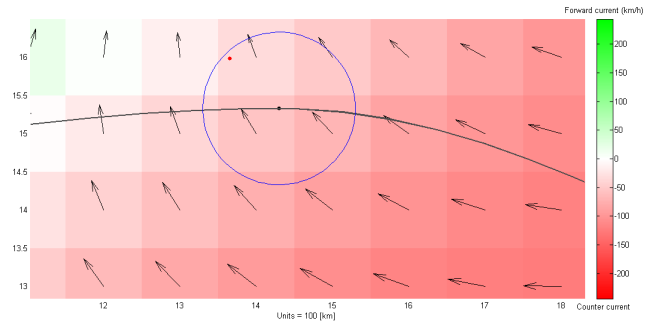

Fig. 4. Simulation of the uncertainty.

In figure 5, the new path, resulting from the new second optimization, is represented in gray, with four waypoints, less one than in the first step.

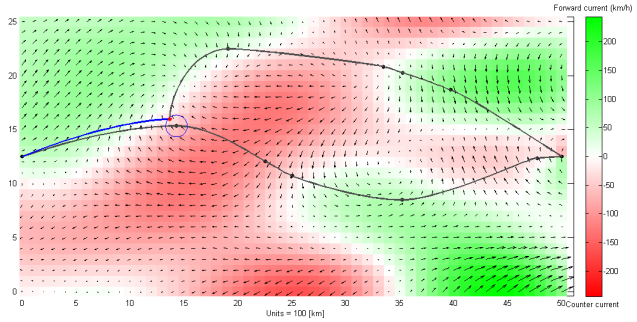

Fig. 5. Representation of the recomputing of the optimal path.

Also, an estimate of what should have been the trajectory described by the vehicle is computed and represented in figure 5 in blue.

In order to obtain the new path points (the blue trajectory), the initial optimal path was divided in small subintervals, one for each original path point. Then, to the vector d_i from the starting waypoint P_0 to the original path point P_i, a fraction of the final displacement d_f, the green vector dPP_i, is added, as shown in figure 6:

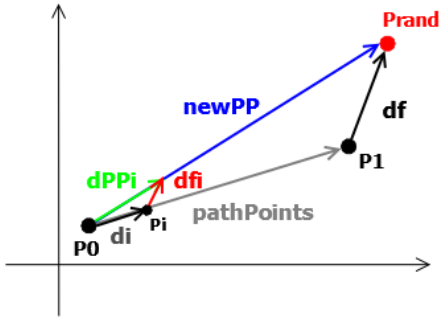

Fig. 6. New path points computation diagram.

The original path points ($pathPoints$ in the diagram) are represented in gray while the estimated new path points are in blue ($newPP$ in the diagram), as in the previously presented simulation plot.

P_0 and P_1, the big black dots, are waypoints while P_{rand}, in red, represents the random sample of the uncertainty area. Therefore d_f is the error between where the vehicle thought it would be at and the actually reached position, as stated in equation 16:

$$d_f = \sqrt{(x_{random} - x_{optimal})^2 + (y_{random} - y_{optimal})^2} \tag{16}$$

where $P_{rand} = (x_{random},\ y_{random})$ and $P_1 = (x_{optimal},\ y_{optimal})$.

An estimate of the real traversed path can then be given by the sum of the $pathPoints$ vector with the displacement d_f:

$$newPP = pathPoints + d_f \tag{17}$$

Furthermore, by splitting the $pathPoints$ vector into smaller vectors d_i, as explained before, the $newPP$ vector can be obtained by summing up all the dPP_i:

$$newPP = \sum dPP_i = \sum d_i + d_{f_i} \tag{18}$$

where $df_i = \frac{d_i}{d_f}$ represent the fraction of the final deviation from P_1 to P_{rand}.

By repeating the aforementioned steps until there are no more waypoints left, one reaches the destination and gets an estimate of the traversed path, as shown in figure 7:

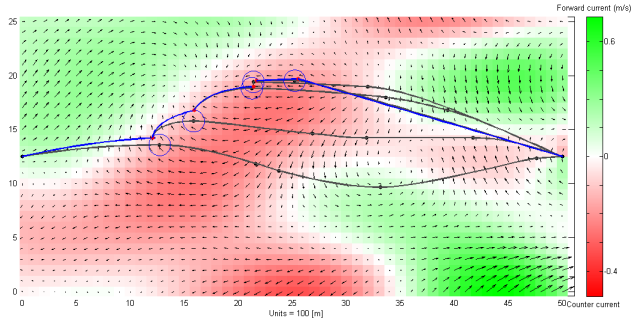

Fig. 7. Results with 5 waypoints with the original and the altered algorithms.

Finally, gathering all the data computed above one should be able to determine the total travel time. For that, two different methods were implemented: the differential time and the direct computation.

In the original script, the travel time was computed from the starting point to the target as explained in section III. Therefore, the first way of computing the total travel time consists of using the original *getTimeFromPath* function and simply alter the initial point to the waypoint one is in. After that, to know how long it took to go from waypoint $i-1$ to waypoint i, the difference between the travel time from $i-1$ to the target and the travel time from i to the target is taken, hence the name differential time.

In equation 19 represents the computation of the total travel time T_{dif}:

$$T_{dif} = \sum_{i=1}^{numWaypoints+1} t_{i-1} - t_i \tag{19}$$

where

numWaypoints+1: total number of checkpoints plus the destination point

t_{i-1}: travel time from check point $i-1$ to the destination

t_i: travel time from check point i to the destination

The direct computation consists of calculating directly the time from each starting point to the next check point and summing up this subinterval's times, as stated in equation 20:

$$T = \sum_{i=1}^{numWaypoints+1} t_i \tag{20}$$

where

T: total travel time obtained by the direct computation method

t_i: time to go from check point $i-1$ to check point i

A. Results and discussion

In this section the tests performed to confirm the algorithm's implementation are described and the results presented and discussed.

The vehicle speed was set to 1 m/s and the water currents can take values from -0.5 m/s up to 0.5 m/s (the signal represents its direction). The total travel distance is 5 km.

After setting these parameters the first test performed consisted on applying a constant vector field in the direction of the intended movement to verify its behavior. It was expected to have a straight line from start to finish, with some small deviations due to the uncertainty simulation.

In figure 8, a strong field was generated to represent an intense forward current and, as expected, in every check point the recomputed path converged towards the destination point.

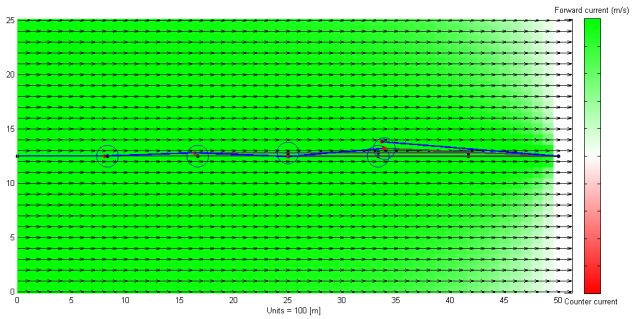

Fig. 8. Application of a constant vector field in the direct of the movement.

Having set the current to 1 m/s, the expected value for a straight line path would be:

$$T = \frac{d}{v_{auv} + v_{field}} \Rightarrow T = \frac{5000}{1+1} = 2500 \text{ s} \tag{21}$$

This value is equivalent to 41 min 40 s, similar to the obtained 41 min 55 sec, from the differential time computation and 41 min 57 sec, from the directly computed time.

After having this confirmation, a random path was defined and applied to the problem with different numbers of checkpoints. Although a random vector field was generated, the same one was used to test the algorithm with different numbers of checkpoints so that one could assess their influence in the travel time.

First, a straight line was generated with the initially set conditions IV-A, as one can see in figure 9. A straight line corresponds to the shortest path from start to end. So, in order to compare the time computation resulting of the optimization, the first step was to obtain the straight line travel duration, which was 1 hour 26 min 58 sec.

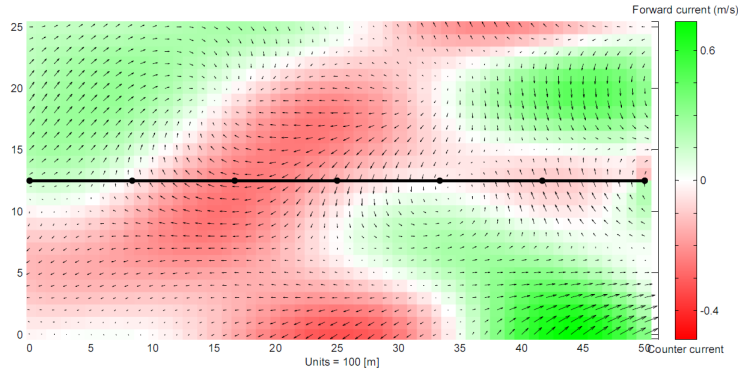

Fig. 9. Straight line: the shortest path one can take from starting point to the destination.

Afterwards, the algorithm was tested with 5, 10, 15 and 20 checkpoints and the respective times obtained. The results of these simulations can be seen in figures 7, 10, 11 and 12, respectively.

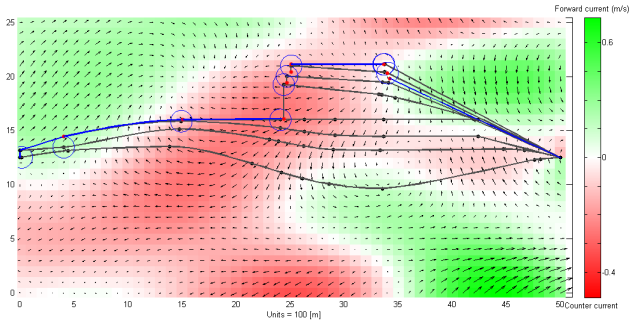

Fig. 10. Results with 10 waypoints with the original and the altered algorithms.

In table I, one can see the resulting travel times for the aforementioned tests.

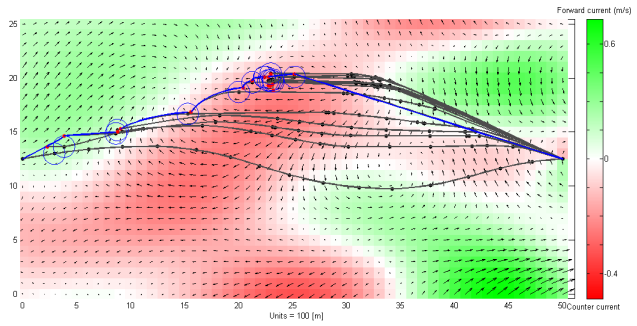

Fig. 11. Results with 15 waypoints with the original and the altered algorithms.

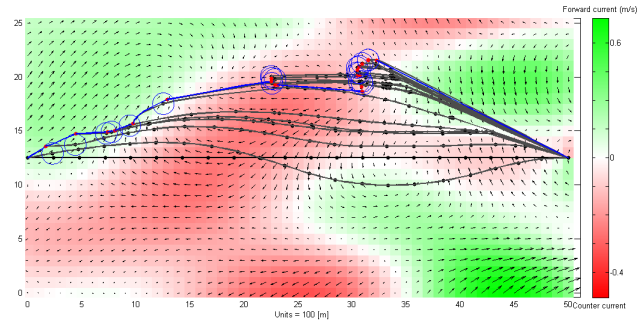

Fig. 12. Results with 20 waypoints with the original and the altered algorithms.

Looking at table I it is verified that despite the original time remains almost constant with any number of waypoints, both the directly computed time and the differential time present variations for different numbers of these points.

Comparing the obtained values with the straight line travel time (1 h 26 min 58 sec), the original algorithm wins over the implemented one only for a smaller number of points (5 points). Therefore, one can conclude the reoptimization requires more waypoints in order to present effective results.

V. Update of the currents' map based in the uncertainty

There are several possible causes for last chapter's uncertainty. The vehicle's odometry and the sensors' errors, amongst others, are typically well known errors and present values within a certain range for which one can perform some corrections.

However water currents present more unpredictable behavior. Also errors tend to be much higher when compared to the aforementioned and, for their randomness, harder to control.

Therefore, in this section the currents' map change is considered to be the main cause for the computed uncertainty. A proposal of how one could broaden the application of the previously computed uncertainty to update the currents' map at each check point throughout the path is performed.

Updating the currents' map is particularly important in this problem resolution, since the optimization objective function

No. Pts	Original script	Directly computed	Differential
5	1 h 25 min 25 sec	1 h 29 min 3 sec	1 h 18 min 30 sec
10	1 h 25 min 19 sec	1 h 23 min 48 sec	1 h 27 min 11 sec
15	1 h 25 min 28 sec	1 h 25 min 55 sec	1 h 23 min 26 sec
20	1 h 25 min 30 sec	1 h 23 min 27 sec	1 h 27 min 29 sec

TABLE I. Comparison between the results attained with the original script and the altered one.

represents the total travel time and this one is dependent on the currents' speed.

A. Base Concept

In previous sections the optimization algorithm was explained and the uncertainty incorporated in it. At every check point the optimization algorithm takes the difference between the position the AUV estimated to have reached and its real position. This difference is called uncertainty.

In order to obtain the new current field a weighted average is performed, as shown in the following equations:

$$d = \frac{distance\_to\_end}{total\_distance} \tag{22}$$

$$newField = d * (P_{rand} - P_{check}) + oldField \tag{23}$$

where

d: weight, representing how far one is from the destination

$distance\_to\_end$: distance from the current check point to the destination

$total\_distance$: total travel distance

$newField$: updated currents' map

$P_{rand} - P_{check}$: difference between the estimated position and the real one, *i.e.* uncertainty

$oldField$: currents' map one wants to update

Closer to the starting point, the uncertainty should be bigger since one doesn't know if the field estimate is correct or not. Therefore, the uncertainty weight is higher and the vector field undergoes a bigger alteration. As one approaches the destination and hence have a more accurate and updated map the uncertainty weight decreases and the vector field's increases, being the update almost null.

B. Implementation and results

The aforementioned methodology was implemented and in table II a comparison between the computed times with and without considering currents changing over time is presented.

Since currents present a slow variation over time and the simulation has a relatively small duration (1 h 25 m approximately), the effect of these variations is not very pronounced in the resulting computation of times. However, in some cases the different vector field produces the reduction of until 2 m in the computed time (if the update of the vector field was in

	Direct time	Direct time with currents
5	1 h 29 m 3 s	1 h 26 m 49 s
10	1 h 23 m 48 s	1 h 23 m 20 s
15	1 h 25 m 55 s	1 h 24 m 52 s
20	1 h 23 m 27 s	1 h 23 m 26 s
	Differential	**Differential with currents**
5	1 h 18 m 30 s	1 h 19 m 3 s
10	1 h 27 m 11 s	1 h 25 m 53 s
15	1 h 23 m 26 s	1 h 23 m 28 s
20	1 h 27 m 29 s	1 h 27 m 29 s

TABLE II. COMPARISON BETWEEN THE RESULTS ATTAINED WITH AND WITHOUT CONSIDERING THE CURRENTS' CHANGE EFFECT.

the opposite direction of the movement instead of decreasing, an increase of the computed times would be seen).

Also, the new vector field was obtained. In figure 13 one can see the final estimate of the total path described by the AUV, in blue, and the new estimated vector field, for 20 waypoints.

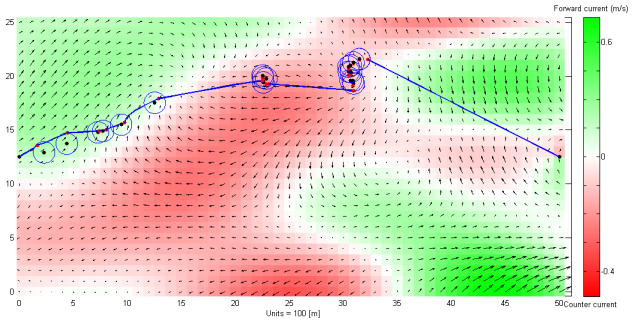

Fig. 13. Final results with 20 waypoints with a time varying field.

VI. CONCLUSIONS AND FUTURE WORK

In this paper a path planning algorithm for marine robotics was presented. This method has the particularity of predicting the possibility of some deviation from the computed path. At each check point the uncertainty is quantified and a new optimal path is computed for the remaining way points until the end, allowing one to have the best possible path from each mission check point.

The total travel time was computed in two different ways. The most accurate one, the direct computation of time over each segment, between check points, requires a higher number of points to refine the interpolation. Therefore, although the differential time computation is not as accurate as the optimal time procedure it is close enough to the real value and much less computationally costly.

Also a prediction of the vehicle's true path was proposed, to give some intuition about the deviations effects on the movement.

Finally, the vector field update based on the difference between the computed optimal point and the "real" position of the check point, considering that it changed over time because of a change of the currents map was implemented and tested.

There are several improvements and future implementations which one could perform. From the initial list on section IV only the uncertainty was implemented. Therefore, the script responsible for the water currents' definition could be customizable (*e.g.* reading a text file and converting it in a vector field). Also other optimization procedures could be tested and the 'active-set' method switched for another or even the *fmincon* function could be replaced.

Another future implementation would be to convert the current script in order to have a 3D optimization, since water currents vary with depth.

ACKNOWLEDGMENT

This work is financed by the ERDF European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT Fundao para a Ciłncia e a Tecnologia (Portuguese Foundation for Science and Technology) within projects FCOMP-01-0124-FEDER-015416 (COGNAT-PTDC/MAR/112446/2009) and FCOMP-01-0124-FEDER-022701.

REFERENCES

[1] B. Garau and A. Alvarez, "Path Planning of Autonomous Underwater Vehicles in Current Fields with Complex Spatial Variability : an A *," no. April, pp. 194–198, 2005.

[2] C. Pêtrès, Y. Pailhas, P. Patrón, Y. Petillot, J. Evans, and D. Lane, "Path Planning for Autonomous Underwater Vehicles," vol. 23, no. 2, pp. 9–13, 2007.

[3] R. Smith and M. Dunbabin, "Controlled drift: An investigation into the controllability of underwater vehicles with minimal actuation," in *Proceedings of the Australasian Conference on Robotics and Automation 2011*. Australian Robotics & Automation Association, 2011, pp. 1–10.

[4] J. Witt, M. Dunbabin, C. I. C. T. Centre, and P. O. Box, "Go with the Flow : Optimal AUV Path Planning in Coastal Environments Autonomous Systems," 2008.

[5] D. Kruger, R. Stolkin, A. Blum, and J. Briganti, "Optimal auv path planning for extended missions in complex, fast-flowing estuarine environments," *Robotics and Automation 2007 IEEE International Conference on*, no. April, pp. 4265–4270, 2007.

[6] T. van den Boom and B. D. Schutter, "Optimization in Systems and Control SC4091," 2011.

[7] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. Wiley-interscience, 2006.

[8] Jnsson,Ulf and Trygger,Claes and gren,Petter, "Optimal Control, Lecture notes for Optimization and Systems Theory," 2011.

[9] [Online]. Available: http://bocop.saclay.inria.fr/

[10] [Online]. Available: http://mechatronics.ece.usu.edu/riots/

[11] [Online]. Available: http://abs-5.me.washington.edu/ocp/

[12] [Online]. Available: http://www.psopt.org/

[13] [Online]. Available: http://www.mathworks.com/products/optimization/

[14] T. MathWorks, April 2013. [Online]. Available: http://www.mathworks.com/videos/finding-optimal-path-using-optimization-toolbox-68958.html

[15] February 2013. [Online]. Available: http://www.mathworks.com/help/optim/ug/fmincon.html#f854721

[16] A. G. Thomas Coleman, Mary Ann Branch, "Optimization toolbox user's guide," Math Works, January 1999.

[17] J. Nocedal and S. Wright, *Numerical optimization*. Springer.