OPTIMAL AUTONOMOUS PATH PLANNING OF A SPACE 6-DOF MANIPULATOR ARM

Gianantonio Magnani [(1)], Luca Bascetta [(1)], Gregorio Pannacci [(2)], Francesco Cavenago[(3)], Alessandro Pilati[(3)], Andrea Rusconi[(3)]

[(1)] Politecnico di Milano, piazza Leonardo da Vinci 32, 20133 Milano, gianantonio.magnani/luca.bascetta@polimi.it:
[(2)] Leonardo SpA, via Indipendenza 2, 21018 Sesto Calende (VA), Italy, gregorio.pannacci@leonardo.com
[(3)] Leonardo SpA, viale Europa, 20014 Nerviano (MI), Italy
francesco.cavenago/alessandro.pilati/andrea.rusconi@leonardo.com

ABSTRACT

This paper deals with the autonomous path planning of a 6-DoF robotic arm onboard a mobile platform in charge of picking Mars soil samples and placing them in special storage on the platform. RRT planners were considered first, and then their optimal extensions RRT* and Informed RRT*. Finally, DMI-RRT* based on the Informed RRT* approach was developed, capable of obtaining the suitable paths for a hypothetical mission of gripping an object on the ground and storing it on the platform, with limited computational effort. Path execution time is minimized by fully exploiting the actuators. Moreover, the generated paths respect the position, speed, and acceleration limits of the joints, and are free from collisions. The algorithm is currently coded in Matlab, and intended for offline use only. Nonetheless, since most of the computing effort is inherent to collision checking, a more efficient coding of the body 3D geometries and collision checking algorithm, should make the algorithm suitable for execution onboard the robot for real-time path re-planning.

1. INTRODUCTION

The reference scenario of this study is that of a mobile manipulator, composed of a manipulator arm and a mobile platform, in charge of picking Mars soil samples and placing them in special storage on the vehicle, as envisaged for the ESA Mars Sample Return mission.
Geometric, kinematic, and dynamic figures of a space arm similar to ESA / Leonardo′s arm Delian [1] are taken as reference. This arm has 6 degrees of freedom (DoF), is about one meter long when fully extended, is extremely lightweight, has very high gear ratios, and moves at a very low speed.
The problem of finding a path for a 6-DoF arm, from a start pose x_{start} to a target one x_{goal}, avoiding self-collisions and collisions with the environment, and minimizing an objective function, like the path length or execution time, while respecting constraints on joint position, velocity, and acceleration is quite challenging. Consequently, autonomous planning capabilities are beneficial even for offline use only, e.g., to plan the nominal operations on ground.
Several of the approaches to the autonomous path planning problem proposed in the literature are unusable because of the tremendous computational effort they would require for the 6-dimensional problem. Among them, there is the optimal version, RRT* [2], of the widely used Rapidly-Exploring-Random-Trees algorithm [3, 4]. Unlike RRT, RRT* allows us to optimize the path by minimizing a state-dependent cost function, in our case the path execution time. RRT* computational effort can be drastically reduced for high-dimensional problems by adopting the approach of informed planners [5, 6].
Informed RRT* is derived from RRT* and includes small but effective modifications in the sampling process. It behaves like RRT* until a first solution is found, after which it uses heuristics to restrict the region to be sampled to a subset of the configuration space, a hyper-ellipse, which contains the optimal searched path with a high probability.
Following the Informed RRT* approach, we have implemented DMI-RRT*, an algorithm that turns out to be able to find a viable solution to our 6-dimension planning problem with a sensible computational effort. The path search is performed in the joint space, and it is based on the kinematic and 3D geometric model of the arm and of the mobile platform. The minimized cost function is an estimate of the path execution time, computed as the sum of approximate estimations of the times that the manipulator arm needs to travel across each segment (edge) of the current path, considering the velocity limits (i.e., nominal velocities) of the single joints, and the space they have to travel.
The algorithm has been validated through numerical simulation considering a realistic operational scenario, which includes sample pick-up and storage in the mobile platform. The algorithm is capable of providing viable paths with limited computational effort.
This paper is organized as follows. In Section 2 the

reference arm kinematics is described, and the 3D model of the arm and the mobile platform is given. Three reference poses of the arm are introduced. In Section 3 the problem of autonomous path planning is posed, and three algorithms considered for its solution are described. Section 4 deals with the trajectory generation, also referred to as time parameterization, where the travel time is associated with the path to creating a trajectory, considering joint velocity and acceleration limits. Section 5 compares the different algorithms with respect to the quality of the path obtained and the computational effort required. In section 6, the main conclusions are drawn.

2. REFERENCE ARM MODELING

A sketch of the reference arm is shown in Fig. 1

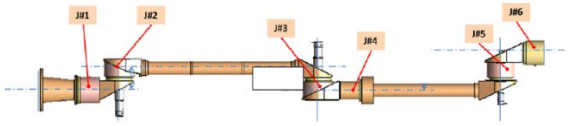

Figure 1. Reference arm schematic representation

The arm kinematics is given in Tab. 1

Joint	Type	a (mm)	d (mm)	α (deg)	θ (deg)
1	Revolute	0	190	90	ϑ_1
2	Revolute	536	-46.60	0	ϑ_2
3	Revolute	0	55.90	-90	ϑ_3
4	Revolute	0	435.50	90	ϑ_4
5	Revolute	0	-82.80	-90	ϑ_5
6	Revolute	0	179.35	0	ϑ_6

Table 1. Denavit-Hartenberg parameters

Tab. 2 gives the joint limits and shows the outstanding gearbox reduction ratios. The nominal velocity of joint motors is 200 rad/s, while the nominal (maximum) acceleration is 7840 rad/s^2.

Joint	Min (deg)	Max (deg)	Reduction Ratio
1	-170	170	44340
2	10	350	15700
3	-80	260	44340
4	10	350	27480
5	-350	-10	27480
6	-170	170	27480

Table 2. Joint position limits and gear ratios

A 3D-graphic model of the arm, the mobile platform carrying it, and of ground is needed for collision check and visualization of the planned trajectories. To restrain the calculation times, the vehicle and its wheels were approximated with boxes, as shown in Figures 2, 3, 4 which define three reference poses of the arm, namely Unstowage, SampleStorage, and Idle. The results of the autonomous planning of the paths between these poses will be illustrated in Section 5

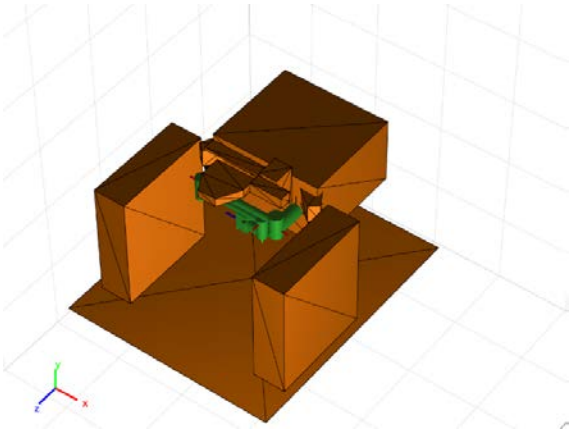

Figure 2. Unstowage pose

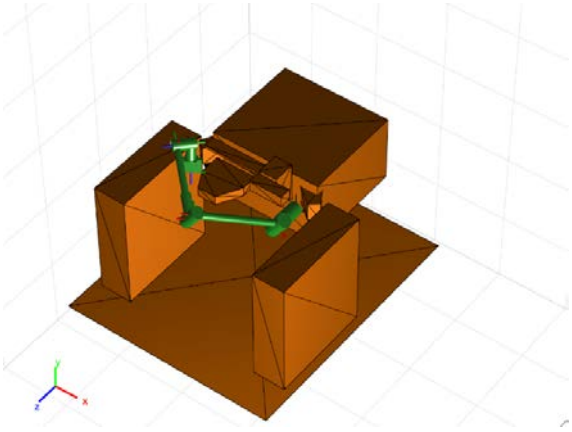

Figure 3. SampleStorage pose

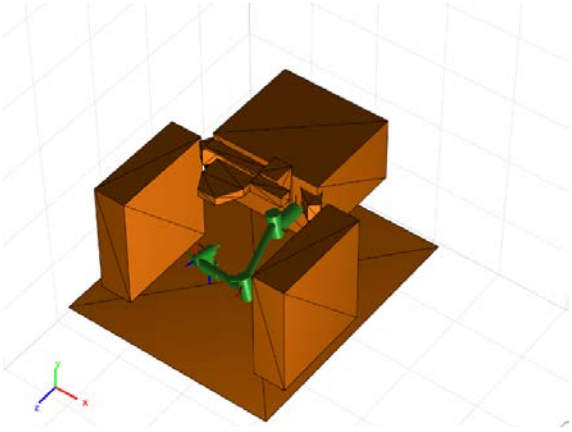

Figure 4. Idle pose

3. THE AUTONOMOUS PATH PLANNING ALGORITHM

Autonomous planning is performed in the joint space, where the workspace limits are defined, as well as the joint velocity limits. This choice also avoids possible difficulties with inverse kinematics.
Since the robot shall operate in a poorly structured and variable environment, single query sample-based planners were considered, starting from the widely used Rapidly-Exploring-Random-Trees algorithm.

RRT is widely used for its simplicity and ability to search efficiently non-convex and high-dimensional spaces. It builds a tree-like structure, rooted in the initial joint position set and directed towards the final one until a path from the source to the goal is obtained. The tree is built iteratively, adding at each iteration a new sample, namely a randomly selected, collision-free joint position set. The added sample is a new vertex of the tree, while edges are the segments connecting two consecutive vertices.

RRT allows to obtain the solution with acceptable computational effort even for problems with a space search of dimension 6, such as the path planning of the 6 DoF reference arm, but it does not allow to introduce optimality criteria on the searched path, such as minimizing the path length, or the travel time or the energy used by the actuators. To this end, RRT* algorithms can be used.

RRT* algorithms are based on RRTs, but are able to obtain probabilistically optimal paths as the number of samples approaches infinity. Unfortunately, the computational effort required by RRT* for our application, where it was required to minimize the execution time of the operations, and therefore the travel time between the initial and final joint position of each path, proved to be practically unacceptable.

Informed RRT* planners are a further evolution of RRT and RRT* planners which may drastically reduce the computational effort for high dimensional search spaces, as they use heuristics to focus the search of new samples on a restricted subset of the joint space which contains the optimal solution with a high probability.
Informed RRT* planners behave like RRT*s until a first solution is found, then they search for an improved solution only sampling the obstacle free part of a region shaped as a hyper-ellipsoid with focal points on x_{start} and x_{goal} with a transverse diameter of c_i, and conjugate diameter of $\sqrt{c_i^2 - c_{min}^2}$, c_i being the cost (length) of the current path, and c_{min} is the theoretical minimum cost (length or Euclidean distance between x_{start} and x_{goal}).

Following the Informed RRT* approach, we have implemented DMI-RRT*, an algorithm that proved able to find a viable solution to our reference planning problem, with a sensible computational effort.
DMI-RRT* is characterized by the following specifications:

- The path search is performed in the joint space, and it is based on the kinematic and 3D geometric model of the arm, and on a 3D "box model" of the mobile platform suitable for collision-checking;

- The distance between two joint position sets x_i, x_j ($x_i, x_j \in \mathbb{R}^{\mathbf{6}}$) is a weighted Euclidean distance
$$d(x_i, x_j) = \left\| w^T (x_i - x_j) \right\|_2$$
$w \in \mathbb{R}^{\mathbf{6}}$ being a weighting vector whose elements $w_i = \frac{v_{nmax}}{v_{ni}}$ depend on the *i*-th joint nominal velocity v_{ni} compared to the maximum nominal joint velocity $v_{nmax} = \max_{1 \le i \le 6} v_{ni}$

- The cost function to minimize is an estimate of the path execution time, computed as the sum of approximate estimations of the times that the manipulator arm needs to travel across each segment (edge) of the current path. The travel time (cost) $c_i(x_i, x_{i+1})$ of segment *i*, from joint position set x_i to x_{i+1} (intermediate joint positions between x_{start} and x_{goal}) is defined as:
$$c_i(x_i, x_{i+1}) = \max(tt)$$
$tt \in \mathbb{R}^{\mathbf{6}}$ being a vector whose 6 elements $tt_k = \left| \frac{x_{i,k} - x_{i+1,k}}{v_k} \right|$ are the times that joint *k* spends to move from position $x_{i,k}$ to position $x_{i+1,k}$ at its nominal velocity v_{nk}. The execution time therefore considers the velocity limits (i.e., nominal velocities) of the single joints and the space they have to travel. It is assumed that all six joints move synchronously in each segment.

We have coded in Matlab the RRT, RRT*, and Informed RRT* algorithms using powerful functions of the Robotics System Toolbox to assess their behavior and performance, and the relevant computational effort.

Inputs of the algorithm are the initial and final joint positions x_{start} and x_{goal}, and few tuning parameters, while the output is a structure *Sol* containing:
- the tree $\boldsymbol{T} = (\boldsymbol{V}, \boldsymbol{E})$, defined by its vertices $\boldsymbol{V}$ and edges $\boldsymbol{E}$, iteratively generated by the algorithm;
- X_{sol}, a set of *n* vertices, named waypoints, $V_i, i = 1, \dots, n$, being $V_0 := x_{start}$ and V_n close enough to x_{goal}, which composes the solution path. The waypoints are sequentially connected by *n*-1 edges $E_i, i = 1, \dots, n-1$;
- c_{sol}, cost of the solution.

At each iteration a sample joint position set free from collisions is extracted from the informed reduced region. Since it is computationally inefficient to sample points in 6D and to check whether they are or not inside the ellipse being the joint space too wide, joint positions are initially sampled inside a 6-D unitary sphere and then, through a proper transformation, transferred into the hyper-ellipse shape.

The sample (x_{new}) is then connected to the node of the current tree, among those within a given distance from it (*neighbor* nodes), that gives the cheapest path from x_{start} and x_{new} (*best parent* node), provided that the relevant edge is free from collisions.

The tree may be *rewired* if there is a path to any neighbor node passing through x_{new} that is cheaper than the current path; if a path with this characteristic exists the relevant node is connected to x_{new}.

Finally, the cheapest path given the current tree is found, and the outputs X_{sol} and c_{sol} are computed.

The iterative process stops when the difference of cost between two consecutive iterations falls under a given tolerance, or the eccentricity of the ellipse exceeds a given tolerance.

4. TRAJECTORY GENERATION

Once a path is derived it is necessary to associate time with it, to create a trajectory. Joint velocity and acceleration limits must be considered.

Consistently with the goal of completing the path in minimum time, we try to exploit the full potential of the actuators on each edge of the path. We then look for the joint that, moving to its nominal speed and acceleration, will take longer to complete its path. The time taken by this joint is taken as the edge execution time, and it is computed according to the velocity profile in Fig. 5, referred to edge *i* and joint *k:* $\dot{q}_C^k$, $\ddot{q}_C^k$ are joint *k* nominal velocity and acceleration (positive values), $\dot{q}_i^k$is joint *k* actual velocity in edge E_i, t_{Ci}^k and t_{fi}^k are respectively the duration of the acceleration phase and the time to run across edge E_i by joint *k* running at its own nominal velocity and acceleration.

Denoting the actual rotation of joint *k* across edge E_i as: $\Delta q_i^k = q_{i+1}^k - q_i^k$, q_i^k being the position of joint *k* at waypoint V_i, assuming for simplicity $\Delta q_i^k > \frac{1}{2}\ddot{q}_C^k {t_{Ci}^k}^2$, from the velocity profile we get:

$$t_{Ci}^k = \frac{\dot{q}_C^k - \dot{q}_{i-1}^k}{\ddot{q}_C^k};$$

and

$$t_{fi}^k = \frac{\Delta q_i^k + \left(\dot{q}_C^k - \dot{q}_{i-1}^k\right) t_{Ci}^k - \frac{1}{2}\ddot{q}_C^k {t_{Ci}^k}^2}{\dot{q}_C^k};$$

The actual time to run across edge E_i for all joints is then $T_i = \max_{k=1,6} t_{fi}^k$, and the acceleration time T_{Ci} is that of the joint of the maximum t_{fi}^k. Acceleration and velocity of each joint, but the one moving at nominal figures, are scaled so that all joints move synchronously and reach waypoint V_{i+1} at the same time instant. In this way, the edge is covered in minimum time, the motion is more regular and the power picks required by the power supply and drivers are minimized. The passage through waypoint V_{i+1} is guaranteed, even if the absence of collisions during the whole movement cannot be guaranteed. For this purpose, further collision checks would need, unless edges are short enough and/or suitable safety margins are taken.

The last edge has to be treated a special way as the constraint of zero final speed is added to that of the space to be covered.

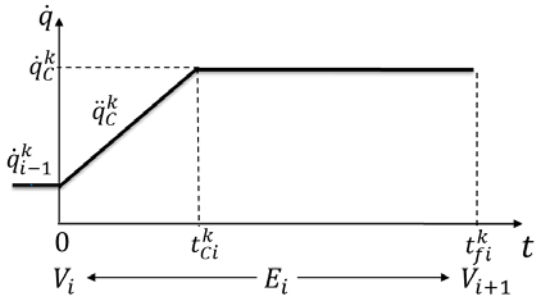

Figure 5 Velocity profile to run across edge E_i from waypoint V_i to waypoint V_{i+1}

5. RESULTS

DMI-RRT* is compared to RRT and RRT* to plan paths of a 2-DoF manipulator consisting of joints and links 1 and 2 of the reference arm, because this allows the easy presentation of joint space paths, and because RRT* needs too much computational time for the 6-DoF arm.

As illustrative examples, the result of planning the following two paths are shown:

- Unstowage-SampleStorage, representing a path with an obstacle in between. The trees built by the three algorithms and the relevant solution paths are shown in figures 6, 7, 8 and Tab. 3, comparing the solution paths in terms of quality (cost) and computational effort, underlines the advantages of using informed sampling; DMI-RRT* clearly outperforms the other planners both in terms of computational and solution

cost.

- Unstowage-Idle, representing a free from obstacles path. The results are shown in figures 9, 10, 11 and in Tab. 4.

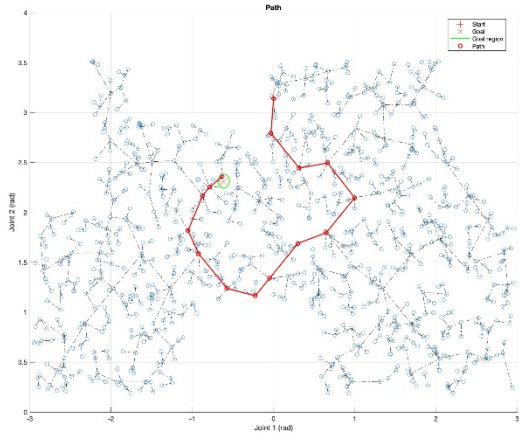

Figure 6 Path Unstowage–SampleStorage obtained by RRT planner

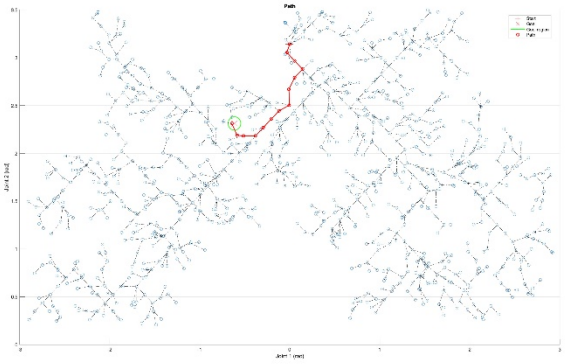

Figure 7. Path Unstowage– SampleStorage obtained by RRT* planner

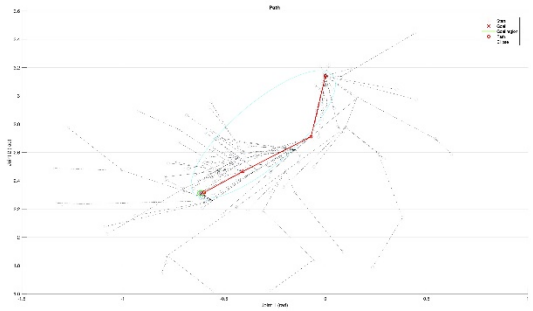

Figure 8. Path Unstowage–SampleStorage obtained by DMI-RRT* planner

Algorithm	Sol. Cost [s]	Comp. Cost [s]
RRT	230	317
RRT*	60	1538
DMI-RRT*	55	300

Table 3. Comparison in terms of cost (Unstowage-SampleStorage path)

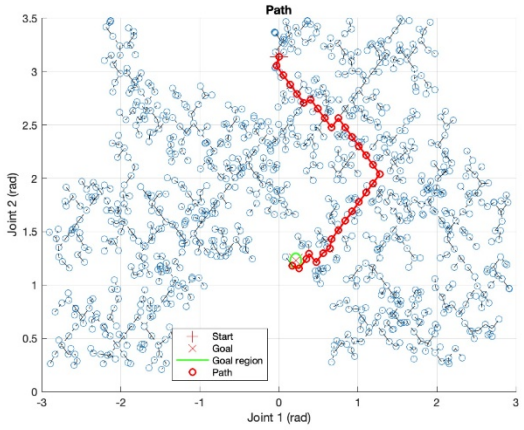

Figure 9 Path Unstowage–Idle obtained by RRT planner

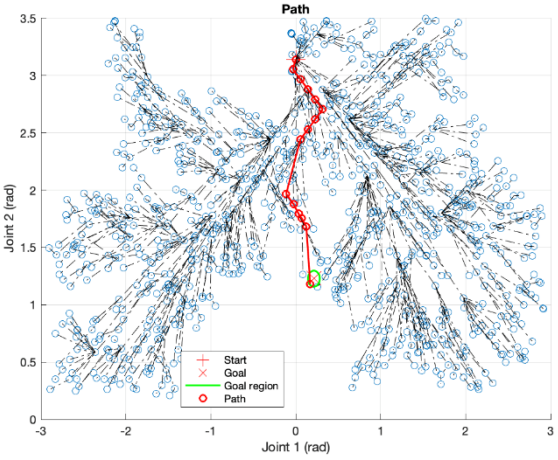

Figure 10 Path Unstowage–Idle obtained by RRT* planner

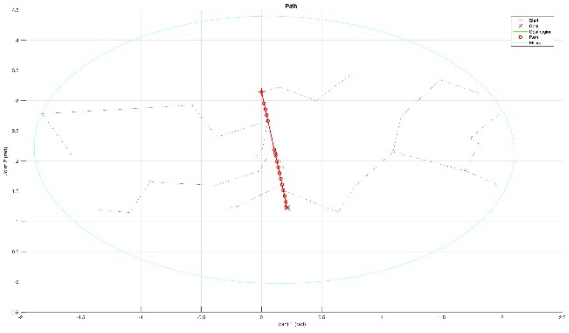

Figure 11 Path Unstowage–Idle obtained by DMI-RRT* planner

Algorithm	Sol. Cost [s]	Comp. Cost [s]
RRT	147	67
RRT*	113	1280
DMI-RRT*	109	10

Table 4. Comparison in terms of cost (Unstowage-Idle path)

Further tests showed that DMI-RRT* was able to find the Unstowage-SampleStorage path of the 6-DoF arm almost at the same computational effort spent by RRT* to find the same path of the 2-DoF arm.

The paths computed by DMI-RRT* for the complete mission of picking a soil sample and placing it in the storage on the platform is sketched in Figure 12.

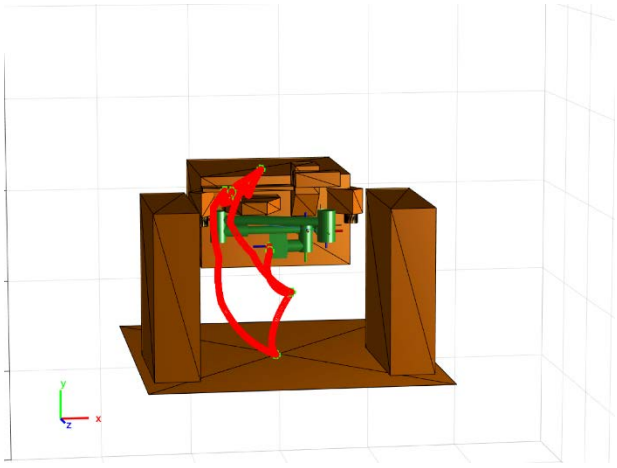

Figure 12 Paths of the complete mission of the 6DoF arm

6. CONCLUSIONS

In conclusion, a viable solution to the autonomous path planning of a 6 DoF manipulator arm operating in an unstructured and dynamic environment has been found, which respects and fully exploits the arm actuator's velocity and acceleration limits.

The solution is based on the idea of Informed RRT* planners, which are able to find optimal solutions, with a high probability, with moderate computational effort even for a 6-dimensional problem. We have coded the DMI-RRT* algorithm, as well as RRT and RRT* planning algorithms, in Matlab, using powerful functions of the Robotics System Toolbox, to assess the relevant performance and computational effort.

The choice to minimize a practical approximation of the path execution time, based on the nominal velocity of the arm joints, is a peculiarity of DMI-RRT*, intended to limit the time the robot needs to carry out its missions, partly compensating for its low speed.

The optimal version RRT* of the widely used Rapidly-Exploring-Random-Trees algorithm proved practically useless to solve the problem due to the excessive computational effort required.

On the contrary, DMI-RRT* proved to be able to find all envisaged paths for the application of collecting objects on Martian soil with limited and practically acceptable computational effort. Tests have shown that most of the computing effort is inherent to collision checking. This could be dramatically reduced with a more efficient coding of the body 3D geometries and collision checking algorithm, which could possibly allow the algorithm to be run onboard the robotic controller for real-time path re-planning.

7. REFERENCES

1. A. Rusconi, P. Magnani, S. Michaud, Delian-dextrous lightweight arm for exploration, ASTRA 2015
2. S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. The international journal of robotics research, 30(7):846–894, 2011.
3. LaValle, Steven M. (October 1998). "Rapidly-exploring random trees: A new tool for path planning" (PDF). Technical Report. Computer Science Department, Iowa State University (TR 98–11).
4. LaValle, Steven M.; Kuffner Jr., James J. (2001). "Randomized Kinodynamic Planning" (PDF). The International Journal of Robotics Research (IJRR). 20 (5): 378–400. doi:10.1177/02783640122067453. S2CID 40479452.
5. J. D. Gammell. Informed anytime search for continuous planning problems. Ph.D. thesis, University of Toronto (Canada), 2017.
6. J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Informed rrt*: Optimal sampling based path planning focused via direct sampling of an admissible ellipsoidal heuristic, 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2997–3004. IEEE, 2014.