2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Adapting the Wavefront Expansion in Presence of Strong Currents

Michaël Soulignac[*,**] Patrick Taillibert[*] Michel Rueher[**]

[*] THALES Aerospace
2 Avenue Gay Lussac
78852 Elancourt, France
{firstname.lastname}@fr.thalesgroup.com

[**] Nice Sophia Antipolis University
I3S/CNRS, BP 145
06903 Sophia Antipolis, France
rueher@essi.fr

Abstract— **The wavefront expansion is commonly used for path planning tasks and appreciated for its efficiency. However, the existing extensions able to handle currents are subject to incorrectness and incompleteness issues when these currents become strong. That is, they may return physically infeasible paths or no path at all, even if a feasible path exists. This behavior endangers the robot, especially in a dynamic replanning context. That is why we propose a new extension called *sliding wavefront expansion*. This algorithm, combining an appropriate cost function and continuous optimization techniques, guarantees the existence of a path with an arbitrary precision.**

I. INTRODUCTION

Autonomous robots are more and more used to collect data in hostile or hardly accessible areas. In a civil context, these data may concern rescue tasks (locating survivors after a natural hazard) or prevention tasks (keep a watch on fragile areas, such as forests). In military missions, they may concern surveillance tasks (about enemy installations or troops).

To perform these complex tasks, robots are guided by on-board planners. These planners have to be very reactive, because the environment is often changing or unknown. Thus, they implement very simple (but fast) algorithms, such as the wavefront expansion [3]. This algorithm is computationally efficient, but ignores or underestimates the weather conditions, in particular currents.

However, in the case of Unmanned Air Vehicles (UAVs) or Autonomous Underwater Vehicles (AUVs), which are generally small or slow, the impact of (air or water) currents is significant. In this context, several extensions of the original wavefront expansion have been proposed [5][7], but they become incorrect in presence of strong currents. That is to say, they may provide a path which is not physically feasible by the robot. This behavior is due to the use of invalid cost functions, which consider some cells as reachable, whereas it might not be the case.

Using valid cost functions allows to eliminate incorrectness issues, but leads to incompleteness ones: the algorithm may fail to find a path, even if one exists. This behavior is due to the discrete motion model, imposing on the robot to move from a cell to another.

Both behaviors are dangerous for the robot, because they can lead to an energy breakdown or worse, to collisions. Consequently, we propose a new algorithm called *sliding wavefront expansion*, mixing a valid cost function and a continuous motion model: (1) the cost function represents the actual travel time of the robot in currents; (2) costs are propagated among new entities, called *sliders*, which positions are computed by continuous optimization techniques.

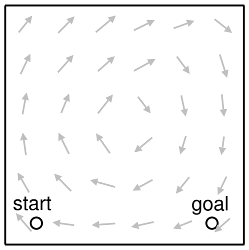

Fig. 1. An example of environment with currents. Each current node is represented by a grey arrow.

II. PROBLEM STATEMENT

A robot, moving in a planar environment from a start site, has to reach a goal site in a minimum time taking currents into account, as depicted in figure 1.

The robot and sites are modeled by single points, and the environment by a 2-D Euclidean space E. We denote $R = (O,\vec{x},\vec{y})$ the frame embedded in E, (u_x,u_y) the coordinates of a vector $\vec{u}$ in R and u its modulus.

The current can be seen as a 2-D vector field $\vec{c}$, known either by measurement or forecasting, and hence discontinuous. They are defined on the nodes of a mesh (not necessary regular), called *current nodes*. The mean distance between current nodes may correspond to the resolution of measures or the precision of the forecast model.

The robot's velocity relative to R is denoted $\vec{v^R}$, and its velocity relative to the current $\vec{c}$ is denoted $\vec{v^c}$. Applying the velocity composition law, these two quantities are linked by the following relation:

$$\vec{v^R} = \vec{v^c} + \vec{c} \tag{1}$$

The modulus v^c is assumed constant; it depends on the robot's engine capabilities.

Our problem consists in planning the time-optimal path between the start and goal sites, given (1) the value of v_c and (2) a finite number of current nodes.

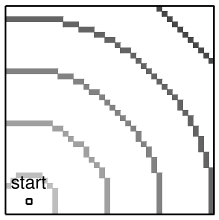

Fig. 2. Different stages of the wavefront expansion in the environment of figure 1, discretized into a 40 × 40 grid, without currents. The 8-neighborhood and the Euclidean metric are used for cost propagation.

III. THE WAVEFRONT EXPANSION

A. Description

In [6], Jarvis proposed to exploit a computer vision technique, called distance transform, for path planning tasks. This technique consists in propagating the Manhattan distance required to reach a goal cell in a regular grid, corresponding to a discretized representation of the environment. After this phase, a path to the goal can be built from any cell to the goal, by applying the hill climbing algorithm.

Later, Dorst and Trovato [3] generalized this algorithm to a space of any dimension, using any neighborhood and any metric. Figure 2 depicts a wavefront expansion using a neighborhood of size 8 and the Euclidean distance for the metric. In this case, the expansion is isotropic. The successive wavefronts are thus discretized circles.

The wavefront expansion can be interpreted in various ways. It can be seen as a distance transform (Jarvis' viewpoint), an application of the Dijkstra's algorithm (Dorst's viewpoint) or a numerical potential field method (Barraquand's viewpoint [1]).

B. Extensions to currents

To handle currents, specific cost functions have been proposed [5][7]. They lead to anisotropic wavefront expansions (see figure 3 on next page), which gives better paths (in terms of travel time) than isotropic expansions.

However, in presence of strong currents, incompleteness or incorrectness issues may raise, as explained in the next section.

IV. ISSUES DUE TO STRONG CURRENTS

We consider that a current is *strong* when its velocity is greater than the robot's velocity. More formally, every current $\vec{c}$ verifying $c \geq v^c$ is qualified as strong.

A. Incorrectness issues

We show here that the specific cost functions proposed in literature to handle currents are *invalid*: they consider some cells as reachable, whereas moves towards these cells are physically impossible. Unfortunately, the use of invalid cost functions may lead to incorrect planners, providing partially infeasible paths.

To illustrate this fact, let us apply the functions proposed by Pêtrès and Garau to the following example:

"The robot has to perform a unitary move $\vec{d} = 1\vec{x}$ between two cells, with a velocity $\vec{v^c} = 100\vec{x}$, relative to a slightly faster current $\vec{c^{opp}} = -120\vec{x}$ ".

Of course, this move is impossible, because the actual robot's velocity (relative to the ground) is $\vec{v^R} = -20\vec{x}$.

Pêtrès [7] proposed to use the following composite cost function:

$$\tau_1 = \tau_1^{dist} + \alpha \tau_1^{cur} \qquad (2)$$

This cost function can be interpreted as follows: the quantity τ_1^{dist} measures the traveled distance, and τ_1^{cur} the angular difference between the robot heading and the direction of currents. Therefore, τ_1 represents a compromise (tuned by α) between traveling the minimal distance and pointing in the same direction than currents.

In our particular example, we have:

$$\begin{cases} \tau_1^{dist} & = & d \\ \tau_1^{cur} & = & 1 - \langle \tau_1 \vec{x} \cdot \vec{c^{opp}} \rangle / Q \end{cases}$$

where $Q = (d + 2\alpha)c^{max}$ is a normalization term (c^{max} denoting the maximal current strength), α a positive gain and $\langle \cdot \rangle$ the scalar product.

Thus, equation 2 becomes:

$$\tau_1 = \frac{d + \alpha}{1 + (\alpha / Q) c_x^{opp}}$$

Using $c^{max} = c^{opp}$ and $d = 1$, the cost of move $\vec{d}$ without current is $\tau_1 = 1 + \alpha$. Adding $\vec{c^{opp}}$, this cost becomes $\tau_1' = (1+\alpha)/(1-\alpha/(1+2\alpha)) = k \cdot \tau_1$. In other words, in presence of $\vec{c^{opp}}$, the cost of $\vec{d}$ is penalized by a factor k equal to:

$$k = 1/(1 - \alpha/(1+2\alpha)) \in [1, 2[$$

However, in spite of this penalization, the move $\vec{d}$ is still considered as possible.

Garau [5] proposed the cost function τ_2 defined by:

$$\tau_2 = d / ||\vec{v^c} + \vec{c^{opp}}||$$

where $\vec{v^c}$ is the robot velocity relative to the current and $||\cdot||$ the norm operator.

The idea of this cost function is to reflect the impact of currents on the robot's velocity. A pushing current globally implies a velocity increase, and an opposite one a decrease.

Without current, the cost of move $\vec{d}$ is $\tau_2 = 1/100$. Adding $\vec{c^{opp}}$, this cost becomes $\tau_2' = 1/20 = 5\tau_2$. In other words, in presence of $\vec{c^{opp}}$, the cost of $\vec{d}$ is penalized by a factor 5. This penalization is more prohibitive than the Pêtrès' one, but the move is still considered as possible.

To sum up, both functions penalize physically impossible moves by a factor γ, instead of forbidding them. In those conditions, planners based on these functions may return a path containing infeasible parts, even if an entirely feasible path exists.

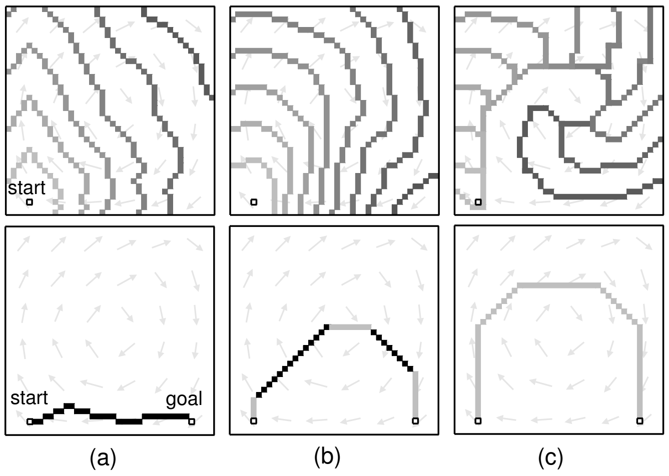

Fig. 3. Results obtained with $c = 120$ km/h and $v^c = 100$ km/h, applying the following cost functions: (a) Pétrès (τ_1); (b) Garau (τ_2); (c) ours (τ_3). The upper part depicts different stages of the wavefront expansion; the lower part the paths found. Feasible parts are drawn in grey and infeasible parts in black.

This is illustrated in figure 3. The path guided by Pétrès' cost function is completely infeasible. The one guided by Garau's is better (because impossible moves are more severely penalized) but some parts remain infeasible.

However, there exists an entirely feasible path, depicted in figure 3c. It has been obtained by using our cost function τ_3, introduced it the next subsection.

B. Incompleteness issues

In this part, we first introduce a valid cost function τ_3. Then we show that the use of this function solves the previous problem but brings a new one. Indeed, applying a valid function on discretized elements (the cells of the environment) leads to an incomplete planner, which may fail to find a feasible path, even if such a path exists.

Contrary to the functions τ_1 and τ_2 above, the function τ_3 we propose here models the *actual* travel time of the robot. In particular, τ_3 has the fundamental property to correctly capture impossible moves.

Let us consider a move $\vec{d}$, in a current $\vec{c}$. Using equation 1, the duration τ_3 of this move verifies: $\vec{d} = (\vec{c} + \vec{v^c}) \cdot \tau_3$. Contrary to Garau, we do not directly apply the norm operator to this equality, but we project it on x and y axis:

$$\left\{ \begin{array}{l} d_x = (v^c_x + c_x) \cdot \tau_3 \\ d_y = (v^c_y + c_y) \cdot \tau_3 \end{array} \right. \tag{3}$$

The relation $v^{c2} = v^{c2}_x + v^{c2}_y$, allows us to eliminate v^c_x and v^c_y in equation 3, leading to the following relation:

$$(d_x - c_x \cdot \tau_3)^2 + (d_y - c_y \cdot \tau_3)^2 = v^{c2} \cdot \tau_3^2 \tag{4}$$

Solving this second degree equation gives[1]:

$$\tau_3 = \frac{-(c_x \cdot d_x + c_y \cdot d_y) + \sqrt{\Delta}}{v^{c2} - c^2} = \frac{\sqrt{\Delta} - \langle \vec{d} \cdot \vec{c} \rangle}{v^{c2} - c^2} \tag{5}$$

[1]It can be shown that (5) is always the positive root of (4).

where $\Delta = v^{c2} \cdot (d_x^2 + d_y^2) - (c_x \cdot d_y - c_y \cdot d_x)^2$.

Note that in the particular case of $v^c = c$ we simply have:

$$\tau_3 = \frac{d^2}{2\langle \vec{d} \cdot \vec{c} \rangle} \tag{6}$$

Like Pêtrès' and Garau's cost functions, τ_3 naturally incites the robot to point in the direction of the current: if the scalar product $\langle \vec{d} \cdot \vec{c} \rangle$ increases, τ_3 decreases.

Moreover, τ_3 is defined only if $\Delta \geq 0$. This condition allows to determine the region the robot can reach, in presence of the current $\vec{c}$.

If the robot is faster than the current (i.e. if $v^c \geq c$), this region is unlimited. Otherwise, this region is delimited by a cone, called *accessibility cone*. We can show that this cone forms an angle α^{max} equal to:

$$\alpha^{max} = 2 \cdot \arctan\left(\frac{v^c}{\sqrt{c^2 - v^{c2}}}\right)$$

The move $\vec{d}$ is thus physically feasible if and only if the angle α between vectors $\vec{d}$ and $\vec{c}$ lies in $A = [-\alpha^{max}/2, \alpha^{max}/2]$.

When α tends towards the bounds of the domain A, the cost τ_3 tends towards $+\infty$. Therefore, the cost of every move $\vec{d}$ such that $\alpha \notin A$ is set to $+\infty$.

The use of such a cost function guarantees that the resulting paths are entirely feasible. However, the realism of τ_3 is not compatible with a discretized world. Indeed, problems appear when propagation directions are close to the borders of the accessibility cone: these directions are deleted because they are invalid, but the remaining direction may not be sufficient to reach the goal.

This point is illustrated in figure 4. The angle of the accessibility cone is about 84°, which eliminates the horizontal and vertical moves. The only remaining diagonal does not allow to reach the goal. Consequently, the planner answers that no feasible path exists, whereas the dotted path is entirely feasible. This last path has been obtained by using the sliding wavefront expansion, proposed in section V).

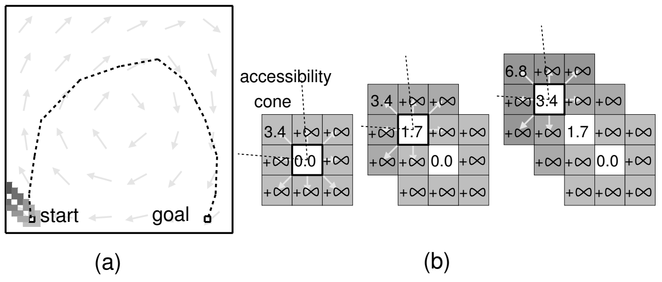

Fig. 4. Same situation than in figure 3c, with $c = 150km/h$. (a) different stages of the wavefront expansion; (b) focus around the start cell. Only one diagonal of the grid is explored; the rest is considered as unreachable, in particular the goal point. However, the dotted line path is feasible.

In response, a first idea would consist in using smaller cells. However, whatever the size of the cells, the number of neighbors of each cell stays the same. Therefore, exactly the same propagation directions will be usable during the wavefront expansion, and the problem will remain.

Increasing the size of the neighborhood appears to be a better idea, but it is computationally expensive and simply reduces the visible effects of the problem, without solving it.

As a matter of fact, the cause of the problem is deeper. It concerns the nature of the wavefront expansion itself. Indeed, in order to propagate costs, the environment is first discretized. However, this discretization step, which seems quite natural, is *the* source of incompleteness. Whatever the type or size of cells, each of them has a finite neighborhood, which implies a finite number of propagation directions.

To actually solve the problem, we think that costs should not be propagated in a discrete domain, but in a continuous one. Based on this idea, we propose a new algorithm, called the *sliding wavefront expansion*.

V. THE SLIDING WAVEFRONT EXPANSION

In this section we first introduce two concepts: *Elementary Current Areas* (ECAs), extending punctual values of current within small polygons, and *sliders*, containing the potential viapoints of the path.

Then, we explain the new wavefront expansion algorithm, optimizing the state of sliders at each step.

A. Introducing ECAs

Using a regular grid to discretize the environment seems not pertinent, because the current nodes are not necessary placed in a regular way in the environment. Therefore, many cells could have the same value of current, which represents a useless redundancy of information.

To guarantee the minimal number of cells, we propose the concept of *Elementary Current Area* (ECA). An ECA is an indivisible region of the environment, in which the current is homogeneous. Each ECA contains a unique current node. The value of this node is extended to the whole area.

ECAs are computed by building the Voronoï diagram [4] around the current nodes. This diagram is made up of line segments which are equidistant to the nodes (see figure 5).

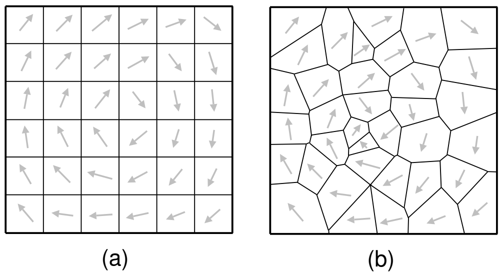

Fig. 5. Illustration of ECAs for two distributions of current nodes (grey arrows): (a) uniform and (b) non-uniform.

It is clear that ECAs (like grid cells in previous approaches) maintain discontinuity in values of currents. That is, if we consider the border of an ECA, there is one value of current on one side, and another one on the other side, without a smooth transition. This choice is voluntary, for the following reasons:

1) Borders of ECAs allow us to introduce sliders, new entities used in the cost propagation.
2) Since current nodes already includes an error (most often, they are forecast), a very fine modeling of currents seems meaningless.

However, if the application requires to plan a path very precisely, the above adaptations are possible:

- Inserting artificial current nodes between existing ones using interpolation techniques.
- Adapting the cost function τ_3, in order to model a continuous variation of currents.

B. Introducing sliders

We associate a *slider* at each ECA border. Like the graphical component, a slider is made up of a knob which can slide on a rail, with respects to its bounds. More formally, a slider is denoted S_i and has the following attributes (illustrated in figure 6):

- A border B_i, of equation $a \cdot x + b \cdot y + c = 0$, modeling the rail;
- Two points $B_i^- = (x_i^-, y_i^-)$ and $B_i^+ = (x_i^+, y_i^+)$, modeling the bounds;
- A variable l_i, the curvilinear abscissa on B_i, modeling the position of the knob;
- A viapoint $V_i = (x_i, y_i)$, modeling the knob. The values of x_i and y_i can be deduced from l_i by:

$$\begin{aligned} x_i &= x_i^- + \sqrt{l_i^2/(1+(a/b)^2)} \\ y_i &= y_i^- + (-a \cdot x_i - c)/b \end{aligned} \tag{7}$$

- A domain D_i for l_i values: $D_i = [0, l_i^+]$, modeling the rail bounds;
- The pair $A_i = \{ECA_1, ECA_2\}$ of the two adjacent ECAs to S_i.

The notion of neighborhood can be defined between viapoints as follows: two viapoints V_i and V_j are neighbors if and only if $A_i \cap A_j \neq \emptyset$

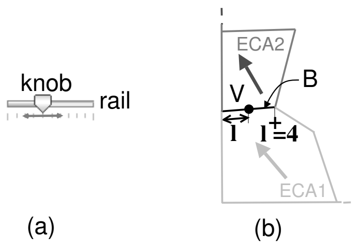

Fig. 6. Illustration of sliders: (a) the graphical component and (b) the mathematical item associated to each ECA border.

C. New path representation

Since ECAs are an exact cell decomposition of the environment, the path P between start and goal sites goes necessary through a set of adjacent ECAs.

As mentioned before, an ECA is characterized by a constant current. This implies that, within each ECA, the triangular inequality (concerning the travel time) is verified. Thus, the fastest way to go through an ECA is to follow a straight line linking two borders, i.e. a straight line between two neighbor viapoints.

Applying this reasoning to all traveled ECAs, we can deduce that P is a succession of line segments $[Start, V_1]$, $[V_1, V_2]$, ... , $[V_{n-1}, V_n]$, $[V_n, Goal]$. For short, we will model P as a list of viapoints V_i, with $V_0 = start$ and $V_{n+1} = goal$.

D. Cost propagation

In the propagation process, grid cells introduced in subsection III-A are replaced by sliders. In other terms, the costs are propagated among sliders and no more among cells.

The main difference between cells and sliders comes from the mobility of their viapoints. Indeed, in the case of cells, viapoints are static. Their position is defined in advance; generally set to the center of the cell. On the contrary, viapoints of sliders have one degree of freedom: as explained before, they can slide on their rail.

This mobility allows to solve the incompleteness issues due to discretization, mentioned in subsection IV-B. Indeed, from a viapoint V_i all propagation directions within the accessibility cone are available, using different sliders, as shown in figure 7. In particular, these propagation directions can be arbitrary close to the borders of the cone. Thus the existence of a path can be guaranteed with an arbitrary precision.

Figure 8 illustrates this fact: the sliding wavefront expansion succeed in finding a feasible path, where the classical wavefront failed (see figure 4).

E. Cost evaluation

Let us consider a viapoint V_i, which predecessors are V_j ($j \in [0, i-1]$). The cost T_i associated to V_i is given by[2]:

$$T_i = \min_{l_j} \sum_{j=0}^{i-1} \tau_3^{j,j+1} \tag{8}$$

[2]It can be shown that this cost corresponds to a global minimum, because all $\tau_3^{j,j+1}$ functions are convex.

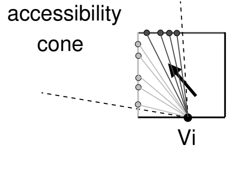

Fig. 7. Illustration of the continuous motion model (bottom-left corner of fig. 5). Using different sliders, all the propagation directions are available in the accessibility cone.

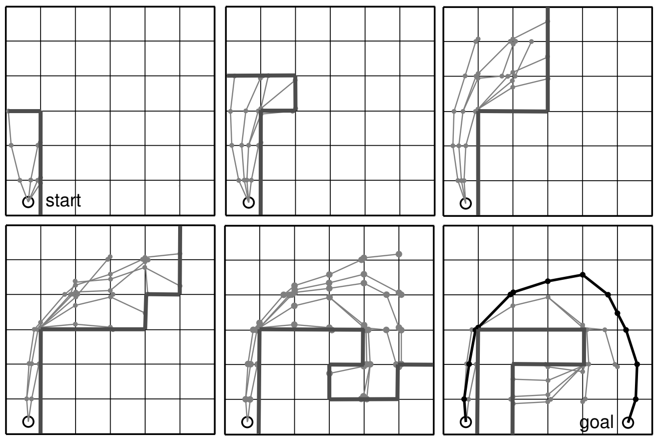

Fig. 8. The sliding wavefront expansion, using ECAs of 5a, with $c = 150$ km/h, and $v^c = 100$ km/h. Contrary to the figure 4, the wavefront of sliders (in dark grey) reaches the goal, thanks to the ability of viapoints to slide on borders. Potential paths are drawn in light grey and the final path in dark. This expansion is similar to the one of figure3c, with bigger cells.

where $\tau_3^{j,j+1}$ is the cost function given by equations 5 and 6, modeling the travel time between viapoints j and $j+1$. We have $d_x = x_{j+1} - x_j$, $d_y = y_{j+1} - y_j$. c is the current within the common ECA to sliders S_j and S_{j+1} (given by $A_j \cap A_{j+1}$).

As shown in equation 7, x_j and y_j can be expressed in function of l_j. Therefore, the quantity T_i only depends on variables l_j.

In other words, the cost of the viapoint V_i is computed by minimizing the i-variables function T_i. Since there are only bound constraints[3] on variables l_i, this minimization task is performed analytically or numerically, depending on the status of bound constraints:

- If all bound constraints are inactive, the optimal position of viapoints can be expressed analytically, by solving the equation $\vec{\nabla} T_1 = 0$ ($\vec{\nabla}$ denoting the gradient operator). Let us consider the example of $V_1 = (l_1, y_1)$ in figure 9a. The equation $\vec{\nabla} T_1 = 0$ reduces to $dT_1/dl_1 = 0$. Solving this last equation gives:

  $$l_1 = l_{start} + (y_1 \cdot c_x)/(v^c + c_y)$$

  This equation reflects that the robot directs its velocity vector $\vec{v^c}$ along y axis, letting itself derive on x axis.
- If some bound constraints are active, computations become too complex to be done analytically. Thus, we apply a numerical approach called *projected gradient* [2]. This algorithm is similar to a gradient descent, but restricted to the valid domain by a projection process, illustrated in figure 9b.
  It is important to note that this algorithm generally converges to the optimal solution, without reaching it ($\nabla T \to 0$ during iterations). To stop this convergence, we opted to the common condition $\nabla T < \varepsilon$.

[3]A priori, there are also constraints linked to the accessibility cone, i.e $\Delta \geq 0$ for all functions $\tau_3^{j,j+1}$. Instead of posting these constrains explicitly, each function $\tau_3^{j,j+1}$ is artificially continued outside its domain, with an appropriate form.

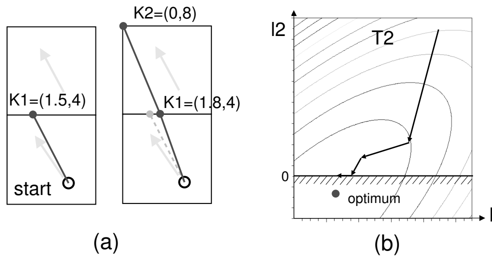

Fig. 9. (a) Focus around the start cell. First propagation step: V_1 is analytically evaluated. Second step: V_2 is numerically evaluated, by the projected gradient method, changing the position of V_1; (b) steps of the projected gradient descent on T_2 surface.

F. The algorithm

As the classical wavefront expansion, the sliding version is a Dijkstra-like algorithm: during an iterative process, the viapoint V_i with the lowest cost T_i is selected in the wavefront W and expanded. An expansion step consists in evaluating all the neighbors of V_i, and adding them in the wavefront. This process, detailed below, is repeated until the goal viapoint is selected for expansion.

```
SLIDING_WAVEFRONT_EXPANSION($Start, Goal, S, \varepsilon$)
  ▷ Input $Start$,$Goal$: viapoints
  ▷ Input $S$: list of sliders
  ▷ Input $\varepsilon$: desired precision on the gradient
  ▷ Auxiliary $W$: wavefront
  ▷ Auxiliary $N$: local neighborhood
  ▷ Auxiliary $V_i$: viapoint, with cost $T_i$
 1 Begin
 2   $W \leftarrow \{Start\}$
 3   do
 4     $V_i \leftarrow argmin_{V_k \in W}(T_k)$
 5     $W \leftarrow W \setminus V_i$
 6     $N \leftarrow$ NEIGHBORHOOD($V_i, S$)
 7     for each $V_j \in N$ do
 8       do
 9         MINIMIZE_COST($V_j, V_i$)
10         CHOOSE_PREDECESSOR($V_i$)
11       while $\nabla T_j \geq \varepsilon$
12       ADD_PREDECESSOR($V_j, V_i$)
13       $W \leftarrow W \cup V_j$
14   while $V_i \neq Goal$
15 End
```

The procedure NEIGHBORHOOD(V_i) returns all neighbors of V_i, using the definition of subsection V-B.

The procedure MINIMIZE_COST(V_j, V_i) computes the cost of a new viapoint V_j, coming from V_i, applying the continuous optimization techniques described in part V-E.

The procedures ADD_PREDECESSOR and CHOOSE_PREDECESSOR allow to handle the predecessors of V_j in two phases. These two phases are necessary because of the continuous motion model of sliders. Indeed, since V_j can slide on its border, its optimal predecessor can change, depending on the following ECAs, as shown in figure 10.

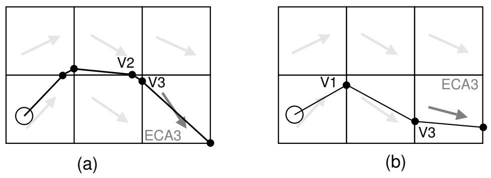

Fig. 10. Depending on the current present in ECA_3, the optimal predecessor of viapoint V_3 is either V_1 or V_2. However, when evaluating V_3, information about next ECAs is unknown.

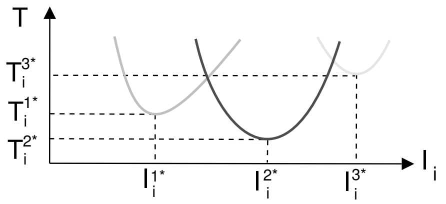

Fig. 11. Choice between 3 potential predecessors P_1, P_2, P_3 of viapoint V_i: (a) if $l_i \leq l_i^{1*}$, P_1 is chosen; (b) if $l_i^{1*} < l_i < l_i^{3*}$, P_2 is chosen; (c) if $l_i > l_i^{3*}$, P_3 is chosen.

The problem is that, by definition, information about future ECAs is unknown, when evaluating V_j.

Therefore, V_i is added to the list of potential predecessors of V_j, by the procedure ADD_PREDECESSOR(V_j, V_i).

Next, when V_j is evaluated, the procedure CHOOSE_PREDECESSOR(V_i) selects the best predecessor of V_i as follows:

Depending on the traveled ECAs, each predecessor P_k of V_i leads to a different cost function T_i^k. The minimum of each function T_i^{k*} is known. It is denoted T_i^{k*} and it is situated at l_i^{k*}. The procedure selects the predecessor P_k such that $l_i \in [l_i^{k-1*}, l_i^{k+1*}]$ and T_i^{k*} is minimal. This is illustrated in figure 11.

VI. EXPERIMENTAL RESULTS

The aim of this part is to compare the capability of the classical wavefront expansion and the sliding wavefront expansion to plan a valid path in currents. To do this, we applied the following procedure:

1) We collected wind charts W on Meteo France website[4] during three months, to constitute a sample of 90 realistic environments.

[4]http://www.meteofrance.com/FR/mer/carteVents.jsp

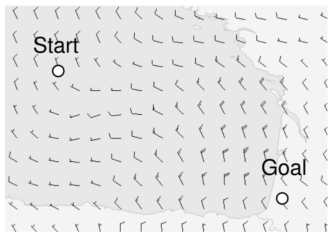

Fig. 12. An example of test-case.

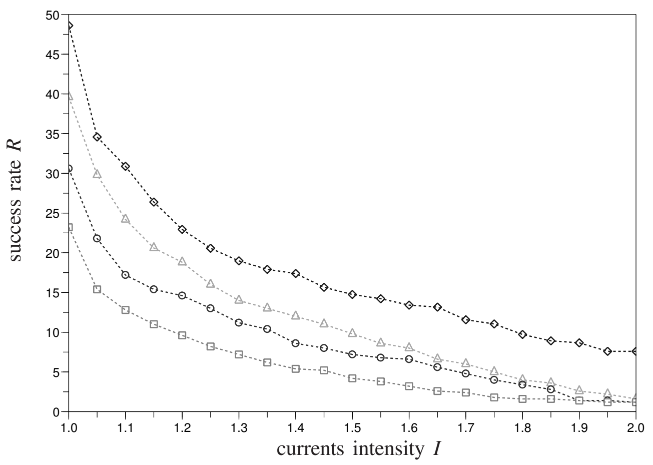

Fig. 13. Success rate curves, from up to bottom: (Sli) : diamonds, (Cla_3) : triangles, (Cla_2) : circles and (Cla_1) : squares.

2) We generated $N = 500$ test-cases. Each test-case $T = \{W, Start, Goal\}$ is built in the following way: (1) a wind-chart W is randomly chosen between the 90 possible ones; (2) the *Start* and *Goal* sites are randomly placed on W.

3) For each test-case, we varied the intensity I of currents, defined by:

$$I = \max\{c\}/v^c$$

In our tests, we chose $I > 1$ to voluntarily induce the incompleteness and incorrectness issues mentioned above.

Note that for such intensities, there naturally exists some test-cases where no feasible path exists (for instance, test-cases where the currents are globally against the robot).
As explained in section IV-B, $I = 1$ implies that only a half plan of the environment is reachable at each move. Therefore, each move has the probability of 1/2 to be invalid. In those conditions, about 50% of tests-cases are potentially impossible. This percentage increases with I.

4) For each intensity I, we applied the following algorithms to the N test-cases:
   - (Cla_1): classical wavefront expansion, using τ_1;
   - (Cla_2): classical wavefront expansion, using τ_2;
   - (Cla_3): classical wavefront expansion, using τ_3;
   - (Sli): the sliding wavefront expansion, using τ_3.

5) Finally, we computed the success rate R defined by:

$$R = \frac{N_{success}}{N} \cdot 100$$

where $N_{success}$ denotes the number of test-cases leading to a successful planning.
A planning is said *successful* if the algorithm succeeds in computing a valid path between the *Start* and *Goal* sites. Success rates are plotted in figure 13 for each algorithm.

The worst results are obtained by (Cla_1), which suffers from important incorrectness issues: more than 75% of computed paths were invalid, even for small intensities.

Then, results improves with the quality of the cost function. Indeed, since the cost function τ_2 penalizes impossible moves more severely than τ_1, success rates obtained by (Cla_2) are slightly better. (Cla_3) obtains even better results due to the use a valid cost function: since impossible moves are forbidden, only valid paths are returned. However, incompleteness issues occur, because of the discrete motion model of the robot. The algorithm fails to find a path, even on very simple instances.

As expected, the use of (Sli) considerably reduces this phenomenon. Since (Sli) combines a continuous motion model and a valid cost function, it obtains the best results. Its success rate of is about 10% upper than (Cla_3).

However, the price to pay is the computational efficiency. We roughly observed that the classical wavefront expansion performed on a 50×50 grid (the resolution used in our experiments) is about 5 times faster than the sliding wavefront expansion. This was expected, since the sliding wavefront expansion integrates some optimization processes, which are time-consuming, in particular in the cases of slow convergence. A deeper study of this aspect is in progress.

VII. CONCLUSION

In this paper, we showed that in presence of strong currents, using the existing extensions of wavefront expansion may lead to incorrectness or incompleteness issues. In response, we proposed a new extension, called the *sliding wavefront expansion*. Combining a valid cost function and a continuous motion model, this new extension succeeds in finding a path where the existing extensions may fail.

Further works will concern a deeper study on the properties of algorithm (notably the distance from the optimum and the time complexity), and also its extension in a 3-D space.

REFERENCES

[1] BARRAQUAND, J., LANGLOIS, B., AND LATOMBE, J.-C. Numerical potential field techniques for robot path planning. In *Transactions on Systems, Man, and Cybernetics* (1992), vol. 22, pp. 224–241.
[2] CALAMAI, P. H., AND MORE, J. J. Projected gradient methods for linearly constrained problems. *Mathematical Programming: Series A and B* (1987), 9–116.
[3] DORST, L., AND TROVATO, K. Optimal path planning by cost wave propagation in metric configuration space. In *Proceedings of SPIE-The International Society for Optical Engineering* (1988), pp. 186–197.
[4] FORTUNE, S. A sweepline algorithm for voronoi diagrams. In *Proceedings of the second annual symposium on Computational geometry* (1986), pp. 313–322.
[5] GARAU, B., ALVAREZ, A., AND OLIVER, G. Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an a* approach. In *Proceedings of the International Conference on Robotics and Automation* (2005), pp. 194–198.
[6] JARVIS, R. A. Collision-free trajectory planning using the distance transforms. *Mechanical Engineering Transactions of the Institution of Engineers 3* (1985), 187–191.
[7] PETRES, C., PAILHAS, Y., PATRON, P., PETILLOT, Y., EVANS, J., AND LANE, D. Path planning for autonomous underwater vehicles. *Transactions on Robotics 23* (2007), 331–341.