An Interval-based Sliding Horizon Motion Planning Method [⋆]

Julien Alexandre dit Sandretto [*] **Elliot Brendel** [**] **Alexandre Chapoutot** [*]

[*] *U2IS, ENSTA ParisTech, Université Paris-Saclay, 828 bd des maréchaux, 91762 Palaiseau, France (e-mail: {alexandre,chapoutot}@ensta.fr).*
[**] *ONERA - The French Aerospace Lab, Palaiseau F-91761, France (e-mail: elliot.brendel@onera.fr).*

Abstract: A new algorithm of motion planning based on set-membership approach is presented. The goal of this algorithm is to find a safe and optimal path taking into account various sources of bounded uncertainties on the dynamical model of the plant, on the model of the environment, while being robust with respect to the numerical approximations introduced by numerical integration methods. The main approach is based on a sliding horizon method to predict the behavior of the system allowing the computation of an optimal path. As an example, the motion planning algorithm is applied to an Autonomous Underwater Vehicle (AUV) case study, showing the benefit of the proposed approach.

Keywords: Formal synthesis, Reachability and safety analysis, Robust control.

1. INTRODUCTION

Motion planning algorithms are a center piece in the control framework of mobile robots as they contribute to give them the ability of autonomous behaviors. Furthermore, such class of algorithms is critical as a failure can cause the abort of the mission or can cause important amount of damage such as human loss. The validation of such algorithms is then mandatory in order to increase the confidence of the end users. However, those algorithms are also subject to constraints, *e.g.*, to reduce fuel consumption. In consequence, computing a safe path is usually not enough, and an optimal one is search to minimize some costs.

Moreover, one of the challenge in order to design robust and reliable motion planning algorithms is to take into account various sources of uncertainties. For example, the environment is not exactly know and some disturbances must be considered. Mathematical models of the mobile robots are not perfect, and they usually come from some simplifications in order to apply well-known control algorithms (*e.g.*, in linear cases) or efficient simulation methods. Lastly, computer-aided design produces approximated results as it is based on numerical methods which cannot produce the closed form solution of a problem, *e.g.*, the solution of an initial value problem for ordinary differential equations. The set-membership framework is suitable to deal with such kinds of uncertainties.

The last but often the most important source of uncertainty, the robot sensors cannot produce a global map of the environment. Indeed, the environment surrounding autonomous robot is usually built little by little as for example in SLAM approaches Leonard and Durrant-Whyte (1991). In consequence, motion planning algorithms have to be robust to partial information on the environment (*e.g.*, static obstacles) and should be incremental in order to produce pieces of trajectory to the goal objective. Sliding horizon approach is a good way to fulfill this requirement in the design of motion planner.

Contributions: The main contribution of this article is the combination of set-membership methods with an optimizing approach to define a motion planner acting on a partial map of the environment. Hence, a correct-by-construction algorithm is defined with the intrinsic properties to be robust to bounded uncertainties as it relies on set-membership approach Jaulin et al. (2001). Moreover, embedding the motion planning problem into a *Constraint Satisfaction Problems* (CSP) Rueher (2005), and more precisely into a global optimization framework Hansen (2003), the proposed algorithm produces an optimal free-collision paths with respect to a given cost function which is minimized. This CSP approach is also combined with a sliding horizon method to produce robust paths with a partial knowledge on the environment and having to deal only with finite horizon dynamical CSP problems. An application of the proposed motion planner on an AUV is given in Section 4 to show its relevancy.

Related Work: Motion planning is an active research area in Robotics and many methods have been developed. Artificial potential fields has been used in motion planning problems such as in Bemporad et al. (1996) but without producing optimal path. A popular approach is to use stochastic sampling to discretize the configuration space, *e.g.*, the *Rapidly-exploring Random Trees* (RRT) LaValle and Kuffner (2000) path planning algorithm and its many variants. The (asymptotic) optimality of the solution is provided by the optimal Rapidly-exploring Random Trees (RRT*) first proposed in Karaman and Frazzoli (2010). Other methods based on receding horizon approach have also been considered to produce optimal collision-free paths. For example, Mendes Filho and Lucet (2016) uses receding horizon in context of multi-robot or Wongpiromsarn et al. (2012) considers complex mission described by temporal logic.

[⋆] This work benefited from the support of the "Chair Complex Systems Engineering - École polytechnique, THALES, DGA, FX, DASSAULT AVIATION, DCNS Research, ENSTA ParisTech, Télécom ParisTech, Fondation ParisTech and FDO ENSTA".

While all these methods are efficient to produce collision-free paths, they usually did not take into account uncertainties and so the robustness of the solution is not guaranteed.

Uncertainty in motion planning has been considered mainly based on two representations: set-membership Page and Sanderson (1995); Pepy et al. (2009); Alexandre dit Sandretto et al. (2017) or co-variance matrices Lambert and Gruyer (2003); Censi et al. (2008). While the latter is able to find paths with a collision probability under a given threshold, set-membership approaches can guarantee safe trajectories under a bounded noise assumption. In Pepy et al. (2009); Alexandre dit Sandretto et al. (2017), a preliminary conceptual reliable and robust path planner based on RRT principles and solved in an set-membership framework, where all uncertainties are considered, bounded is introduced. Interval analysis principle along with graph algorithms were previously used Jaulin (2001) to find a collision-free shortest path for a polygonal rigid object in a given configuration. Nevertheless, these approaches build robust and safe paths but usually cannot produce optimal paths.

Contents: The paper is organized as follows. Some preliminary notions on motion planning problem, set-membership methods and sliding horizon method are introduced in Section 2. The main contribution of the paper is presented in Section 3 by formulated the problem and the presentation of the algorithms. In Section 4 a case study focus on AUV is described showing the relevance of the proposed approach. Conclusion and perspective are drawn in Section 5

2. PRELIMINARY NOTIONS

In this section, the main notions useful for our approach to solve the robust motion planning problem are introduced.

2.1 Dynamics of a vehicle

The dynamics of a vehicle, such as a car, a flight or a ship, can be modeled by differential equations to analyze its behavior. In the special case of autonomous vehicles - Unmanned Aerial Vehicle (UAV), Autonomous Underwater Vehicle (AUV) or Unmanned Ground Vehicle (UGV) - the dynamical modeling is very important to define the controller and the path planner. In this paper, it is assumed that the dynamics is modeled by nonlinear ordinary differential equations as the ones coming from the Newton's laws.

Starting from a given point at time zero, an initial value problem is defined by:

$$\dot{\mathbf{y}}(t) = f(t, \mathbf{y}(t), \mathbf{u}(t)), \text{ with } \mathbf{y}(0) = \mathbf{y}_0 \ , \tag{1}$$

where $\mathbf{y}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathbb{R}^m$ denote the vector of states and inputs, respectively.

2.2 Motion planning

A motion planning consists in producing a continuous path in the configuration space, that satisfies system dynamics (movement constraints), safety constraints (obstacles), inputs limitations, and possibly optimizes a cost linked to a given aspect of the movements. In a more enriched motion planning, the considered constraints can take into account the quality of ground, ocean stream, up-draft, seabed shape, etc. These constraints can be considered in connection with the task and/or the sensors. A cost function is often added. This latter, linked to states of the system (and not linked to inputs) can be a distance expressed in the configuration space or a more complex functional cost involving state and/or state derivatives.

2.3 Uncertainties and Validated Simulation

The simplest and most common way to represent and manipulate sets of values is *interval arithmetic* (see Moore (1966)). An interval $[x_i] = [\underline{x_i}, \overline{x_i}]$ defines the set of reals x_i such that $\underline{x_i} \leq x_i \leq \overline{x_i}$. $\mathbb{IR}$ denotes the set of all intervals over reals.

Interval arithmetic extends to $\mathbb{IR}$ elementary functions over $\mathbb{R}$. For instance, the interval sum, *i.e.*, $[x_1] + [x_2] = [\underline{x_1} + \underline{x_2}, \overline{x_1} + \overline{x_2}]$, encloses the image of the sum function over its arguments. An interval vector or a *box* $[\mathbf{x}] \in \mathbb{IR}^n$, is a Cartesian product of n intervals.

Validated numerical integration methods are interval counterpart of numerical integration methods. A validated numerical integration of a differential equation, as defined in (1) assuming piece-wise constant input, consists in a discretization of time, such that $t_0 \leqslant \cdots \leqslant t_{\text{end}}$, and a computation of enclosures of the set of states of the system $\mathbf{y}_0$, ..., $\mathbf{y}_{\text{end}}$, by the help of a guaranteed integration scheme. In details, a guaranteed integration scheme is made of

- an integration method $\Phi(f, \mathbf{y}_j, t_j, h)$, starting from an initial value $\mathbf{y}_j$ at time t_j and a finite time horizon h (the step-size), producing an approximation $\mathbf{y}_{j+1}$ at time $t_{j+1} = t_j + h$, of the exact solution $\mathbf{y}(t_{j+1}; \mathbf{y}_j)$, *i.e.*, $\mathbf{y}(t_{j+1}; \mathbf{y}_j) \approx \Phi(f, \mathbf{y}_j, t_j, h)$;
- a truncation error function $\text{lte}_\Phi(f, \mathbf{y}_j, t_j, h)$, such that $\mathbf{y}(t_{j+1}; \mathbf{y}_j) = \Phi(f, \mathbf{y}_j, t_j, h) + \text{lte}_\Phi(f, \mathbf{y}_j, t_j, h)$.

A validated numerical integration method is a two step method starting at time t_j and for which *i)* it computes an enclosure $[\tilde{\mathbf{y}}_j]$ of the solution of (1) over the time interval $[t_j, t_{j+1}]$ to bound $\text{lte}_\Phi(f, \mathbf{y}_j, t_j, h)$; *ii)* it computes a tight enclosure of the solution of (1) for the particular time instant t_{j+1}. There are many methods for these two steps among Taylor series and Runge-Kutta methods see Nedialkov et al. (1999); Alexandre dit Sandretto and Chapoutot (2016) and the references therein for more details.

As a result, validated numerical integration methods produce two functions depending on time

$$R : \begin{cases} \mathbb{R} \to \mathbb{IR}^n \\ t \mapsto [\mathbf{y}] \end{cases} \tag{2}$$

with for a given t_i, $R(t_i) = \{\mathbf{y}(t_i; \mathbf{y}_0) : \forall \mathbf{y}_0 \in [\mathbf{y}_0]\} \subseteq [\mathbf{y}]$, and

$$\tilde{R} : \begin{cases} \mathbb{IR} \to \mathbb{IR}^n \\ [\underline{t}, \overline{t}] \mapsto [\tilde{\mathbf{y}}] \end{cases} \tag{3}$$

with $\tilde{R}([\underline{t}, \overline{t}]) = \{\mathbf{y}(t; \mathbf{y}_0) : \forall \mathbf{y}_0 \in [\mathbf{y}_0] \wedge \forall t \in [\underline{t}, \overline{t}]\} \subseteq [\tilde{\mathbf{y}}]$.

In Alexandre dit Sandretto et al. (2017), validated numerical integration has been combined with CSP tools. Indeed, functions $R()$ and $\tilde{R}()$ are abstracted, but guaranteed, solutions of (1). Therefore, the process of validated simulation, mixed with constraint programming and abstraction of time functions provides an efficient tool for the prediction of the system evolution.

2.4 Sliding Horizon Method

A sliding, or receding, horizon-based method is used when the future cannot be anticipated, such as in a moving environment or a progressively discovered environment.

In the proposed method, the predicted horizon needs to be prolonged, but long term validated simulations are difficult to obtain. A sliding horizon approach is then interesting to progressively discover the environment and by the way reducing the duration of the simulations. A sliding horizon based motion planning is then close to Model Predictive Control. The main difference is the desired result: MPC is a control synthesis, then it produces solutions in the control space (with cost on control variables), while motion planning is a path planner, then it produces solutions in the state space (with cost on states).

A sliding horizon-based method consists on computing a value (an optimal control or a path) from a time t_0 for a period T_0 (as long as possible) in order to anticipate the future, *i.e.*, the horizon, injecting this value in the system, and re-computing this value at time t_1, with $t_0 + t_1 < T_0$, for a new horizon period T_1 (T_1 may be equal to T_0). The initial condition of the problem on t_1 is the state of the system evaluated or measured (in the case of control synthesis) at t_1.

3. ROBUST AND GUARANTEED MOTION PLANNING BASED ON SLIDING HORIZON

3.1 Problem Formulation

A dynamical system as defined in (1), where $\mathbf{y}$ is the state and $\mathbf{u}$ is an input function, is considered.

Remark 1. The input function $\mathbf{u}$ is used to modify the motion of the system and it is not necessary a control variable. For example, in the case of an autonomous car, $\mathbf{u}$ would be the Euler's angles and the speed of the car (bounded by the dynamical limits of the car), while the control variables could be the angle of wheels and the engine regime.

In addition, a constraint function $h : \mathbf{y} \mapsto h(\mathbf{y})$ and a cost function $g : \mathbf{y} \mapsto g(\mathbf{y})$ are defined, constraints coming from safety properties (obstacle avoiding in particular) and cost from task constraints for example. Thus, the problem is to find the trajectory $\mathbf{y}(t,\mathbf{u})$ depending on $\mathbf{u}$ which is the solution of

$$(P) : \begin{cases} \operatorname{argmin}_{\mathbf{u}} \quad g(\mathbf{y}(t,\mathbf{u})) \\ \dot{\mathbf{y}}(t) = f(\mathbf{y}(t),\mathbf{u}(t)) \\ h(\mathbf{y}(t,\mathbf{u})) < 0 \end{cases} .$$

Remark 2. Only the solution function $\mathbf{y}(t,\mathbf{u})$, *i.e.*, the path, is relevant for the motion planning problem. Despite, the computation is based on the input function $\mathbf{u}$ in Problem (P), the main result will be $\mathbf{y}(t,\mathbf{u})$.

3.2 Overview of the Algorithm

The robust and guaranteed motion planning using sliding horizon proposed in this paper is defined by the steps:

(1) an input $\mathbf{u}_1$ solution of (P) on a time span $[0,T]$, with T the prediction period, and from the initial state $\mathbf{y}_0$, is computed using a prediction procedure and an optimization algorithm;
(2) a simulation on a time span $[0,T']$, with T' the sliding period ($T' < T$), and from the initial state $\mathbf{y}_0$, is performed. It provides the trajectory $\mathbf{y}_1(t)$ driven by the input $\mathbf{u}_1$. This step is used to construct the path;
(3) an input $\mathbf{u}_2$ solution of (P) on a time span $[T', T'+T]$, and from the initial state $\mathbf{y}_1(T')$, is computed. This is the first step slided of T';
(4) as in the second step, a simulation on a time span $[T', 2T']$, and from the initial state $\mathbf{y}_1(T')$ is performed. It provides the trajectory $\mathbf{y}_2(t)$ driven by the input $\mathbf{u}_2$. This path is concatenated with the one obtained at step 2);
(5) and so forth.

To find the input $\mathbf{u}_i$, we browse through the set of the dynamically acceptable inputs, assuming that it is a finite set, until the input which drives to $\mathbf{y}(t)$ satisfies the constraint $h(\mathbf{y}(t)) < 0$ and minimizes the cost $g(\mathbf{y}(t))$, $\forall t \in [(i-1)T', iT']$ is found.

3.3 Algorithms

The overall method is divided in two algorithms, the main one (see Algorithm 1) is the global procedure handling the sliding horizon progress. The second one (see Algorithm 2) solves Problem (P). A third algorithm (see Algorithm 3) which improved the second one is also given.

Algorithm 1 Sliding horizon

```
Require: $\mathbf{y}_0, T, T', n, f, h, g, D$, Path=()
Ensure: Path (optimal and safe)
  for $n$ steps do
    $\mathbf{u}_{\text{optim}} \longleftarrow$ SolveP$(\mathbf{y}_0, T, f, h, g, D)$
    $(R(t), \tilde{R}(t)) \longleftarrow$ Predict$(f, \mathbf{y}_0, \mathbf{u}_{\text{optim}}, T')$
    $\mathbf{y}_0 \longleftarrow R(T')$
    Concatenate (Path, $\tilde{R}(t)$)
  end for
```

Main Algorithm In Algorithm 1, n is the total iteration number of the algorithm and $\mathbf{y}_0$ is the initial state for an iteration. h and g are the constraint function and the cost function, respectively. T' and T are the same than given in Section 3.2 and D is the set of inputs which is assumed to be finite. The function SolveP solves our problem (P) (and is given in Algorithm 2). The Predict method returns a list of boxes $R(t)$ and $\tilde{R}(t)$ that contains the solution of $\dot{\mathbf{y}} = f(\mathbf{y},\mathbf{u})$ for a given $\mathbf{u}$ in a guaranteed way using validated simulation method (see Section 2.3). Path is the list of trajectories from the initial state driven by the inputs found for every iteration of the algorithm, this is then the solution of our problem of path planning (considering the dynamics).

Optimization Algorithm In order to find an input which is solution of (P), a discretization of the set of the acceptable inputs is performed, and the one which minimizes g while satisfying the constraint h is returned. In Algorithm 2, D is a finite set of inputs, $\mathbf{u}_{\text{temp}}$ is the previous validated input, and c is the cost corresponding to $\mathbf{u}_{\text{temp}}$. The result is given in term of the "optimal input" $\mathbf{u}_{\text{optim}}$, which is proven to drive the system to the optimal and safe trajectory, while being easier to save in memory than the complete path.

Algorithm 2 SolveP function

```
Require: $\mathbf{y}_0, T, f, h, g, D$
Ensure: Optimal input $\mathbf{u}_{\text{optim}}$
  $c \longleftarrow +\infty$
  for $\mathbf{u} \in D$ do
    $(R(t), \tilde{R}(t)) \longleftarrow$ Predict$(f, \mathbf{y}_0, \mathbf{u}, T)$
    if $g(R(T)) < c$ and $h(\tilde{R}([0,T])) < 0$ then
      $\mathbf{u}_{\text{temp}} \longleftarrow \mathbf{u}$
      $c \longleftarrow g(R(T))$
    end if
  end for
  $\mathbf{u}_{\text{optim}} \longleftarrow \mathbf{u}_{\text{temp}}$
```

Algorithm 3 Improvement of Algorithm 2: SolveP function

```
Require: $\mathbf{y}_0, T, f, h, g, D$
Ensure: Optimal input $\mathbf{u}_{\text{optim}}$
  $c \longleftarrow +\infty$
  $D' \longleftarrow D$
  for $\mathbf{u} \in D'$ do
    $(R(t), \tilde{R}(t)) \longleftarrow$ Predict($f, \mathbf{y}_0, \mathbf{u}, T$)
    if $g(R(T)) < c$ and $h(\tilde{R}([0,T])) < 0$ then
      $\mathbf{u}_{\text{temp}} \longleftarrow \mathbf{u}$
      $c \longleftarrow g(R(T))$
    end if
    $D' \longleftarrow D' \setminus \{\mathbf{u}\}$
    $(R_{D'}(t), \tilde{R}_{D'}(t)) \longleftarrow$ Predict($f, \mathbf{y}_0, D', T$)
    if $c \leqslant g(R_{D'}(T))$ then
      break
    end if
  end for
  $\mathbf{u}_{\text{optim}} \longleftarrow \mathbf{u}_{\text{temp}}$
```

Improvement of Algorithm 2 An improvement of the previous algorithm using the set-membership approach is proposed.

Remark 3. The proposed improvement makes the algorithm more complex. We provide the two versions for clarity.

For each input $\mathbf{u}$ found, considering that D' is the set of the remaining untested inputs, a set-based simulation on all the set D' can be performed by the help of validated numerical integration methods (see Section 2.3). We denote by $(R_{D'}(t), \tilde{R}_{D'}(t))$ the obtained over-approximated trajectories. If $c \leqslant g(R_{D'}(T))$, we ensured that the previous validated input $\mathbf{u}$ is optimal and no more iteration over D is needed.

3.4 Discussion

In this section, complexity and soundness of the approach are discussed.

In Algorithm 1, the number n of iterations depends on the complete path duration τ such as $n = \lceil \frac{\tau - T}{T'} + 1 \rceil$ with T the prediction period and T' the sliding period with $T' < T < \tau$.

Naive `SolveP` algorithm (see Algorithm 2) has a linear complexity in function of the discretization of the input u. If an input u is found, it is safe, *i.e.*, the solution of the dynamical system respects all the constraints, and it is optimal with respect to the discretization. In the improved version, see Algorithm 3, if a solution is found, it is the same than the one found by Algorithm 2 so it is safe and optimal. On contrary, although the complexity is still linear in the inputs, the `Predict` function is called twice, except that the second call of `Predict` function can quickly prune the search space of the inputs.

`Predict` function is based on the guaranteed numerical integration methods as presented in Section 2.3. In consequence, the complexity depends on the dimension of the dynamical system and the order of the integration methods Nedialkov et al. (1999).

4. MOTION PLANNING FOR AN AUV

The motion planning of an AUV which has to move the closest to the seabed is considered. In consequence, the cost function is the depth of the gravity center of the AUV. As security constraints, we want to ensure that the AUV is closer to the seabed than the distance d_{max} and further than d_{min}. It is the procedure for the AUV to avoid the seabed, which is the main obstacle in this example. Other obstacles can be added without any differences in the algorithm as we consider that everything is motionless during the sliding period. If something crosses the trajectory, the avoidance will be handled by a decision procedure or the controller, but not by the motion planner.

4.1 Dynamics of an AUV

The gravity center of the AUV is subjected to the ODE defined in Jaulin (2015) and given in Equation (4).

$$\begin{cases} \dot{x} = v\cos\theta\cos\psi \\ \dot{y} = v\cos\theta\sin\psi \\ \dot{z} = -v\sin\theta \\ \dot{\psi} = \dfrac{\sin\varphi}{\cos\theta}\cdot v\cdot u_1 + \dfrac{\cos\varphi}{\cos\theta}\cdot v\cdot u_2 \\ \dot{\theta} = \cos\varphi\cdot v\cdot u_1 - \sin\varphi\cdot v\cdot u_2 \\ \dot{\varphi} = -0.1\sin\varphi + \theta\cdot v\cdot(\sin\varphi\cdot u_1 + \cos\varphi\cdot u_2) \end{cases} \tag{4}$$

where $\mathbf{s} = (x, y, z, \psi, \theta, \varphi)$ is the state vector which can be split into the vector (x, y, z) of the coordinates of the gravity center and the vector (ψ, θ, φ) of the Euler angles; $\mathbf{u} = (u_1, u_2)$ is the input vector; v is the velocity.

Note that (4) has been simplified by substituting $\tan\theta$ by θ in the definition of $\dot{\varphi}$ to avoid technical issues of the implementation. Nonetheless, the algorithm remains valid.

4.2 Underwater environment

We define a function $(x, y) \mapsto \text{seabed}(x, y)$ which returns the depth of the seabed at the coordinates (x, y). We also define d_{min} and d_{max} two constants such that the AUV stays at a distance to the seabed between d_{min} and d_{max}. Some constraints on AUV angles are considered: yaw and roll are bounded in an interval (to go in a quite straight way and to not capsize) and pitch is bounded by an extreme value (to limit the dive angle). Finally, in order to force the AUV to move forward through the x dimension, we impose $x_{\text{end}} > x_{\text{init}}$. Thus, the problem is to find the input $\mathbf{u}$ solution of

$$(P_{\text{AUV}}): \begin{cases} \min\limits_{\mathbf{u}} \quad z \\ \dot{\mathbf{s}} = f(\mathbf{s}, \mathbf{u}) \\ z > \text{seabed}(x, y) + d_{\text{min}} \\ z < \text{seabed}(x, y) + d_{\text{max}} \\ x_{\text{end}} > x_{\text{init}} \\ \theta < 0.8 \\ \varphi, \psi \in [-0.5, 0.5] \end{cases}$$

It is important to keep in mind that the seabed is globally unknown by the system. It is discovered during the motion (by the help of sensors), with a window corresponding to the sliding horizon. It is then impossible to know the seabed after the prediction period.

4.3 Experiments

Settings of the experiments The **seabed function** is defined by

$$\begin{aligned} \text{seabed}: (x, y) \mapsto\ & 3(1-x)^2 e^{-x^2-(y+1)^2} \\ & -10\left(\frac{x}{5} - x^3 - y^5\right) e^{-x^2-y^2} \\ & -\frac{e^{-(x+1)^2-y^2}}{3} \end{aligned}$$

Then the following seabeds are considered

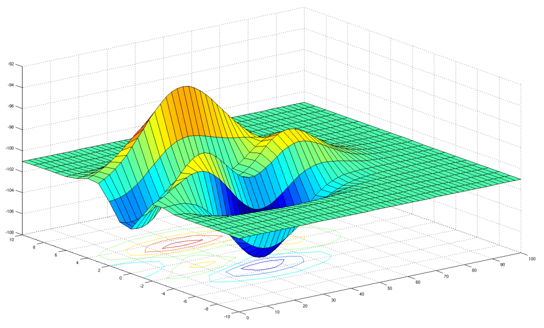

Fig. 1. Seabed picture.

Scenario	Original algorithm	Improved algorithm	Gain
Seabed 1	3535	2665	25%
Seabed 2	3535	3235	9%
Seabed 3	3535	2685	24%
Seabed 4	3535	2355	34%

Table 1. Number of predictions for the four scenarios with the Algorithm 2 and its improved version Algorithm 3.

- seabed 1: $(x,y) \mapsto \text{seabed}\left(\frac{x-30}{20}, \frac{y}{2}\right) - 100$;
- seabed 2: $(x,y) \mapsto \text{seabed}\left(\frac{y}{2}, \frac{x-30}{20}\right) - 100$;
- seabed 3: $(x,y) \mapsto \text{seabed}\left(\frac{-x+30}{20}, \frac{y}{2}\right) - 100$;
- seabed 4: $(x,y) \mapsto \text{seabed}\left(\frac{y}{2}, \frac{-x+30}{20}, \frac{y}{2}\right) - 100$.

Each seabed is a rotation of the first one (see Figure 1). Seabeds are defined by a function for experiments, but the motion planner has access only on the sliding window.

The **limits on depth** are defined such that $d_{\min} = 1$ meter $d_{\max} = 10$ meters. The **velocity** v is fixed to 0.1 m.s^{-1} and the **initial state** $\mathbf{s}_0$ to $(0,0,-92,0.1,0.1,0.1)$. As **discretization** of the set of the inputs, we define

$$D = \left\{ -0.3 + \frac{0.6k}{9} : k \in [\![0,9]\!] \right\}^2 .$$

Finally, the prediction and sliding periods are $T = 30$s and $T' = 15$s, for a total number of steps $n = 35$.

Implementation of the SolveP function The presented method has been implemented in the DynIBEX framework [1]. It proposes a validated simulation procedure based on Runge-Kutta methods and provides some differential constraint programming facilities Alexandre dit Sandretto et al. (2017). Every **simulation** has been computed using Heun's method with a 10^{-4} precision which is a good trade-off between speed and precision of the results.

With the parameters described in Section 4.3.1, solving the motion planning problem requires less of calling to the prediction method (*i.e.*, ODE solver) with the improved Algorithm 3 than with the Algorithm 2, as shown in Table 1.

4.4 Discussion

Results of the presented method are given in term of depth w.r.t. time in Figure 2. The depth is computed with seabed(x,y). The path provided by the motion planning method for the seabeds 1, 3 and 4 allows the AUV to follow the seabed remaining between a distance $d_{\min}$ and $d_{\max}$ from it. As shown in Figure 2 top right, the execution for the second seabed failed after 9 steps, because of the variations of the seabed (a deep ditch and then a climb). In this picture, after $t = 135$ seconds, a simulation is done with the entire interval of inputs $u = [-0.3, 0.3]^2$, and no solution can be found without collision with the seabed. This phenomenon is due to the fact that the dive velocity of the AUV is high at the moment when the seabed grows quickly.

A solution to this failure of the algorithm could be obtained by increasing T and decreasing T' in order to see the obstacle sooner but with a longer computation time (see Figure 3). An other solution is to add a constraint to the problem, for example we bounded $\text{lb}\,(\dot{z}) \geqslant -0.03$ m.s^{-1} and the algorithm achieved to find a path across the seabed (see Figure 4). Indeed, it succeeds by avoiding the gap and reaching the optimal depth further.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed a validated algorithm to solve a motion planning problem, considering constraints and cost on states. Our approach showed its efficiency in an application of motion planning for an Autonomous Underwater Vehicle. An improvement exploiting set membership is used to avoid some tests on inputs which reduces significantly the number of simulations. The presented approach can in some cases failed because of its dependency to many parameters that the user has to calibrate.

As a future work, a procedure to dynamically choose the parameters such as prediction period or additional constraints on system behavior could be considered, w.r.t. a prediction of the seabed for example.

REFERENCES

Alexandre dit Sandretto, J. and Chapoutot, A. (2016). Validated Explicit and Implicit Runge-Kutta Methods. *Reliable Computing*, 22.

Alexandre dit Sandretto, J., Chapoutot, A., and Mullier, O. (2017). Formal verification of robotic behaviors in presence of bounded uncertainties. In *Proc. of IEEE International Conference on Robotic Computing*.

Bemporad, A., Luca, A.D., and Oriolo, G. (1996). Local incremental planning for a car-like robot navigating among obstacles. In *Proc. of IEEE International Conference on Robotics and Automation*, 1205–1211.

Censi, A., Calisi, D., Luca, A.D., and Oriolo, G. (2008). A bayesian framework for optimal motion planning with uncertainty. In *Proc. of IEEE International Conference on Robotics and Automation*.

Hansen, E.R. (2003). *Global Optimization Using Interval Analysis*. Marcel Dekker Inc.

Jaulin, L. (2001). Path planning using intervals and graphs. *Reliable Computing*, 7(1), 1–15.

Jaulin, L. (2015). *Mobile Robotics*. ISTE Press - Elsevier.

Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. (2001). *Applied Interval Analysis*. Springer.

Karaman, S. and Frazzoli, E. (2010). Optimal kinodynamic motion planning using incremental sampling-based methods. In *Proc. of IEEE Conference on Decision and Control*.

Lambert, A. and Gruyer, D. (2003). Safe path planning in an uncertain-configuration space. In *Proc. of IEEE International Conference on Robotics and Automation*.

[1] `http://perso.ensta-paristech.fr/~chapoutot/dynibex/`

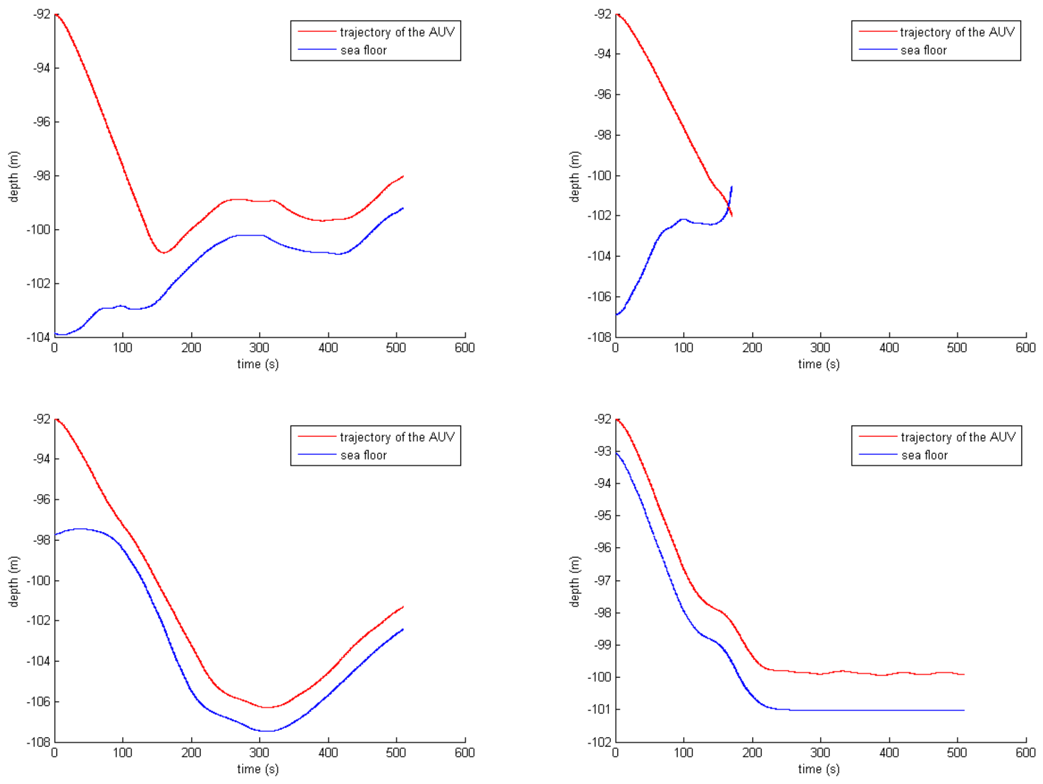

Fig. 2. Results of the AUV motion planning with seabed 1 to 4 from top left to bottom right.

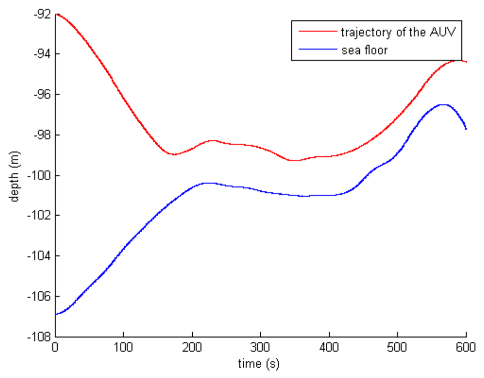

Fig. 3. Result for the seabed 2 with longer prediction period.

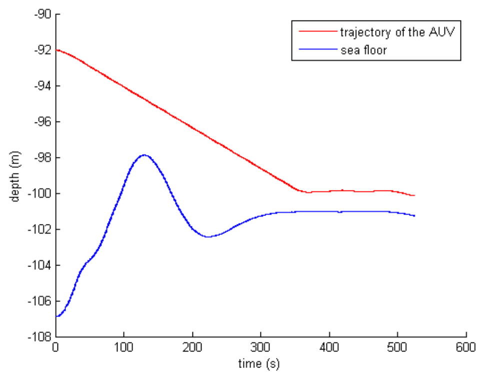

Fig. 4. Result for the seabed 2 with additional constraint on diving speed.

LaValle, S.M. and Kuffner, J.J. (2000). Rapidly-exploring random trees: Progress and prospects. In *Proc. of Workshop on the Algorithmic Foundations of Robotics*.

Leonard, J.J. and Durrant-Whyte, H.F. (1991). Simultaneous map building and localization for an autonomous mobile robot. In *Proc. of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, volume 3, 1442–1447.

Mendes Filho, J.M. and Lucet, E. (2016). Multi-robot motion planning: a modified receding horizon approach for reaching goal states. *Acta Polytechnica*, 56(1), 10–17.

Moore, R.E. (1966). *Interval Analysis*. Prentice Hall.

Nedialkov, N.S., Jackson, K., and Corliss, G. (1999). Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1), 21 – 68.

Page, L.A. and Sanderson, A.C. (1995). Robot motion planning for sensor-based control with uncertainties. In *Proc. of IEEE International Conferenc on Robotics and Automation*, volume 2, 1333–1340.

Pepy, R., Kieffer, M., and Walter, E. (2009). Reliable robust path planning with application to mobile robots. *International Journal of Applied Mathematics and Computer Science*, 19(3), 413–424.

Rueher, M. (2005). Solving continuous constraint systems. In *International Conference on Computer Graphics and Artificial Intelligence*.

Wongpiromsarn, T., Topcu, U., and Murray, R.M. (2012). Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11), 2817–2830.