# Heuristic Guidance Techniques for the Exploration of Small Celestial Bodies

Francesco Capolupo, Thierry Simeon, Jean-Claude Berges

# Heuristic Guidance Techniques for the Exploration of Small Celestial Bodies

**Francesco Capolupo** [*] **Thierry Simeon** [**]
**Jean-Claude Berges** [***]

[*] *Airbus Defence and Space, 31400 Toulouse, France (e-mail: francesco.capolupo@airbus.com)*
[**] *LAAS–CNRS, Université de Toulouse, CNRS, 31400 Toulouse, France (e-mail: nic@laas.fr)*
[***] *Centre National d'Études Spatiales, 31400 Toulouse, France (e-mail: jean-claude.berges@cnes.fr)*

**Abstract:** Sampling Based Motion Planning (SBMP) techniques are widely used in robotics to plan feasible trajectories of a vehicle/robot evolving in a complex and constrained environment. Algorithms such as Rapidly Exploring Random Trees (RRT) and Sampling Based Model Predictive Optimization (SBMPO) allow for an efficient exploration of the state space, and the construction of a feasible sequence of maneuvers and trajectories that respect the kynodynamic and path constraints of the system. Proximity operations around small bodies are characterized by complex dynamics and constraints that can be easily and autonomously handled by motion planning techniques. This paper presents two motion planning algorithms designed to solve two different guidance problems: the landing on a small body and its observation. The mission scenarios considered to test the algorithms are the landing of Rosetta on the comet 67P/Churyumov-Gerasimenko and the observation of Didymain in the Didymos binary asteroid system. To conclude, the applicability of SBMP techniques to small body proximity operations are discussed. In particular, the advantages of implementing SBMP algorithms to solve complex high-level planning problems or to guide a spacecraft in a cluttered environment are highlighted.

*Keywords:* Guidance Navigation and Control of Vehicles, Space Exploration and Transportation, Autonomous Systems, Trajectory and Path Planning, Mission Planning and Decision Making

## 1. INTRODUCTION

Motion planning constitutes a very active research domain within the robotics community. Sampling Based Motion Planning (SBMP) algorithms were introduced in the '90 (Kavraki et al. (1996)) to overcome the computational complexity of the motion planning problem. The idea behind SBMP is to replace the explicit representation of the configuration space occupied by obstacles by a random sampling of the obstacle-free space, and the construction of admissible paths between samples.

The use of Sampling Based Motion Planning techniques for small bodies proximity operations was recently proposed by Pavone et al. (2014) as an effective way to deal with these challenging mission phases. Future space exploration missions will require an unprecedented level of autonomy. This need is driven by the communication delays between the ground and the spacecraft, as well as by the dynamical complexity of the explored environments. Future guidance and control solutions will also have to deal with stringent collision avoidance constraints, that considerably complicate the task of trajectory design and GNC engineers. This paper shows how complex guidance problems, such as the landing on a small body and its complete observation, can be efficiently solved with SBMP algorithms. In particular, two reference mission scenarios were considered to benchmark motion sampling techniques: the landing of Rosetta on the comet 67P/Churyumov-Gerasimenko and the observation of Didymain in the Didymos binary asteroid system. These two examples represent two very different guidance problems.

The case of Rosetta is an example of a typical constrained optimal control problem. The complex shape of the comet causes the resulting parameter optimization problem to be a hard to solve non-convex and non-smooth optimization problem. It will be shown how SBMP algorithms can solve this type of transfer problems without any difficulty.

The Didymos scenario is an example of "high-level" mission where the spacecraft is supposed to autonomously plan its trajectory in order to complete a scientific goal. No predefined waypoints are specified by mission analysts, so that the algorithm must be able to find the optimal path that fulfills a high-level task such as the observation of the entire surface of an asteroid. As it will be discussed in the following sections, SBMP algorithms are able to handle high-level objectives, and therefore can be successfully used for this type of autonomous trajectory planning problems.

⋆ This work has been co–funded by the Centre National d'Études Spatiales (the French Space Agency) under contract n°151180.

## 2. SMALL BODY LANDING

### *2.1 The Guidance Problem*

The guidance problem of the small body landing scenario (i.e. Rosetta landing on the 67P/Churyumov-Gerasimenko comet, in our case) consists in bringing a lander from an initial state to a desired landing site, while minimizing the propellant consumption. The simplified translational dynamics of a spacecraft in the vicinity of the comet, written in the comet body fixed reference frame is given by

$$\ddot{\mathbf{r}} = -2\boldsymbol{\omega} \times \dot{\mathbf{r}} - \boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{r} + \mathbf{g}_{67P/V} + \mathbf{u} \tag{1}$$

where $\mathbf{r}$ is the spacecraft position vector with respect to the comet's center of mass, $\boldsymbol{\omega}$ is the angular rotation vector of the comet (supposed constant), $\mathbf{g}_{67P/V}$ is its gravitational attraction on the vehicle, and $\mathbf{u}$ is the control acceleration vector. The nonlinear dynamical system represented by Equation 1 can be written in a more general form as a non-linear first order autonomous system

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$$

Mathematically, the landing problem can be translated into a constrained trajectory optimization problem (or optimal control problem)

$$\begin{aligned} \underset{\mathbf{u}(t),\mathbf{x}(t),t_f}{\text{minimize}} \quad & J = \int_{t_0}^{t_f} \|\mathbf{u}\| \, dt \\ \text{subject to} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \\ & \mathbf{x}(t_0) = \mathbf{x}_0 \\ & \mathbf{x}(t_f) = \mathbf{x}_f \\ & t_0 < t_f \leq t_{f,\max} \\ & \mathbf{x} \in X_{free} \\ & \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} \end{aligned} \tag{2}$$

where collision avoidance constraints are taken into account by defining a collision free state subspace $X_{free}$. The collision free subspace is mainly determined by the shape of the small body, and the presence of regions to be avoided, such as out-gassing cones. An irregularly shaped body like 67P leads to a non-convex domain $X_{free}$. Control authority limits are also taken into account, with the introduction of upper and lower control bounds, $\mathbf{u}_{\max}$ and $\mathbf{u}_{\min}$. There exist several numerical methods to solve the optimal control problem of Equation 2 (e.g. direct and indirect shooting and collocation methods). Nevertheless, the translation of the non-convex obstacle avoidance constraint into a nonlinear non-smooth function reduces the robustness of classic optimization methods and significantly increases the computational time. In addition, classic methods require an initial guess for both state and control profiles, and can only converge to a local optimal solution in the vicinity of the initial guess. To overcome these limitations, a new landing guidance algorithm is proposed. The new algorithm, described in detail in the following section, is based on motion planning techniques that are commonly used in robotics.

### *2.2 The Algorithm*

The optimal Rapidly Exploring Random Tree algorithm (RRT*) was chosen to solve the landing guidance problem. RRT* was introduced by Karaman and Frazzoli (2011) to optimally solve motion planning problems in robotics. RRT* is a sampling-based motion planning algorithm designed to efficiently search non-convex, high-dimensional spaces by randomly building a space-filling tree. The tree consists of a set of vertices $V$ (states) and edges $E$ (trajectories connecting states), and is constructed incrementally from samples drawn randomly in the state space. The tree is rooted at the initial state and the exploration is performed until the goal is reached. Trajectories connecting state samples are computed by a local unconstrained optimization algorithm called the *steering method*.

As well explained by Karaman and Frazzoli (2011), the first step of the algorithm is to randomly sample a state vector $x_{\texttt{rand}}$ (i.e. position and velocity) from the open subspace $X_{free}$. The `sampleOpenSpace` function is designed to return the target state instead of a random one in a certain number of cases, as specified by the user (typically 1 to 10% of cases). The `nearest` function is then called to provide the closest node $x_{\texttt{nearest}}$ in $V$ to $x_{\texttt{rand}}$. Next, the steering method is used to find a trajectory $\Gamma_{\texttt{new}}$ connecting $x_{\texttt{nearest}}$ to $x_{\texttt{rand}}$. As the steering method might not be able to exactly reach $x_{\texttt{rand}}$, a new node $x_{\texttt{new}}$ close to $x_{\texttt{rand}}$ is obtained (in our case, $x_{\texttt{new}} = x_{\texttt{rand}}$). If $\Gamma_{\texttt{new}}$ respects all the constraints of the problem, then a set of near (within a radius $\gamma$) neighbors $V_{\texttt{near}}$ are evaluated using the `near` function. Next RRT* calls `chooseParent` to find a candidate for a parent node to $x_{\texttt{new}}$. The function `chooseParent` returns the node in the set $V_{\texttt{near}}$ that reaches $x_{\texttt{new}}$ with minimum cost and respecting all the constraints, and it adds it to the search tree. The algorithm then tries to "rewire" the nodes in $V_{\texttt{near}}$ by calling the `rewire` function. If the feasible path that connects $x_{\texttt{new}}$ to the near node $x_{\texttt{near}}$ reaches $x_{\texttt{near}}$ with cost less than that of its current parent, then $x_{\texttt{near}}$ is rewired to $x_{\texttt{new}}$ by connecting $x_{\texttt{rand}}$ and $x_{\texttt{near}}$.

**Algorithm 1** RRT*

```
V ← {x_init}
E ← ∅
x_sol ← ∅
for i = 1, ..., n do
    x_rand ← sampleOpenSpace(V_target)
    x_nearest ← nearest((V, E), x_rand)
    x_new ← steer(x_nearest, x_rand, V_target)
    if constraintsRespected(Γ_new) then
        V_near ← near((V, E), x_new, γ)
        (V, E) ← chooseParent(x_new, V_near, V_target)
        (V, E) ← rewire(x_new, (V, E), V_near, V_target)
        x_sol ← checkTargetReached(x_new, x_sol, V_target)
    end if
end for
return x_sol
```

The algorithm can be easily adapted to kinodynamic motion planning problems, i.e. problems having differential constraints such as $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, provided that an appropriate steering method can be designed (Karaman and Frazzoli (2011)). As the steering method is repeatedly called during the execution of the algorithm, it must be fast enough to allow for reasonable execution times. In order to guarantee the optimality of the solution, the steering method must connect two arbitrary states by a local optimal trajectory. Unfortunately, no analytic optimal solution exists to connect two states of the system

**Algorithm 2** RRT*: `chooseParent`

$\Gamma_{\text{newBest}} = \Gamma_{\text{new}}$
**for** $i = 1, \ldots, n_{\text{near}}$ **do**
  $x_{\text{near}} \leftarrow V_{\text{near},i}$
  $x_{\text{newTest}} \leftarrow \texttt{steer}\left(x_{\text{near}}, x_{\text{new}}, V_{\text{target}}\right)$
  **if** $x_{\text{newTest}} == x_{\text{new}}$ **then**
    **if** $\texttt{costToCome}\left(x_{\text{newTest}}\right) < \texttt{costToCome}\left(x_{\text{new}}\right)$ **then**
      **if** $\texttt{constraintsRespected}\left(\Gamma_{\text{newTest}}\right)$ **then**
        $\Gamma_{\text{newBest}} = \Gamma_{\text{newTest}}$
      **end if**
    **end if**
  **end if**
**end for**
$V \leftarrow V \cup \{x_{\text{new}}\}$
$E \leftarrow E \cup \{\Gamma_{\text{newBest}}\}$
**return** $(V, E)$

**Algorithm 3** RRT*: `rewire`

**for** $i = 1, \ldots, n_{\text{near}}$ **do**
  $x_{\text{near}} \leftarrow V_{\text{near},i}$
  $x_{\text{rw}} \leftarrow \texttt{steer}\left(x_{\text{new}}, x_{\text{near}}, V_{\text{target}}\right)$
  **if** $x_{\text{rw}} == x_{\text{near}}$ **then**
    **if** $\texttt{costToCome}\left(x_{\text{rw}}\right) < \texttt{costToCome}\left(x_{\text{near}}\right)$ **then**
      **if** $\texttt{constraintsRespected}\left(\Gamma_{\text{rw}}\right)$ **then**
        $E \leftarrow E \backslash \{\Gamma_{\text{near}}\}$
        $E \leftarrow E \cup \{\Gamma_{\text{rw}}\}$
      **end if**
    **end if**
  **end if**
**end for**
**return** $(V, E)$

described by the Equation 1, not even for the unconstrained case. Nevertheless, a good (and extremely fast) approximation of the local optimal trajectory can be found with a polynomial guidance algorithm.

The polynomial guidance algorithm, proposed by Ploen et al. (2006) for a Mars landing scenario and direct descendant of the Apollo guidance law, analytically solves the fixed horizon two-point boundary value problem of a double integrator

$$\ddot{\mathbf{r}} = \mathbf{a}$$

by supposing that the total acceleration profile of the vehicle $\mathbf{a}(t)$ can be approximated by a polynomial function of the $N$-th order

$$\begin{aligned}
\mathbf{a}(t) &= \mathbf{c}_0 + \mathbf{c}_1 t + \mathbf{c}_2 t^2 + \ldots + \mathbf{c}_N t^N = \Phi_a(t)\, C \\
\dot{\mathbf{r}}(t) &= \dot{\mathbf{r}}_0 + \Phi_v(t)\, C \\
\mathbf{r}(t) &= \mathbf{r}_0 + \dot{\mathbf{r}}_0 t + \Phi_p(t)\, C
\end{aligned}$$

where

$$\Phi_v(t) = \int_0^t \Phi_a(t)\, dt \qquad \Phi_p(t) = \int_0^t \Phi_v(t)\, dt$$

The boundary conditions in terms of position, velocity and total acceleration, are then translated into an over-determined linear equality constraint for $C$

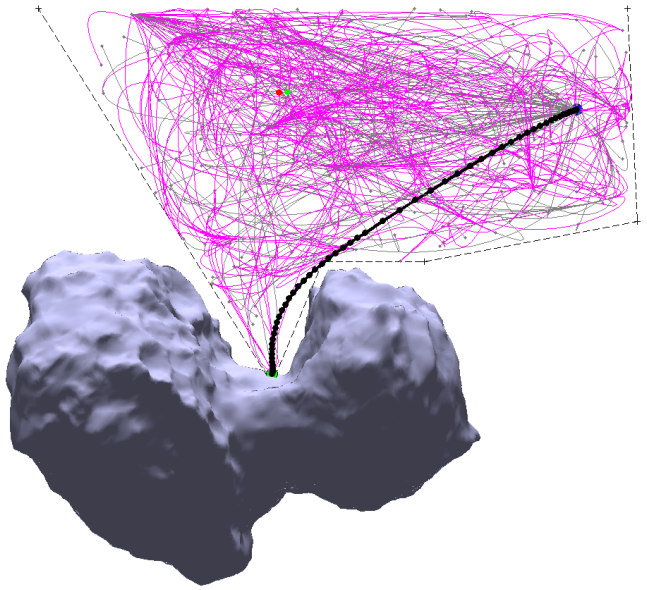

Fig. 1. Descent trajectory on 67P.

$$\begin{bmatrix} \Phi_a(0) \\ \Phi_a(t_f) \\ \Phi_v(t_f) \\ \Phi_p(t_f) \end{bmatrix} \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_N \end{bmatrix} = \begin{bmatrix} \ddot{\mathbf{r}}_0 \\ \ddot{\mathbf{r}}_f \\ \dot{\mathbf{r}}_f - \dot{\mathbf{r}}_0 \\ \mathbf{r}_f - \mathbf{r}_0 - \dot{\mathbf{r}}_0 t_f \end{bmatrix} \Rightarrow AC = b \quad (3)$$

The remaining degrees of freedom in $C$ can be chosen to minimize the total acceleration energy

$$J_a = \int_0^{t_f} \mathbf{a}^T \mathbf{a} dt = C^T \int_0^{t_f} \Phi_a^T \Phi_a dt C = C^T S C \quad (4)$$

The problem given by Equations 3 and 4 is a linearly constrained weighted least norm problem for $C$, whose solution is given by

$$C = S^{-1} A^T \left(A S^{-1} A^T\right)^{-1} b$$

Once the total acceleration and the optimal trajectory have been computed, the control acceleration profile $\mathbf{u}(t)$ can be found as

$$\mathbf{u}(t) = \mathbf{a}(t) + 2\boldsymbol{\omega} \times \dot{\mathbf{r}} + \boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{r} - \nabla U_{67P}(\mathbf{r})$$

The optimal transfer time $t_f$ can be determined by a line search optimization that minimizes the $J$ functional as defined in Equation 2 (optimal, but slower solution), or drawn randomly by the steering function (non optimal, fastest solution).

### *2.3 Simulation Results*

The RRT* landing algorithm was tested on a Rosetta-like landing scenario. The complex shape of 67P allows for a challenging benchmark of the proposed guidance law. The landing site was chosen to be between the two comet's lobes, in a hardly accessible region of the surface. Therefore, collision avoidance constraints drive the design of a feasible descent trajectory.

In simulations, the initial spacecraft state was chosen equal to $\mathbf{r}_0 = (-3.5, 0, 3)$ km and $\dot{\mathbf{r}}_0 = (0, 0, 0)$ m/s, and the landing site was chosen equal to $\mathbf{r}_f = (-0.5, 0, 0.4)$ km and $\dot{\mathbf{r}}_f = (0, 0, -0.1)$ m/s. The control force was limited to 10 N, and the vehicle mass was taken equal to 1500 kg.

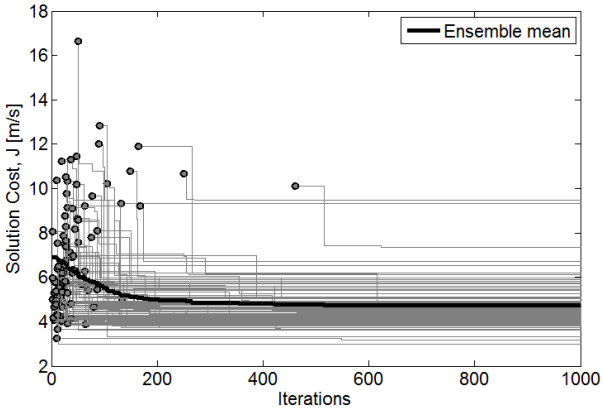


Fig. 2. Solution cost versus number of iterations.

Figure 1 shows an example of result provided by the algorithm. The solid black trajectory represents the optimal trajectory found within the prescribed number of iterations, the solid grey trajectories are the edges between sampled states, while rewired edges are displayed in magenta. The dotted contour represents the boundary of the search space: the definition of such a boundary allows for the translation of the complex shape of the comet into an arbitrarily complex polyhedron; it also limits the computational burden by limiting the state sampling domain. In this figure, we recognize the behavior of RRT* search: the algorithm tries to connect the randomly sampled states of the search space, until it finds a feasible trajectory that respects all the imposed constraints. The continuous search for optimal connections between nodes allows refining the tree during the iterations, so that the algorithm can converge towards an optimal solution. Figure 2 shows the cost of the solution found by the algorithm as a function of the number of iterations, for a test campaign of 100 simulations. Each run is represented by a grey line. The grey dots at the beginning of each line represent the cost of the first feasible trajectory found. The solid black line is the mean over the ensemble of 100 simulations. The maximum solution cost (i.e. the first solution cost) was considered for the computation of the ensemble mean for simulations not having found a feasible solution yet.

It is worth noticing that a very limited number of iterations are necessary to compute a first feasible trajectory. The majority of the starting points are found in less than 100 iterations. Moreover, the solution cost quickly drops and reaches "reasonable values" (i.e. around 4 m/s) in about 200 iterations. On a common PC workstation (Intel Core i3 @3.30 Ghz, 8Gb of RAM) the time needed to perform 100 iterations is less than 10 seconds. This means that feasible solutions are found within 10 seconds from the beginning of the run, and "reasonably optimal" solutions are provided in less than 20 seconds.

Direct optimization techniques (direct collocation and multiple shooting) were also used to solve the same problem. The results were first used to validate SBMP algorithms, and then compared to motion planning solutions. The Matlab `fmincon` solver was used to solve the nonlinear programming problem (NLP) obtained from the transcription of the optimal control problem of Equation 2. Despite the local optimality of classic direct optimization techniques, the best solution cost obtained with direct optimization is of 3.10 m/s, and is therefore comparable to the lowest cost of 2.98 m/s obtained by the RRT* algorithm. Nevertheless, the mean computational time for direct optimization to solve the problem, starting from a linear state and null control initial guess, is of about 5 minutes. In addition, classic optimization is very sensitive to the choice of transcription parameters: an incautious transcription parametrization might severely affect the possibility for the NLP solver to converge towards a local optimal solution.

On the other hand, RRT* appears to be very robust, as it can quickly provide a feasible solution without the need of an initial guess. Moreover, the list of parameters directly linked to the motion planning algorithm is limited to a neighboring radius $\gamma$ and a target aim percentage. Motion planning algorithms are not ready yet to be run on-board a spacecraft processor, mainly because of their computational load. However, their robustness and rapidity, if compared to classic trajectory optimization techniques, as well as their ability to autonomously handle almost any kind of constraint, represent some very interesting features that upcoming autonomous exploration missions might soon require.

## 3. SMALL BODY OBSERVATION

### *3.1 The Guidance Problem*

This second mission scenario is directly inspired by the NASA/ESA Asteroid Impact and Deflection Assessment (AIDA) mission. The spacecraft has to observe and map the complete surface of Didymain, the largest body of the Didymos binary asteroid system. The observation trajectory shall ensure a satisfactory illumination of the observed scene, and must avoid any collision with Didymain and its small moon, Didymoon.

The spacecraft dynamics in the Didymain Local Vertical/Local Horizontal (LVLH) reference frame is given by

$$\begin{aligned}\ddot{\mathbf{r}} = &- \dot{\boldsymbol{\omega}} \times \mathbf{r} - 2\boldsymbol{\omega} \times \dot{\mathbf{r}} - \boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{r} + \dots \\ &+ \mathbf{g}_{S/V} + \mathbf{g}_{D/V} + \mathbf{g}_{M/V} - \mathbf{g}_{M/D} - \mathbf{g}_{S/D} + \mathbf{u}\end{aligned} \tag{5}$$

where $\boldsymbol{\omega}$ is the Didymain orbital angular rate, $\mathbf{g}_{X/Y}$ is the gravitational attraction of $X$ on $Y$, and subscripts are defined as follows: $V$ for the vehicle, $D$ for Didymain, $M$ for Didymoon and $S$ for the Sun. In the LVLH frame, the Sun direction is a constant vector aligned along the $y$ axis. In this scenario, the control input corresponds to a series of impulsive commands, so that at the instant when the command is delivered, $t_i$, the spacecraft velocity is instantaneously modified

$$\begin{aligned}\Delta\mathbf{v}\left(t_i\right) &= \int_{t_i^-}^{t_i^+} \mathbf{u} dt \\ \dot{\mathbf{r}}\left(t_i^+\right) &= \dot{\mathbf{r}}\left(t_i^-\right) + \Delta\mathbf{v}\left(t_i\right) \\ t_i^+ - t_i^- &\to 0\end{aligned}$$

This approximation is valid if the level of thrust available on-board the spacecraft allows for very short maneuvers.

In order to quantify the time spent by the spacecraft in observing a certain region of the asteroid's surface, an observation model was introduced. The model, shown

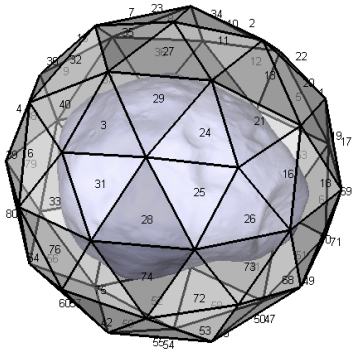

Fig. 3. Surface observation model.

in Figure 3, approximates the surface of the body by an icosahedron with 80 triangular faces. The $i$-th face is observable if the vector connecting the asteroid's center of mass and the spacecraft (relative position vector) passes through it. The $i$-th face is considered to be well illuminated only if the angle between the sun direction and the normal to the face (illumination angle) is less or equal to 45° (illumination cone). This means that each face can be observed if and only if the spacecraft passes over the face and, at the same time, the illumination angle is within the prescribed illumination cone. Because of the direction of the spinning axis of Didymain with respect to the ecliptic, the polar regions (20 faces in total) are never sufficiently well illuminated by the Sun, therefore, they are considered to be unobservable. The observation is considered complete once each observable face ($n_f = 60$ faces in total) has been observed for at least $T_{\text{obs,min}} = 20$ minutes, i.e. when the mission completion percentage $M_c$, defined as

$$M_c = \frac{1}{n_f T_{\text{obs,min}}} \sum_{i=1}^{n_f} \min\left(T_{\text{obs,min}}, T_{\text{obs},i}\right)$$

reaches 100%.

The translation of the observation problem into an optimal control problem is not straightforward. The high level objective of "observing the entire surface of the body" may be either transcribed as an optimization objective or as an isoperimetric constraint. The next section shows how such a high level planning goal can be easily handled by sampling based motion planning techniques.

### *3.2 The Algorithm*

The algorithm proposed to solve the observation problem is a Sampling Based Model Predictive Optimization (SBMPO) algorithm, inspired by the work of Surovik and Scheeres (2014), Komendera et al. (2012) and Dunlap et al. (2008).

The basic idea behind the algorithm is to build, for each command constituting the maneuver sequence, a 3D map that associates an observation score $C$ to each point of the admissible command space $\mathcal{V}$ (i.e. the set of maneuvers that respect $\Delta v = \|\Delta\mathbf{v}\|_2 \leq \Delta v_{\max}$). The score is computed by propagating each sampled command and by analyzing the resulting trajectory $\Gamma(t)$. The observation score is defined by

**Algorithm 4** SBMPO

```
M_c ← ∅
i = 1
while M_c < 100 do
    for j = 1, ..., n_mesh do
        if j == 1 then
            ΔV ← initMesh(Δv_max, n_{0,Δv})
            C ← ∅
        else
            ΔV ← refineMesh(ΔV, C, Δv_max, n_Δv)
        end if
        for k = 1, ..., n_Δv do
            Γ_k ← propagateTrajectory(Δv_k, [t_i; t_{i+1}])
            C_k ← computeScore(Γ_k)
        end for
    end for
    Δv_opt(t_i), M_c ← computeOptimalMan(C, ΔV)
    i++
end while
```

$$C(\Gamma) = \int_{t_i}^{t_{i+1}} \frac{\varepsilon}{|1 - d(t)/d_{\text{des}}| + \varepsilon} dt + w_t \Delta T_{\text{obs}} \qquad (6)$$

where $d(t) = \|\mathbf{r}(t)\|$ is the distance of the spacecraft from the observed body, $d_{\text{des}}$ is a desired distance set by the user, $\Delta T_{\text{obs}}$ is the additional observation time that can be obtained by following the trajectory $\Gamma$, $\varepsilon$ is a desired decay rate, and $w_t$ is a weighting factor. $C$ is set to zero if the trajectory does not respect the imposed path constraints (i.e. collision avoidance or escape). The $\Delta T_{\text{obs}}$ is also set to zero if the observed face has been observed for longer than $T_{\text{obs,max}}$: this allows for a more uniform observation of all faces.

Let's suppose that at the time $t_i$ a new maneuver $\Delta\mathbf{v}(t_i)$ shall be computed. The first step of the algorithm is to call the `initMesh` function to uniformly drawn $n_{0,\Delta v}$ maneuver samples in $\mathcal{V}$. These samples, $\Delta V$, are numerically propagated through the system and the resulting trajectories and scores are computed. Once this first mapping $C(\Delta\mathbf{v})$ is complete, a refinement heuristic is called (`refineMesh` function), in order to bias the next $\Delta\mathbf{v}$ sampling towards the most interesting region of the command space. The observation score, the observation score gradient and the level of exploration of a certain region in $\mathcal{V}$ are the three criteria used to heuristically quantify this interest, as proposed by Surovik and Scheeres (2014). The propagation, analysis, and refinement procedure is repeated (for the same maneuver $\Delta\mathbf{v}(t_i)$) $n_{\text{mesh}}$ times. Next, a planner is called to compute the optimal maneuver among all the maneuvers sampled by the algorithm and to be potentially executed at time $t_i$. The planner takes into account both the observation score $C_q$ of the $q$-th maneuver of the ensemble, and its fuel consumption $\Delta v_q(t_i)$, so that its planning score $P_q$ is given by

$$P_q = \frac{w_c C_q}{\max\left[C_1, C_2, \dots\right]} + \left(1 - \frac{\Delta v_q}{\max\left[\Delta v_1, \Delta v_2, \dots\right]}\right)$$

where $w_c$ is an appropriate weighting factor. The optimal maneuver to be executed by the spacecraft at time $t_i$, $\Delta\mathbf{v}_{\text{opt}}(t_i)$, is then selected as the one with the highest planning score. Once the maneuver is executed, and the spacecraft reaches the next maneuvering point $t_{i+1}$, the entire procedure is repeated for the computation of

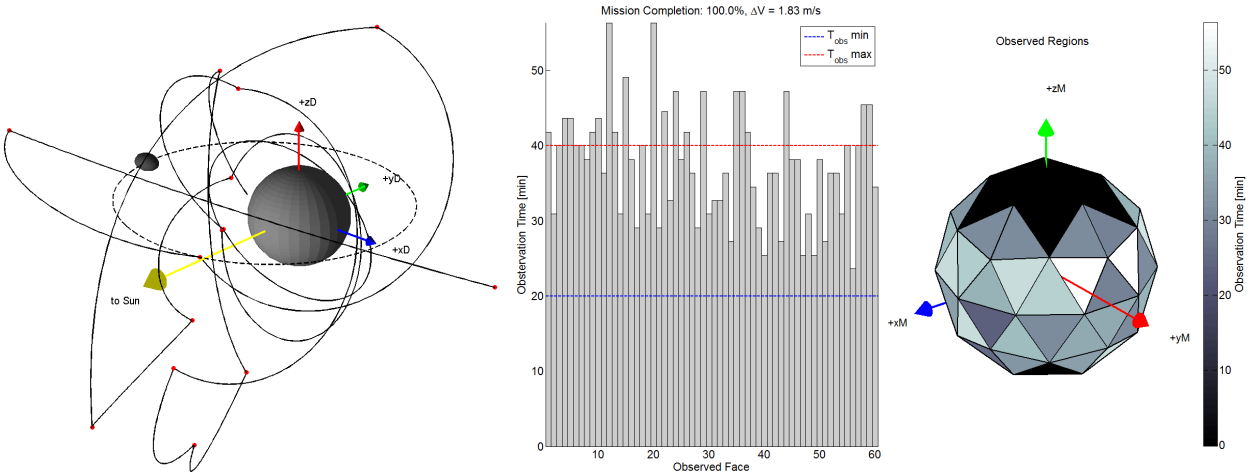


Fig. 4. Example of observation trajectory and performance.

$\Delta\mathbf{v}_{\text{opt}}(t_{i+1})$. New maneuvers are added to the sequence until $M_c = 100\%$.

### *3.3 Simulation Results*

Figure 4 shows an example of solution of the observation problem. The trajectory followed by the spacecraft is represented in the left plot by a solid black line . Red dots interconnecting trajectory arcs correspond to the maneuver points, while the dashed black line represents the orbit of Didymoon around Didymain. It is worth noticing that the majority of the trajectory develops between the Sun and Dydimain, and that the "pericenters" of observation arcs are placed at night. This means that the algorithm is trying to maximize the time spent in good illumination conditions, by passing very rapidly through the shadow cone of the asteroid. The second plot (middle) shows the observation time for each observable face representing the surface. At mission completion, the accumulated observation time for each face is above the minimum required observation time $T_{\text{obs, min}}$. Very few faces are observed for a period of time longer than $T_{\text{obs,max}}$, as required. The third plot (right) graphically represent the observation time of each face: brighter faces are the most observed ones, while black faces correspond to the unobservable ones.

This example highlights perhaps the most important feature of SBMPO algorithms: given as input an high-level scientific task (defined by an appropriate mission score), the algorithm was perfectly able to plan the spacecraft trajectory, compute maneuvers and ensure the completion of the task, while respecting all the input and state constraints, without any further intervention of the user.

## 4. CONCLUSIONS

Motion planning algorithms represent an interesting complementary approach to classic spacecraft trajectory optimization techniques. Their ability to solve complex planning problems while fulfilling an "high-level" mission objective (e.g. a scientific task that cannot be directly translated into state waypoints or a predefined trajectory to be followed) is an extremely useful feature. SBMP solutions are perfectly adapted to solve the next generation of guidance problems, such as the autonomous landing on a small celestial body, its observation and mapping, as well as the assembly of large space structures by smaller assembler cooperative vehicles, the inspection of a space station, or the capture of a rapidly spinning large debris. These complex high-level missions, often characterized by a cluttered environment that is difficultly handled by classic optimization methods, represent the ideal application domain for SBMP techniques.

Even if the possibility to run these algorithms on-board a space vehicle has yet to be proven, the use of motion planning solutions appears to be an important step towards the design and on-board implementation of autonomous guidance and mission planning laws.

## REFERENCES

Dunlap, D.D., Collins, E.G.J., and Caldwell, C.V. (2008). Sampling Based Model Predictive Control with Application to Autonomous Vehicle Guidance. *2008 Florida Conference on Recent Advances in Robotics.*

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for Optimal Planning. *Int. Journal of Robotics Research.*

Kavraki, L., Svestka, P., Latombe, J.C., , and Overmars, M.H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation.*

Komendera, E., Scheeres, D.J., and Elizabeth, B. (2012). Intelligent Computation of Reachability Sets for Space Missions. *Proceedings of the Twenty-Fourth Innovative Applications of Artificial Intelligence Conference.*

Pavone, M., Acikmese, B., Nesnas, I., and Starek, J. (2014). Spacecraft Autonomy Challenges for Next Generation Space Missions. *Book chapter in Springer Lecture Notes in Control and Information Sciences.*

Ploen, S.R., Acikmese, A.B., and Wolf, A. (2006). A Comparison of Powered Descent Guidance Laws for Mars Pinpoint Landing. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Keystone, Colorado.*

Surovik, D.A. and Scheeres, D.J. (2014). Autonomous maneuver planning at small bodies via mission objective reachability analysis. *AIAA/AAS Astrodynamics Specialist Conference.*