Intelligent Service Robotics
https://doi.org/10.1007/s11370-023-00461-x

ORIGINAL RESEARCH PAPER

Multi-stage warm started optimal motion planning for over-actuated mobile platforms

Gonzalo J. Paz-Delgado[1,3] · Carlos J. Pérez-del-Pulgar[1] · Martin Azkarate[2] · Frank Kirchner[3] · Alfonso García-Cerezo[1]

Received: 21 November 2022 / Accepted: 10 March 2023

Abstract
This work presents a computationally lightweight motion planner for over-actuated platforms. For this purpose, a general state-space model for mobile platforms with several kinematic chains is defined, which considers dynamics, nonlinearities and constraints. The proposed motion planner is based on a sequential multi-stage approach that takes advantage of the warm start on each step. Firstly, a globally optimal and smooth 2D/3D trajectory is generated using the Fast Marching Method. This trajectory is fed as a warm start to a sequential linear quadratic regulator that is able to generate an optimal motion plan without constraints for all the platform actuators. Finally, a feasible motion plan is generated considering the constraints defined in the model. In this respect, the sequential linear quadratic regulator is employed again, taking the previously generated unconstrained motion plan as a warm start. The motion planner has been deployed into the Exomars Testing Rover of the European Space Agency. This rover is an Ackermann-capable planetary exploration testbed that is equipped with a robotic arm. Several experiments were carried out demonstrating that the proposed approach speeds up the computation time and increases the success ratio for a martian sample retrieval mission, which can be considered as a representative use case of goal-constrained trajectory generation for an over-actuated mobile platform.

Keywords Motion and path planning · Motion control · Mobile manipulation · Space robotics and automation

1 Introduction

Motion planning of mobile platforms is a well-known problem in the literature. It can be defined as finding a feasible trajectory for each actuator of the platform to perform a goal task interacting with the real world [1]. This entails many challenges depending on the characteristics of the platform, the scenario and the goal task. Nonetheless, it is remarkable that most systems have a common challenge for the motion planner: the computational effort. Mobile platforms computational resources are usually limited, even more for space [2, 3], air [4] or underwater [5] applications; or small-sized systems like sub-gram robots [6]. On top of that, the computational requirements increase substantially according to the system complexity, such as platforms that have redundant Degrees of Freedom (DoF), i.e. have more actuators than actually required to reach a particular pose. These are commonly called over-actuated platforms, where a clear example can be found on mobile manipulators, which are mobile platforms equipped with a robotic arm. The over-actuation entails the existence of infinite solutions for the

✉ Carlos J. Pérez-del-Pulgar
carlosperez@uma.es

Gonzalo J. Paz-Delgado
gonzalopd96@uma.es

Martin Azkarate
martin.azkarate@esa.int

Frank Kirchner
frank.kirchner@dfki.de

Alfonso García-Cerezo
ajgarcia@uma.es

[1] Department of Systems Engineering and Automation, Space Robotics Laboratory, Universidad de Málaga, Andalucia Tech, 29070 Málaga, Spain

[2] Automation and Robotics Section, European Space Agency, 2201 AZ Noordwijk, The Netherlands

[3] DFKI Robotics Innovation Center, Robert-Hooke-Str. 1, 28359 Bremen, Germany

Published online: 25 April 2023

motion planning problem, since the same final pose can be reached differently in function of the selected joints to be moved. Although a drawback in computational terms, over-actuation is actually an advantage for motion performance, since there are a great variety of solutions where to look for the one that best fits the problem needs.

There are multiple techniques that have been used to solve the motion planning problem, such as optimal control, potential fields, artificial intelligence or probabilistic algorithms. One of the most suitable techniques to tackle over-actuation without excessive computational load are the optimal control based ones: since there are infinite combinations of joint movements to reach the goal, the optimization will find at least the locally optimal one by defining a cost function that could take into consideration energy consumption, required time, etc. For instance, the Linear Quadratic Regulator (LQR) is a well-known method to solve optimal control problems, which can be executed iteratively to consider nonlinearities, which is commonly called iterative LQR (iLQR) [7] or Sequential LQR (SLQ) [8]. Nevertheless, these solvers find a main drawback when considering constraints, because it increases substantially the complexity of the problem and the required computational effort [9, 10]. Constraints can be commonly found in non-holonomic platforms and actuators limits. Also, problem related requirements are sometimes considered as constraints, e.g. accomplishing with a goal predefined trajectory. Different techniques can be found in the literature to tackle constraints. For example, Lazy Trajectory Optimization (LTO) was proposed in [11], which is a framework using Graph-Search Planning (GSP) altogether with Trajectory Optimization (TO) to plan long-term trajectories for robots. It was used to plan the motion of an over-actuated six legged robot in very cluttered scenarios, demanding a considerable computational effort (in the order of 500 s total planning time) due to considering kinematic, collision avoidance and equilibrium constraints. Another example can be found in [12], where the authors presented a particular modelling method called Virtual Kinematic Chain (VKC), able to integrate, for mobile manipulators, the kinematics of the base, arm and the object to be manipulated. It was applied to motion planning of daily tasks in a confined household environment. Using VKCs led to high success rates of the trajectory optimization, with less than 10 s computation time when including inequality constraints such as the mobile manipulation goals, the joint kinematic limits and collision avoidance. Additionally, in [13] authors proposed the use of Constrained SLQ [10] as a kinematic planner for over-actuated non-holonomic platforms, achieving high replanning frequencies in the order of 50 Hz but only considering equality constraints like joint position goals or end effector trajectories.

Even though the aforementioned methods are able to generate feasible motion plans to perform different tasks, even considering constraints, most of them make use of only kinematic models of the over-actuated platform. This simplification is often sufficient, however, there are use cases which require to take into consideration the system dynamics, for instance, torque-controlled or energy constrained systems. Considering the system dynamics widely increases the computational cost of each optimization iteration, consuming between 30% and 90% of the CPU time of many state-of-art motion planners [14]. This can be mitigated by reducing as much as possible the average number of iterations until convergence, which is achievable by means of the so-called warm start. The objective of warm start is to provide the optimization algorithm with a fast calculated initial solution of the problem, which is not necessarily feasible, but is used as a starting point that places the algorithm close to the convex area surrounding a local (or global) optimal solution. Hereby, the optimization is boosted to find the solution faster, in much fewer iterations. An example of warm start for over-actuated platforms was presented in [15], where different function approximation methods like *k-Nearest Neighbor* (k-NN), *Gaussian Process Regressor* (GPR) or *Bayesian Gaussian Mixture Regression* (BGMR) were used altogether to warm start multiple trajectory optimizations in parallel. This improved noticeably the performance of the motion planner, reaching 71% success rate and four times faster convergence when applied to the 34 DoFs humanoid robot ATLAS (Boston Dynamics) with the goal of reaching a random Cartesian pose.

It is also remarkable the use of several stages to warm start the optimization, as shown in [16], where a sequential refinement method for optimization was used to generate trajectories for a mobile manipulator to pick-up parts. This method keeps introducing problem constraints (grasping pose, gripper speed, collisions) sequentially to the optimization problem, continuously warm-starting and refining the solution, which improves the performance of the motion planner in comparison with the cold-started planner. These results were extended by the authors in [17], where a multi-staged warm started motion planner for a group of robots (up to three manipulators or a mobile manipulator with two robotic arms) was presented, including a deep analysis on the best sequence of introduction of the problem constraints (position, velocity, orientation of the end effector, collisions). The same multi-staged warm started motion planner was also used in [18] for surface disinfection with mobile manipulators, generating first a path to cover the goal area by means of a branch and bound-based tree search. This generated path was used as a constraint to generate the mobile manipulator actuators trajectory through the motion planner. As can be observed, these multi-staged warm start approaches have been applied to path-constrained trajectory generation problems, i.e. tasks where the end effector trajectory is precisely known a priori, thus, it is added as a problem constraint

in terms of end effector position, velocity and orientation. However, there are other cases where the initial trajectory is completely unknown or undefined, which can be denominated as goal-constrained trajectory generation problems. Additionally, the mentioned approaches have a major drawback for energy constrained use cases, since the models of the mobile platform do not consider system dynamics.

Seeing the current state of the art, it is clear that over-actuated systems are specially difficult to handle by motion planners, with the computational requirements becoming a bottleneck as the system constraints and complexity increases. To tackle this issue, multi-staged warm start arises as an interesting approach, which has demonstrated promising results to solve path-constrained trajectory generation problems.

Thus, in this paper we contribute with the definition of a novel multi-stage warm start strategy to solve the goal-constrained trajectory generation problem for over-actuated mobile platforms, taking into consideration the system constraints and dynamics. Two main contributions are stated. First, a generic model for over-actuated mobile platforms, that can be applied to systems with several serial kinematic chains. This model considers the system dynamics and external disturbances to later optimize the platform energy consumption. Second, a novel sequence for multi-staged warm started motion planning, aimed to solve the goal-constrained trajectory generation problem on over-actuated mobile platforms. The first warm start stage is a path planner that computes an initial trajectory for the mobile platform, the second stage takes that initial trajectory as warm start and solves only the unconstrained problem, and the third stage uses this unconstrained solution as a warm start to obtain the final motion plan with a constrained optimization solver. The benefits of this approach are demonstrated with a mobile manipulation for a martian sample tube retrieval use case, where the double-Ackermann rover equipped with a manipulator ExoTeR (Exomars Testing Rover), owned by the European Space Agency (ESA), was used. A laboratory test campaign with ExoTeR in the Planetary Robotics Laboratory (PRL) at ESA was carried out, to evaluate the strengths and weaknesses of the proposed approach. Additionally, a tailored replanning methodology was developed, based on the event-triggered replanning presented in [19], which is used to ensure that the mobile manipulator properly tracks the planned motion.

The rest of the paper is organized as follows. In Sect. 2 the motion planning algorithm is presented. In Sect. 3 the use case and the replanning methodology are shown. In Sect. 4 the experimental results are depicted. In Sect. 5 the carried out experiments are examined in detail. Finally, Sect. 6 concludes the paper with a final overview, including some comments on future related work.

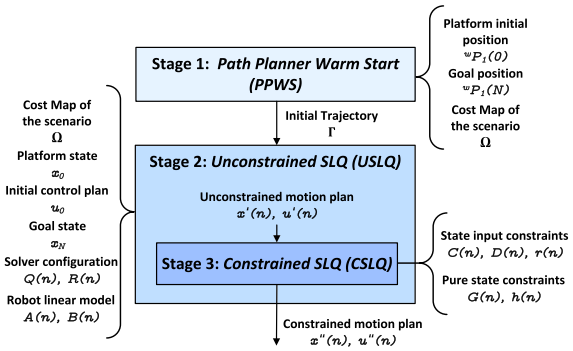

Fig. 1 Scheme summarizing the general functioning of the multi-staged motion planning approach. In the first stage (PPWS) an initial trajectory Γ is computed using Fast Marching Method (FMM). In the second stage (USLQ), this trajectory is used to warm start the Sequential Linear Quadratic (SLQ) optimization algorithm to solve the unconstrained motion planning problem. If the planned motion $x'(n), u'(n)$ satisfies the constraints, the algorithm finishes. Otherwise, the third stage (CSLQ) takes the unconstrained solution as warm start and computes the final motion plan $x''(n), u''(n)$, using again the SLQ optimization solver but with constraints compliance

2 Multi-staged warm started motion planning

The proposed Multi-staged Warm started Motion Planner (MWMP) sequence is designed to deal with high complex, over-actuated systems, looking for an efficient solution of the motion planning problem without severely impacting the computational resources of the system. For that purpose, a sequential warm start procedure is defined in this section, which reduces substantially the average number of iterations until convergence and, thereupon, the computational cost of the planner. A general overview of the functioning of the algorithm and its evolution through the different stages is shown in Fig. 1, which is further explained below.

Note that, in the following, any ${}^{j}P_{k}$ expresses the pose of k w.r.t. j, as ${}^{j}P_{k} = [p\ \phi]$, i.e. a position vector, in 3D $p = [x\ y\ z]$, and an orientation vector, in 3D $\phi = [\varphi\ \vartheta\ \psi]$, the roll, pitch, yaw Euler Angles. Assuming a generic mobile platform composed by a set of K kinematic chains, ${}^{j}P_{k}$ is the pose of the tip link reference frame of the kinematic chain k w.r.t. j, having w as the world reference frame. For instance, the center reference frame of a mobile platform would be ${}^{w}P_{1}$, and if it is equipped with a manipulator, the manipulator end effector pose w.r.t. the mobile platform would be ${}^{1}P_{2}$. To increase readability, a summary of all the expressions and notation used throughout this paper is shown in Table 1.

Each stage is executed in a particular order, sequentially. The first stage, called Path Planning Warm Start (PPWS), consists of a path planner based on the Fast Marching Method (FMM), although any other path planning algorithm could be used. FMM has been selected since it provides globally optimal and smooth solutions, with comparable computational

Table 1 Nomenclature

Symbol	Definition
T	Set of time steps
N	Number of time steps
Δt	Time step size
$x(n)$, $u(n)$	State and actuation vectors
$x'(n)$, $u'(n)$	Unconstrained state and actuation
$x''(n)$, $u''(n)$	Constrained state and actuation
x_0, x_N	Current and goal states of the platform
u_0	Initial control plan
$Q(n)$, $R(n)$	State and input quadratic costs
$A(n)$, $B(n)$	State space model matrices
$C(n)$, $D(n)$, $r(n)$	State-input constraint matrices
$G(n)$, $h(n)$	Pure state constraint matrices
${}^{j}P_k$, ${}^{j}\dot{P}_k$	Pose, speed of k w.r.t j
$p = [x\ y\ z]$	Position vector in 3D
$\phi = [\varphi\ \vartheta\ \psi]$	Orientation vector in 3D
w	World reference frame
${}^{w}P_1(0)$, ${}^{w}P_1(N)$	Platform initial and goal poses
K	Set of kinematic chains
q_k, $\dot{q}_k$, $\ddot{q}_k$	Position, speed and acc. of the joints in k
$\mathbb{I}_j$	Identity matrix with size jxj
${}^{j}R_k$	Rotation matrix given ${}^{j}P_k$
${}^{j}\mathcal{J}_k$	Jacobian matrix of k w.r.t j
I_k, V_k	Inertia and Coriolis matrices of k
e_k	Actuation effort vector of k
δ_z	External perturbations
f_z^k	Matrix representing the effect of δ_z into k
β_z^k	Auxiliary variable, $\beta_z^k = -I_k^{-1} f_z^k$
Γ	Optimal path used as warm start
Ω	Cost map of the scenario
$\tilde{x}_j$	Nodes of the cost map
$\tilde{x}_0$, $\tilde{x}_g$	Start and goal nodes of the cost map
$\hat{x}$	Any point inside the cost map
Υ	Cost to go trajectory planning
v	Path length from $\tilde{x}_0$
J	Total cost to go optimal control
$\Phi(x(N))$	Terminal cost optimal control
$L(x(n), u(n), n)$	Intermediate cost optimal control
$x^0(n)$, $u^0(n)$	State and input references or targets
$C_c(n)$, $D_c(n)$, $G_c(n)$	Active constraints
$\bar{x}_0(n)$, $\bar{u}_0(n)$	References for current iteration
$\hat{D}(n)$, $\hat{r}(n)$	Aux. constraints predefinition SLQ
$\hat{A}(n)$	Aux. state model predefinition SLQ
$\hat{Q}(n)$, $\hat{R}(n)$	Aux. costs predefinition SLQ
$\hat{x}^0(n)$, $\hat{u}^0(n)$	Aux. state input predefinition SLQ

Table 1 continued

Symbol	Definition
$\hat{P}(n)$, $s(n)$, $\hat{M}(n)$	Aux. backward pass variables SLQ
$\mathcal{C}_x$	Set of pure state constraints
$\Psi_c(n)$, $y_c(n)$, $h_c(n)$	Aux. matrices pure state constraints SLQ
$\Psi(c)$, $y(c)$, $H(c)$	Aux. matrices pure state constraints SLQ
$F_{c,j}(n)$, $F(c, j)$, $\nu(c)$	Aux. matrices pure state constraints SLQ
$\bar{v}(n)$, $\mu(n)$	Lagrangian multiplier vectors
$\lambda(n)$	Lagrangian co-state vector
$\bar{x}(n)$, $\bar{u}(n)$	Step plan for current iteration
α	Line search appliance step
θ_d, θ_s, θ_m	Driving, steering, arm joints
τ_d, τ_s, τ_m	Driving, steering, arm actuators torques
g	Gravity acceleration
ρ	Rolling resistance
m	Vehicle mass
d_w	Wheels diameter
$\mathcal{N}_w$	Number of wheels

cost to other non-optimal state-of-the-art planners [20]. As can be observed in Fig. 1, the path planner requires three inputs to compute the trajectory: the platform initial and goal positions w.r.t. the world frame, ${}^{w}P_1(0)$ and ${}^{w}P_1(N)$ respectively, and a cost map Ω representing the characteristics of the scenario, with higher costs where the mobile platform finds difficulties to traverse, like obstacles. Given this information, FMM takes two different steps to generate a path: first, a wave expansion, second, the trajectory extraction. This separation is advantageous, since it allows to generate new trajectories quickly as long as the goal remains the same, as will be explained later. As an output, the path planner generates the global optimal path Γ to reach the goal ${}^{w}P_1(N)$ from the platform initial pose ${}^{w}P_1(0)$ on the cost map Ω.

In order to accelerate the convergence speed, the extracted trajectory Γ is forwarded as a warm start to the next stage, called Unconstrained SLQ (USLQ), which makes use of an optimal solver called Sequential Linear Quadratic (SLQ) regulator [8], which is based on the Riccati equation. Although many other optimal solvers could also be used, SLQ has been selected due to its efficiency when solving nonlinear discrete optimal control problems with near-quadratic convergence, besides, requiring only first derivative information of the system states. As inputs, considering the set of N time steps that define the planning horizon $T = \{t_0,\ t_1,\ \ldots,\ t_n,\ \ldots,\ t_N\}$, the second stage requires the current and goal states of the platform, x_0 and x_N, an initial actuation plan u_0 usually filled with zeros, a configuration for the solver, i.e. the quadratic costs defined by the state matrix $Q(n)$ and the input matrix $R(n)$, and a linear state space model of the system, represented by the state transition matrix $A(n)$ and the input

distribution matrix $B(n)$. USLQ outputs, then, a complete unconstrained motion plan for the whole time horizon T, with $x'(n)$ the states and $u'(n)$ the actuation.

The above solver does not consider constraints yet, which implies faster convergence but could lead to unfeasible solutions. Thus, in the third stage, every generated solution is checked to confirm if the system constraints are satisfied. If they are, then the unconstrained motion plan is taken as correct and the planning pipeline finishes, $x''(n) = x'(n)$ and $u''(n) = u'(n)$, being $x''(n)$ and $u''(n)$ the final motion plan states and actuation, respectively. The overall computational cost of MWMP is reduced noticeably if this happens, since this third stage, called Constrained SLQ (CSLQ) [10], is the computationally most expensive. Otherwise, again the optimal solver is used to consider the system constraints, using the unconstrained solution $x'(n)$, $u'(n)$ as a warm start to boost the CSLQ performance, as will be shown with the obtained results in Sect. 3. This way CSLQ only has to refine the unconstrained solution to ensure constraint compliance, i.e. it already starts in the vicinity of the constrained optimal solution. Constrained SLQ additionally needs the definition of the constraints. On one hand, the system state-input constraints, which indicate actuation caps under particular system states, for instance joint effort limits, defined by the state-input constraints distribution matrices, with $C(n)$ for the state and $D(n)$ for the input, and the state-input constraints level $r(n)$. On the other hand, the pure state constraints, representing restrictions on the state vector that cannot be overcome, for instance joint position limits, defined by the pure state constraints distribution matrix $G(n)$ and the pure state constraints level $h(n)$.

After finding a constraint-compliant solution, the complete motion plan $x''(n), u''(n)$ is ready to be followed by the platform. Any tracking or control algorithm could be used to accurately follow the planned motion and compensate disturbances. For instance, Model Predictive Control (MPC), Event-triggered replanning or other procedures could be adequate for this purpose, in function of the platform and mission requirements.

Finally, the main prerequisite to use the motion planner is to obtain a state space model of the mobile platform, defining $A(n)$ and $B(n)$. Hence, a generic procedure for modelling over-actuated mobile platforms including several kinematic chains is presented below. Afterwards, the two main methods used in the motion planner are analyzed in detail, FMM as a path planner in the first stage and SLQ as an optimal solver for motion planning in the second and third stages.

2.1 State space model for a mobile platform

Let us define a state space model under linear discrete approximations as depicted in (1):

$$x(n+1) = A(n)x(n) + B(n)u(n) \tag{1}$$

Where $A(n)$ and $B(n)$ are the state transition and input distribution matrices, respectively, and $x(n)$ and $u(n)$ are the state and actuation vectors of the system, respectively, at time step n. As aforementioned, a generic mobile platform is represented as a set of K kinematic chains, with the pose of its tip link reference frame represented as ${}^{j}P_k$. The position of all the actuation joints belonging to kinematic chain k are denoted as vector q_k, including rotational and translational joints. Thus, the corresponding state vector $x(n)$ of a generic mobile platform is defined as (2).

$$\begin{aligned} x(n) = \big[{}^{w}P_1 \; {}^{w}\dot{P}_1 \cdots {}^{w}P_K \; {}^{w}\dot{P}_K \; {}^{1}P_2 \; {}^{1}\dot{P}_2 \cdots {}^{K-1}P_K \\ {}^{K-1}\dot{P}_K \; q_1 \; \dot{q}_1 \; \ddot{q}_1 \; q_2 \; \dot{q}_2 \; \ddot{q}_2 \cdots q_K \; \dot{q}_K \; \ddot{q}_K \big]^T \end{aligned} \tag{2}$$

With ${}^{w}\dot{P}_k$ the speed of the kinematic chain k w.r.t. the world reference frame; ${}^{k-1}\dot{P}_k$ the speed of the kinematic chain k w.r.t. the kinematic chain $k-1$; and $\dot{q}_k$, $\ddot{q}_k$ the speed and acceleration, respectively, of each actuation joint of k. Conversely, assuming a force/torque controlled platform, the actuation vector $u(n)$ is defined as (3).

$$u = \big[\, e_1 \, e_2 \cdots e_K \; \delta_1 \cdots \delta_Z \, \big]^T \tag{3}$$

Where e_k represents the actuation effort vector, i.e. forces or torques of the joints of k, and δ_z are external disturbances applying forces to the system, for instance, gravity. Note that, even though the external disturbances δ_z are included in the model inside the actuation vector, doubtlessly they are not under control of the system, hence, they should remain fixed to their expected values.

$$
A(n) = \begin{bmatrix}
\mathbb{I}_6 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & {}^{w}\mathcal{J}_1 \Delta t & 0\,0 & 0 & 0 \cdots 0 & 0 & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & {}^{w}\mathcal{J}_1 & 0\,0 & 0 & 0 \cdots 0 & 0 & 0 \\
\vdots & \vdots \ddots \vdots & \vdots\,\vdots & \vdots \ddots \vdots & \vdots\,\vdots & \vdots & \vdots\,\vdots & \vdots & \vdots \ddots \vdots & \vdots & \vdots \\
0 & 0 \cdots \mathbb{I}_6 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & 0\,0 & 0 & 0 \cdots 0 & {}^{w}R_{K-1}\,{}^{K-1}\mathcal{J}_K \Delta t & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & 0\,0 & 0 & 0 \cdots 0 & {}^{w}R_{K-1}\,{}^{K-1}\mathcal{J}_K & 0 \\
0 & 0 \cdots 0 & \mathbb{I}_6\,0 & \cdots 0 & 0\,0 & 0 & 0\,0 & {}^{1}\mathcal{J}_2 \Delta t & 0 \cdots 0 & 0 & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & 0\,0 & {}^{1}\mathcal{J}_2 & 0 \cdots 0 & 0 & 0 \\
\vdots & \vdots \ddots \vdots & \vdots\,\vdots & \vdots \ddots \vdots & \vdots\,\vdots & \vdots & \vdots\,\vdots & \vdots & \vdots \ddots \vdots & \vdots & \vdots \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots \mathbb{I}_6 & 0\,0 & 0 & 0\,0 & 0 & 0 \cdots 0 & {}^{K-1}\mathcal{J}_K \Delta t & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & 0\,0 & 0 & 0 \cdots 0 & {}^{K-1}\mathcal{J}_K & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & \mathbb{I}_{q_1} & \mathbb{I}_{q_1} \Delta t & 0\,0 & 0 & 0 \cdots 0 & 0 & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & \mathbb{I}_{q_1} - I_1^{-1} V_1 \Delta t & 0\,0 & 0 & 0 \cdots 0 & 0 & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & -I_1^{-1} V_1 & 0\,0 & 0 & 0 \cdots 0 & 0 & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & \mathbb{I}_{q_2} & \mathbb{I}_{q_2} \Delta t & 0 \cdots 0 & 0 & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & 0\,0 & \mathbb{I}_{q_2} - I_2^{-1} V_2 \Delta t & 0 \cdots 0 & 0 & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & 0\,0 & -I_2^{-1} V_2 & 0 \cdots 0 & 0 & 0 \\
\vdots & \vdots \ddots \vdots & \vdots\,\vdots & \vdots \ddots \vdots & \vdots\,\vdots & \vdots & \vdots\,\vdots & \vdots & \vdots \ddots \vdots & \vdots & \vdots \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & 0\,0 & 0 & 0 \cdots \mathbb{I}_{q_K} & \mathbb{I}_{q_K} \Delta t & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & 0\,0 & 0 & 0 \cdots 0 & \mathbb{I}_{q_K} - I_K^{-1} V_K \Delta t & 0 \\
0 & 0 \cdots 0 & 0\,0 & 0 \cdots 0 & 0\,0 & 0 & 0\,0 & 0 & 0 \cdots 0 & -I_K^{-1} V_K & 0
\end{bmatrix} \tag{4}
$$

Once defined the state and input vectors, a generic representation of the state transition matrix $A(n)$ is shown in (4), where $\mathbb{I}_j$ represents the identity matrix with size $(j \mathrm{x} j)$, ${}^{j}R_k$ a rotation matrix given the pose defined in ${}^{j}P_k$, ${}^{j}\mathcal{J}_k$ the Jacobian matrix relating articular with cartesian speeds for the kinematic chain k w.r.t. j, I_k and V_k the inertia and Coriolis/centrifugal matrices of k, respectively, and Δt the time step size. On the other hand, the input distribution matrix $B(n)$ is depicted in (5), with $\beta_z^k = -I_k^{-1} f_z^k$, and f_z^k a matrix representing the effect of the perturbation δ_z into the joints of kinematic chain k. It is crucial to remark that, with this definitions of $A(n)$ and $B(n)$, some of their terms could be nonlinear, which would eventually hinder the proper functioning of the motion planner. For that purpose, the use of Taylor Series Linearization (TSL) on those terms is recommended.

$$
B(n) = \begin{bmatrix}
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
I_1^{-1} \Delta t & 0 & \cdots & 0 & \beta_1^1 \Delta t & \cdots & \beta_Z^1 \Delta t \\
I_1^{-1} & 0 & \cdots & 0 & \beta_1^1 & \cdots & \beta_Z^1 \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & I_2^{-1} \Delta t & \cdots & 0 & \beta_1^2 \Delta t & \cdots & \beta_Z^2 \Delta t \\
0 & I_2^{-1} & \cdots & 0 & \beta_1^2 & \cdots & \beta_Z^2 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & I_K^{-1} \Delta t & \beta_1^K \Delta t & \cdots & \beta_Z^K \Delta t \\
0 & 0 & \cdots & I_K^{-1} & \beta_1^K & \cdots & \beta_Z^K
\end{bmatrix} \tag{5}
$$

Finally, it is key to define the system limits, via state-input and pure state constraints. On the one hand, the actuation effort e_k limits have to be specified as state-input constraints, where $C(n)$ is filled with zeros, since this constraints do not depend on the states, $D(n)$ indicates with ones or zeros which actuation effort limit is being defined, and $r(n)$ includes the actual limit values. On the other hand, as pure state constraints it is important to define the kinematic chains world ${}^{w}\dot{P}_k$ and relative ${}^{k-1}\dot{P}_k$ speed limits, and the position q_k, velocity $\dot{q}_k$ and acceleration $\ddot{q}_k$ limits of the actuation joints. Particularly, $G(n)$ indicates which state limit is being defined, filled adequately with ones or zeros, and $h(n)$ includes the values of those limits.

Considering this general definition of $x(n)$, $u(n)$, $A(n)$ and $B(n)$, the motion planning problem for a mobile platform with K serial kinematic chains can be redefined as finding a set of actuation efforts (forces/torques) e_k that generate a motion profile $(q_k, \dot{q}_k, \ddot{q}_k)$ for each joint of the platform, in order to place the tip link of the last kinematic chain in certain poses $({}^{w}P_K, {}^{w}\dot{P}_K)$, given the effect of external perturbations (δ_j) and the system limits, expressed through the state-input constraints ($C(n)$, $D(n)$ and $r(n)$) and the pure state constraints ($G(n)$ and $h(n)$). Note that some of the states are not strictly necessary in the model $({}^{w}\dot{P}_k, {}^{k-1}\dot{P}_k, \ddot{q}_k)$, but, as will be explained later, are helpful to set desired behaviours by tuning their corresponding costs, or to establish system constraints as aforementioned. Also bear in mind that, although required for an appropriate behavior of the motion planner, the linearization with $A(n)$ and $B(n)$ induces execution errors. These errors significance depend on the time step size Δt, being them negligible if Δt is sufficiently small, as it is demonstrated in Sect. 5.

2.2 Trajectory planning with FMM

The goal of the first stage, PPWS, is to generate an initial reference trajectory for the mobile platform to reach the goal, which will be later used as a warm start of the optimization algorithm to accelerate its convergence, as can be seen in Fig. 1. In particular, we propose Fast Marching Method (FMM) [20] as the warm start path planner, since, considering the scenario in form of a cost map, it extracts a globally optimal, smooth and continuous path to reach the goal. Remark that FMM has been widely used in the literature as a path planner [21] for different applications, that span from Unmaned Surface Vehicles (USVs) [22] to planetary rovers [23].

First of all, as aforementioned, FMM requires a proper representation of the scenario as an input cost map Ω. The cost map Ω is a discrete 2D or 3D grid, where each regularly scattered node $\tilde{x}_j$ has an associated cost $\Omega(\tilde{x}_j)$ that represents how easy and safe is for the platform to be placed in that position. Subsequently, obstacles should have the highest costs, and traversable areas the lowest ones. Areas surrounding obstacles should also have high costs, to avoid the platform getting close to them. Additionally, any other feature that influences the platform behaviour should be considered in the cost map. For instance, slopes and terramechanic properties of the soil in the case of Unmanned Ground Vehicles (UGVs).

FMM numerically solves a particular nonlinear Partial Derivative Equation (PDE) called the Eikonal equation, modelling the rate of propagation of a wave. This wave expands on the cost map Ω from the goal node $\tilde{x}_g$ visiting each node $\tilde{x}_j$ to generate the cost to go $\Upsilon(\tilde{x}_j, \tilde{x}_g)$, which indicates the accumulation of cost required to reach the goal $\tilde{x}_g$ from the node $\tilde{x}_j$. As indicated in (6), the rate of propagation of the wave at a certain node is equal to the cost at that node $\Omega(\tilde{x}_j)$.

$$\nabla\Upsilon(\tilde{x}_j, \tilde{x}_g) = \Omega(\tilde{x}_j) \quad \forall \tilde{x}_j \in \Omega \tag{6}$$

The higher the cost $\Omega(\tilde{x}_j)$ the slower the propagation of the wave on that node $\tilde{x}_j$. In this case, the wave propagation starts from the goal $\tilde{x}_g$, therefore the cost to go of the goal node is zero ($\Upsilon(\tilde{x}_j = \tilde{x}_g, \tilde{x}_g) = 0$).

The cost to go between the starting $\tilde{x}_0$ and the goal nodes $\tilde{x}_g$ is the minimum possible if $\Omega(\tilde{x}_j)$ always returns positive nonzero values. Thus, following the Dynamic Programming (DP) principles, any point $\hat{x} \in \Omega$ is placed in the optimal path connecting the starting and goal nodes, $\Gamma(\tilde{x}_0, \tilde{x}_g)$, if the sum of the costs to go from the starting node to the point $\Upsilon(\tilde{x}_0, \hat{x})$ and from the point to the goal node $\Upsilon(\hat{x}, \tilde{x}_g)$ is equal to the minimum cost to go $\Upsilon(\tilde{x}_0, \tilde{x}_g)$, as expressed in (7).

$$\Upsilon(\tilde{x}_0, \hat{x}) + \Upsilon(\hat{x}, \tilde{x}_g) = \Upsilon(\tilde{x}_0, \tilde{x}_g) \quad \forall \hat{x} \in \Gamma(\tilde{x}_0, \tilde{x}_g) \in \Omega \tag{7}$$

Hence, the objective of FMM is to solve the optimization problem defined in (8–9), i.e. finding the optimal path $\Gamma(\tilde{x}_0, \tilde{x}_g)$ that minimizes the cost accumulated along the path $\Omega(\Gamma(\tilde{x}_0, \tilde{x}_g, v))$, being $\Gamma(\tilde{x}_0, \tilde{x}_g, v)$ a continuous function that returns a point $\hat{x} \in \Omega$ given the path length v from the starting node $\tilde{x}_0$, with v_g the total length of the path.

$$\underset{\Gamma(\tilde{x}_0, \tilde{x}_g)}{\text{Minimize}} \quad \Upsilon(\tilde{x}_0, \tilde{x}_g) = \int_0^{v_g} \Omega(\Gamma(\tilde{x}_0, \tilde{x}_g, v))\, dv \tag{8}$$

$$\text{with} \quad \Upsilon(\tilde{x}_j = \tilde{x}_g, \tilde{x}_g) = 0 \tag{9}$$

Considering 1 as the kinematic chain defining the mobile platform, then $\tilde{x}_0 = {}^{w}P_1(0)$ and $\tilde{x}_g = {}^{w}P_1(N)$. Consequently, the warm start trajectory Γ is used as a reference for the pose of the platform ${}^{w}P_1$ at each time step n. It is beneficial, besides, to enrich the generated path with some more information. On one hand, the orientation of the platform at each waypoint is computed to also warm start the orientation states of the system. This is particularly helpful for platforms

with non-holonomic constraints, since including the yaw in the trajectory gives the mobile platform a big hint on how to properly follow the path, being the yaw obtained geometrically known the position of two consecutive waypoints. On the other hand, each waypoint should be timestamped, which can be easily done by interpolating the total expected time for finishing the operation t_N at each time step n. Note that there are many different approaches to estimate t_N according to the characteristics of the system, its nominal speed or the use case, which is out of the scope of this paper.

To finalize, remark that the wave propagation is, for FMM, the most computationally expensive step, being the trajectory extraction computationally negligible. This is very convenient for replanning the motion, since a new optimal trajectory from the current pose of the platform can be obtained quickly, i.e. without recomputing the cost to go. This only needs to be done once, offline, or in case that the goal changes. Refer to [23] for more details about the FMM-based path planner.

2.3 Sequential linear quadratic (SLQ) optimal solver

The next stages of MWMP make use of an optimal solver, which tackles the unconstrained problem in stage two (USLQ) and the constrained one in stage three (CSLQ), to generate a motion plan for the platform, as shown in Fig. 1. As aforementioned, the solver is called Sequential Linear Quadratic (SLQ) regulator [8, 10].

Considering the set of N time steps that define the planning horizon $T = \{t_0, t_1, \ldots, t_n, \ldots, t_N\}$, the standard formulation of a discrete-time optimal control problem is shown in (10–13).

$$\underset{u(n),\,x(n)}{\text{Minimize}} \quad J = \Phi(x(N)) + \sum_{n=0}^{N-1} L(x(n), u(n), n) \tag{10}$$

$$\text{subject to} \quad x(n+1) = f(x(n), u(n)), \quad x(0) = x_0 \tag{11}$$

$$C(n)x(n) + D(n)u(n) + r(n) \leq 0 \tag{12}$$

$$G(n)x(n) + h(n) \leq 0 \tag{13}$$

Where $x(n)$ is the state vector at time step n, noticeably, $x(0)$ is the initial state and $x(N)$ is the final one, and $u(n)$ is the actuation vector; x_0 defines the initial state of the system, at time step $t_0 = 0$. Additionally, (12) represents the state-input inequality constraints with the constraints distribution matrices $C(n)$, $D(n)$ and the constraints level vector $r(n)$, and (13) the pure state constraints with the constraints distribution matrix $G(n)$ and the constraints level vector $h(n)$, as aforementioned.

In this formulation, J is defined as the total cost to go, and it is composed of $\Phi(x(N))$, the terminal cost, and $L(x(n), u(n), n)$, the intermediate cost. Assuming a quadratic performance index, these are defined in (14) and (14) respectively.

$$\Phi(x(N)) = \frac{1}{2}[x(N) - x^0(N)]^T Q(N)[x(N) - x^0(N)] \tag{14}$$

$$\begin{aligned} &L(x(n), u(n), n) \\ &= \frac{1}{2}[x(n) - x^0(n)]^T Q(n)[x(n) - x^0(n)] \\ &+[u(n) - u^0(n)]^T R(n)[u(n) - u^0(n)] \end{aligned} \tag{15}$$

With $Q(n)$, $R(n)$ defined as the state and input quadratic cost matrices, respectively, at time step n, and $x^0(n)$, $u^0(n)$ the state and input references or targets. Note that $x^0(N) = x_N$ is the terminal state goal and $Q(N)$ the terminal state cost matrix, which is usually configured to have considerably high costs to ensure that the goal is accomplished. Note also the importance of properly tuning $Q(n)$ and $R(n)$ to precisely represent the desired behaviour of the system.

Solving the aforementioned discrete-time optimal control problem, the objective of the algorithm is to generate the motion plan, $x(n)$ and $u(n)$, for the whole time horizon T. To that purpose, several inputs are required. On one hand, the current state of the system x_0 and the desired goal state x_N are needed. On the other hand, as extensively explained above, an initial trajectory Γ is fed to the solver to accelerate the convergence speed. Consequently, the corresponding intermediate state costs $Q({}^w P_1, n)$ must be tuned with appropriate costs at every time step. These have to be high enough to help the solver, guiding it closer to the globally optimal path, but low enough to avoid forcing the solver to follow exactly the provided trajectory, which would reduce the variety of possible solutions to be explored.

Additionally, obstacle avoidance is always a requirement for any mobile platform. Although the warm start trajectory already considers the presence of obstacles in the scenario, another layer of obstacle avoidance is required, since the solver will draw a probably similar but new trajectory. Thus, USLQ and CSLQ need the same cost map Ω used in FMM at PPWS. $\Omega({}^w P_1(n))$ corresponds to a repulsive cost that gets the system away from danger, which means that the cost increases as the system gets closer to obstacles. This way the generated trajectories for the platform base pose ${}^w P_1$ dynamically get away from obstacles during the motion planning process in function of the cost map Ω, meanwhile trying to follow the warm start trajectory Γ. The intermediate cost defined above in (14) needs, then, to be reformulated, adding the repulsive cost $\Omega({}^w P_1(n))$, i.e. the cost value associated to the platform pose ${}^w P_1$ at time step n. Remark that it is key to maximize the continuity and linearity of the cost map, otherwise the solver will find difficulties to converge when encountering nonlinearities in the costs.

Algorithm 1 SLQ solver: Part 1

Initialization
$x(0) \leftarrow$ getCurrentState()
$u(n) \leftarrow$ getCurrentControlPlan()
$x^0({}^wP_1) \leftarrow$ getWarmStartTrajectory()
$C(n), D(n), r(n), G(n), h(n) \leftarrow$ getConstraints()
repeat
  Linearization and quadratization
  $x(n) \leftarrow$ forwardSimulateSystem$(x(0), u(n))$
  $Q(n), R(n) \leftarrow$ getQuadraticCosts()
  $A(n), B(n) \leftarrow$ getLinearizedSystem$(x(n))$
  $\Omega \leftarrow$ getObstaclesRepulsiveCost()
  $C_c(n), D_c(n), G_c(n) \leftarrow getActiveConstraints(C(n), D(n), r(n), G(n), h(n), x(n), u(n))$
  Reference tracking
  $\bar{x}_0(n) \leftarrow Q(n)(x(n) - x^0(n))$
  $\bar{u}_0(n) \leftarrow R(n)(u(n) - u^0(n))$
  Predefinitions
  $\hat{D}(n) \leftarrow (D_c(n)R(n)^{-1}D_c(n)^T)^{-1}$
  $\hat{r}(n) \leftarrow -D_c(n)R(n)^{-1}\bar{u}_0(n)$
  $\hat{A}(n) \leftarrow A(n) - B(n)R(n)^{-1}D_c(n)^T\hat{D}(n)C_c(n)$
  $\hat{R}(n) \leftarrow B(n)R(n)^{-1}[\mathbb{I} - D_c(n)^T\hat{D}(n)D_c(n)R(n)^{-1}]B(n)^T$
  $\hat{Q}(n) \leftarrow Q(n) + C_c(n)^T\hat{D}(n)C_c(n)$
  $\hat{x}^0(n) \leftarrow \bar{x}_0(n) + C_c(n)^T\hat{D}(n)\hat{r}(n)$
  $\hat{u}^0(n) \leftarrow -B(n)R(n)^{-1}[\bar{u}_0(n) + D_c(n)^T\hat{D}(n)\hat{r}(n)]$
  Backward Pass - Riccati matrix difference equation
  $\hat{P}(N) \leftarrow Q(N)$
  $s(N) \leftarrow \Omega({}^wP_1(N)) + \bar{x}_0(N)$
  for $n \leftarrow (N-1)$**;** n **in** T **do**
    $\hat{M}(n) \leftarrow (\mathbb{I} + \hat{R}(n)\hat{P}(n+1))^{-1}$
    $\hat{P}(n) \leftarrow \hat{Q}(n) + \hat{A}(n)^T\hat{P}(n+1)\hat{M}(n)\hat{A}(n)$
    $s(n) \leftarrow \hat{A}(n)^T\hat{M}(n)^T s(n+1) + \hat{A}^T(n)\hat{P}(n+1)\hat{M}(n)\hat{u}^0(n) + \hat{x}^0(n) + \Omega({}^wP_1(n))$
  end for
  State constraints (c) **management**
  if not isUnconstrained() **then**
    for $c \in \mathcal{C}_x = c_1, ..., c_S$ **do**
      $\Psi_c(t_c) \leftarrow G_c(t_c)$
      $y_c(t_c) \leftarrow 0$
      for $n \leftarrow (t_c - 1); n--; n > 0$ **do**
        $\Psi_c(n) \leftarrow \Psi_c(n+1)\hat{M}(n)\hat{A}(n)$

Algorithm 2 SLQ solver: Part 2

        $y_c(n) \leftarrow y_c(n+1) + \Psi_c(n+1)\hat{M}(n)[\hat{u}^0(n) - \hat{R}(n)z(n+1)]$
      end for
      $H(c), \Psi(c), y(c) \leftarrow h_c(t_c), \Psi_c(0), y_c(0)$
      for $j \in \mathcal{C}_x = j_1, ..., j_S$ **do**
        $n \leftarrow min(c-1, j-1)$
        $F_{c,j}(n+1) \leftarrow 0$
        for $i \leftarrow n; i \geq 0$ **do**
          $F_{c,j}(i) \leftarrow F_{c,j}(i+1) - \Psi_c(i+1)\hat{M}(i)\hat{R}(i)\Psi_j(i+1)^T$
        end for
        $F(c, j) \leftarrow F_{c,j}(0)$
      end for
    end for
    $v \leftarrow F^{-1}[-\Psi\bar{x}(n) + y + H]$
    for $n \leftarrow 0, ..., N$ **do**
      $s(n) \leftarrow s(n) + \sum_{c \in \mathcal{C}_x; c \geq n} \Psi_c^T(n)v(c)$
    end for
  end if
  Forward Pass
  for $n \leftarrow 0$**;** n **in** T **do**
    $\hat{v}(n) \leftarrow \hat{M}(n)(\hat{u}^0(n) - \hat{R}(n)s(n+1))$
    $\bar{x}(n+1) \leftarrow \hat{v}(n) + \hat{M}(n)\hat{A}(n)\bar{x}(n)$
    $\lambda(n+1) \leftarrow s(n+1) + \hat{P}(n+1)\bar{x}(n+1)$
    $\mu(n) \leftarrow \hat{D}(n)[C_c(n)\bar{x}(n) - D_c(n)R^{-1}(n)B(n)^T\lambda(n+1) + \hat{r}(n)]$
    $\bar{u}(n) \leftarrow -R^{-1}(n)[B^T(n)\lambda(n+1) + D_c(n)^T\mu(n) + \bar{u}_0(n)]$
  end for
  Computed step plan appliance
  if checkConstraints$(x(n), \bar{x}(n), u(n), \bar{u}(n), C(n), D(n), r(n), G(n), h(n))$ **or** isUnconstrained() **then**
    $\alpha \leftarrow$ computeLineSearch$(x(n), x^0(n), u(n), u^0(n), \bar{u}(n))$
  else
    $\alpha \leftarrow$ satisfyConstraints$(x(n), \bar{x}(n), u(n), \bar{u}(n), C(n), D(n), r(n), G(n), h(n))$
  end if
  $x(n) \leftarrow x(n) + \alpha\bar{x}(n)$
  $u(n) \leftarrow u(n) + \alpha\bar{u}(n)$
  Termination conditions
  convergence $\leftarrow$ checkTermination$(x(n), x^0(n), u(n), u^0(n), \bar{u}(n), \alpha)$
until convergence

Finally, an overview of the functioning of the solver for the discrete-time optimal control problem defined in (10–13) is depicted in Algorithms 1 and 2. Summarizing, given the current state x_0, the current actuation plan $u(n)$, the warm start trajectory Γ, the quadratic costs $Q(n)$ and $R(n)$, the system model $A(n)$ and $B(n)$, the cost map of the scenario Ω, the state-input $C(n)$, $D(n)$, $r(n)$ and the pure state $G(n)$, $h(n)$ constraints, this solver computes efficiently the motion plan $x(n)$, $u(n)$ by iteratively obtaining step plans $\bar{x}(n)$, $\bar{u}(n)$ to be applied to the current solution. To do so, the current active constraints are stored in $C_c(n)$, $D_c(n)$ and $G_c(n)$, which are later used to consider the constraints during the LQR solution computation. In particular, the state-input constraints are directly handled within the *Predefinitions* step, meanwhile the pure state constraints are managed later within the *State constraints management* step.

Algorithms 1–2 are based on the SLQ solver presented in [8] and [10], with a few differences. First, the state target sequence $x^0(n)$ is initialized with the warm start trajectory, as aforementioned. Second, the obstacles repulsive cost $\Omega({}^wP_1(n))$ is included into the backward pass. Third, at each iteration the constraints compliance is checked. If no new constraint is violated, then a standard *Line Search* for α is performed. The line search consists in finding the best α, which is the step size used to apply the step solutions $\bar{x}(n)$ and $\bar{u}(n)$, in order to reduce to the minimum the total cost to go J at each iteration of the solver. Otherwise, if any constraint is violated, α is generated particularly to satisfy the constraints. Fourth, several termination conditions are defined, on top of the algorithm convergence itself. In particular, one of these conditions checks that the last kinematic chain pose ${}^wP_K$ is close enough to the goal pose ${}^wP_K(N)$ to perform the desired task, depending on a given threshold, and another one ensures that the motion plan is

thoroughly safe. Note that Algorithms 1 and 2 encompass both the unconstrained and the constrained solvers, with $[C(n), D(n), r(n), G(n), h(n), C_c(n), D_c(n), G_c(n)] = 0$ in the unconstrained case, which means that $\hat{A}(n) = A(n)$, $\hat{R}(n) = B(n)R(n)^{-1}B(n)^T$, $\hat{Q}(n) = Q(n)$, $\hat{x}^0(n) = \bar{x}_0(n)$, $\hat{u}^0(n) = -B(n)R(n)^{-1}\bar{u}_0(n)$ and $\mu(n) = 0$. Besides, as can be observed, for the unconstrained case the *State constraints management* is not required, and the *Computed step plan appliance* is reduced to the first *Line Search*.

3 Use case: Martian sample tube retrieval

Planetary exploration vehicles are requiring more and more autonomy since remote teleoperation from Earth hinders to perform complex tasks such as navigation and manipulation [24]. A common strategy to increase autonomy for mid-long range traverses in planetary surfaces is a Guidance, Navigation and Control (GNC) architecture [2], which allows the rover to plan a path to the goal and navigate safely to it avoiding any intermediate hazard. Nevertheless, recent Mars vehicle concepts demand a further effort on the autonomous capabilities of the system, to satisfy the time and energy constraints imposed by the mission. For instance, Sample Fetch Rover (SFR) [25] was designed to collect several soil sample tubes, left by the Perseverance Mars2020 rover, to eventually bring them back to the Earth, with the requirement of going to the samples location, retrieving the samples and coming back to the lander within 150 sols [26]. Considering this critical time restriction, the necessity of performing the sample retrieval operations autonomously and efficiently arises, to increase the overall navigation speed of the system.

A sample tube retrieving rover is a highly over-actuated mobile platform, composed of a mobile base with multiple actuators (mainly driving and steering joints), and a robotic arm with several DoF. Considering, besides, the energy and time efficiency requirements of a planetary exploration mission, a martian sample tube retrieval is the perfect use case to demonstrate the advantages of the proposed optimal motion planning methodology. In particular, this paper focuses on a prototype of the Rosalind Franklin ExoMars rover from the Planetary Robotics Laboratory of the European Space Agency (ESA-PRL), called ExoTeR (Exomars Testing Rover) [27]. It is a triple-bogie, non-holonomic and double-Ackermann steered rover, equipped with a 5 DoF manipulator, which is modelled following the aforementioned generic over-actuated mobile platform state space model.

Finally, to test the generated motion plans in the real platform, it is besides necessary to include a motion plan follower in the loop. This follower filters any external disturbance or error during the execution of the motion plan, by replanning the motion when significant deviations are measured. Thus,

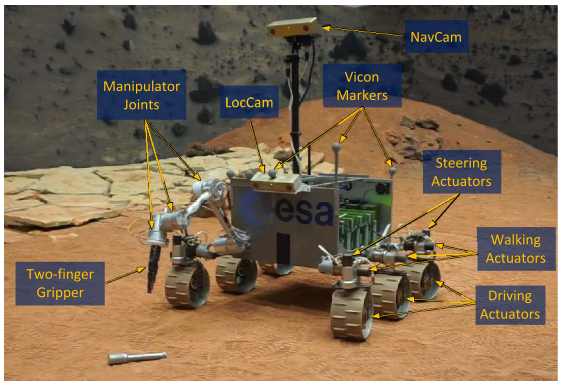

Fig. 2 Detail of the experimental setup with ExoTeR approaching a sample tube, including its actuators (driving, walking, steering and manipulator joints, two-fingers gripper) and its exteroceptive sensors (LocCam, NavCam). Besides, ExoTeR is equipped with an Inertial Measurement Unit (IMU) to estimate its orientation and several Vicon Markers to precisely locate it inside the Martian Analogue Testbed of the Planetary Robotics Laboratory, ESA-ESTEC

an analysis on the platform characteristics, a depiction on the developed state space model of the mobile manipulator and a detailed description of the tailored replanning methodology are presented in this section.

3.1 Mobile platform description

Within the research and development carried out at the Planetary Robotics Laboratory, Automation and Robotics Section, of the European Space Agency (ESA-PRL), the design and testing of planetary rover testbeds stands out. This is the case of ExoTeR [27], which conceptually mimics the early model of the Rosalind Franklin ExoMars rover, with a scaled-down concept. ExoTeR is a triple-bogie, double-Ackermann rover, with a locomotion system of $6 \times 6 \times 4 + 6$. This means 6 wheels with 6 driving actuators, 4 of them steerable (the front and rear ones), which permits double-Ackermann steering or spot turns. Additionally, all 6 wheels include a walking actuator, as depicted in Fig. 2, where ExoTeR is shown at the Martian Analogue Testbed at ESA-PRL. ExoTeR is also equipped with a 5 DoF manipulator, called MA5-E. Its five joints are rotational, with a Roll-Pitch-Pitch-Pitch-Roll configuration, being the first joint placed looking towards the movement direction of the platform. Its end effector has attached a two-fingered gripper for sample retrieval purposes. For localization and perception, ExoTeR has two stereo cameras, a close range LocCam and a long range NavCam. Finally, ExoTeR has also an Inertial Measurement Unit (IMU) for sensing the platform 3D orientation, and has appended several Vicon markers for ground-truth localization inside the martian testbed.

Focusing first on the mobile platform base, ExoTeR is a double-Ackermann rover with steering joints at the front and

Table 2 MA5-E joints characteristics

Joint	1	2	3	4	5
Type	Rot.	Rot.	Rot.	Rot.	Rot.
Orientation	Z	Y	Y	Y	Z
Range (°)	±45	±170	±170	±170	±170
Speed ($°/s$)	0.57	0.57	0.57	0.57	0.57
Power (W)	0.75	0.75	0.75	0.75	0.75
Gear ratio	83200:1	83200:1	83200:1	83200:1	83200:1
Efficiency	0.5	0.5	0.5	0.5	0.5

rear wheels. The central wheels do not steer, which implies that the rover cannot move in every direction depending on the system orientation, i.e. non-holonomic constraints. This is a significant non-linearity, which is tackled inside the system model. The platform minimum turn radius is 0.6 m, due to the geometric distribution of the wheels and the range limit of ±50° in the steering joints, although it can perform point turns (change its orientation with zero linear velocity). Finally, its nominal traslational speed is 5 cm/s, with a 2.85 Nm maximum torque of the driving actuators.

Regarding the robotic arm, MA5-E, its main characteristics are outlined in Table 2. As can be observed, MA5-E joints have huge gear ratios, which allows it to handle heavy payloads, $2kg$, in comparison to the arm weight, $2.4kg$, and considering the joint motors power ($6W$). The dynamic effect of external disturbances is consequently negligible, i.e. gravity and the rover base movements. Nevertheless, the gears also imply an important drawback: the joints move very slowly, with a maximum rotational speed of $0.57°/s$.

The fully extended arm lengths $0.527m$, with additional $0.14m$ taking into account the gripper. The arm end effector reachability is restricted by each joint position limit, as can be observed in Table 2. Doubtlessly, the arm movements are also limited by the rover body itself and the ground. Regarding the end effector, it cannot reach any orientation because of the limitation of the 5 DoF configuration. This issue hinders any manipulation task, especially a sample retrieval operation, since the end effector cannot always approach the sample completely perpendicular to the ground, with the appropriate gripper yaw w.r.t. the sample tube.

3.2 Double-Ackermann mobile manipulator model

As a mobile manipulator, ExoTeR is modelled with two different kinematic chains: the full-Ackermann mobile base and the robotic arm. The dynamics coupling between them is ignored, seeing that the movements of both the platform and the manipulator are very slow, generating negligible dynamics effects between them. Additionally, the effect of gravity into the manipulator joints is also disregarded, considering the huge gear ratios as aforementioned.

Following the generic state space model explained before, the state vector $x(n)$ for ExoTeR is defined in (16).

$$x(n) = \left[{}^{w}P_1 \; {}^{w}\dot{P}_1 \; {}^{w}P_2 \; {}^{1}P_2 \; {}^{1}\dot{P}_2 \; q_1 \; \dot{q}_1 \; \ddot{q}_1 \; q_2 \; \dot{q}_2 \; \ddot{q}_2 \right]^T \quad (16)$$

Where the kinematic chain 1 represents the mobile base and 2 the manipulator, thus, q_1 corresponds to the mobile base joints, i.e. the wheel driving θ_d and steering θ_s joints, and q_2 corresponds to the manipulator joints θ_m, which are all rotational as aforementioned. As a result, the state transition matrix $A(n)$ is extracted straightforwardly from the generic one, but specifically for a platform with two kinematic chains. In particular, I_1, V_1 and ${}^{w}\mathcal{J}_1$ refer to the inertia, Coriolis/centrifugal and Jacobian matrices of the full-Ackermann non-holonomic mobile base, as well as I_2, V_2 and ${}^{1}\mathcal{J}_2$ refer to the 5DoF manipulator. Remark that, as aforementioned, the mobile base Jacobian ${}^{w}\mathcal{J}_1$ is linearized by means of a TSL considering the notable non-linearity that appears due to the non-holonomic constraints.

The actuators of the system are the six driving and four steering joints of the wheels of the mobile base and the five rotational joints of the manipulator. As external disturbances, gravity acts on the mobile platform as a constant acceleration. Thus, the actuation vector $u(n)$ is defined in 17.

$$u(n) = \left[\tau_d \; \tau_s \; \tau_m \; g \right]^T \quad (17)$$

Where τ_d, τ_s and τ_m are the actuation torques to the driving, steering and manipulator joints, respectively, and g is the gravity acceleration. Note that ExoTeR joints only receive position and velocity commands, nevertheless, using the whole dynamics model allows to generate torque-efficient motions. Later on, the joint position and speed commands are directly extracted from the state vector $x(n)$.

Once more, the input distribution matrix $B(n)$ is directly obtained using the generic one presented in Sect. 2 but with only one external disturbance, the gravity g. On one hand, the effect of g into the mobile base, f_1^1, generates a wheel-soil friction, which is modeled in a simplified way by means of the rolling resistance of the terrain as expressed in (18), with ρ the rolling resistance coefficient of the terrain, d_w the diameter of the wheels, m the mass of the vehicle and $\mathcal{N}_w$ the number of wheels of the rover. On the other hand, the effect of g into the manipulator, f_1^2 is ignored, as aforementioned, considering the huge gear ratio of the arm joints.

$$f_1^1 = \rho \, \frac{d_w}{2} \, \frac{m}{\mathcal{N}_w} \quad (18)$$

Finally, several constraints have been defined to consider ExoTeR limits. On the one hand, the maximum actuation torque for the driving (τ_d), steering (τ_s) and manipulator (τ_m) joints are included as state-input constraints. On the other

hand, the limits on the velocity and acceleration of the driving ($\dot{\theta}_d, \ddot{\theta}_d$), the steering ($\dot{\theta}_s, \ddot{\theta}_s$) and the manipulator ($\dot{\theta}_m, \ddot{\theta}_m$) joints are defined as pure state constraints. Additionally, the position limits of the steering θ_s and manipulator θ_m joints are also included as pure state constraints.

3.3 Replanning capability

Given a feasible motion plan, i.e. the state $x(n)$ and actuation $u(n)$ vectors for the complete planning horizon T, a separate component needs to bring it to the mobile platform, ensuring it is properly followed until reaching the goal. If there are deviations from what was planned, then this component has to take the right decisions to ensure that the goal is reached. This deviations can be caused by different means, like the model intrinsic errors because of the discretization and the linearization. But it is also pertinent to consider the effect of other agents into the system, such as external disturbances not considered initially in the model, e.g. the platform localization error, the goal pose estimation error or the non-ideal behaviour of the actuators. In planetary exploration use cases the rover localization has a certain error, which, in the particular case of sample tube retrieval, adds up to the sample positioning error induced by the sample detection and localization subsystem. This sample positioning error is expected to be higher as further the rover is from the sample, arising the necessity of replanning the motion as the positioning error gets smaller, i.e. as the system gets close to the sample.

Therefore, a motion plan follower has been developed, which implements a replanning capability similar to the Event-triggered one proposed in [19], in the following manner. First, the motion plan follower sends sequentially the next actuation command to the platform (or the first one initially). Second, it checks if the goal pose has changed. If this is the case, then the system returns to the wave expansion of stage one (PPWS) and replans the whole motion. If not, a third step checks if there is too much drift in any of the controlled states. This would lead to returning to the trajectory extraction of stage one, which uses the current platform pose to replan the motion. Fourth, if no replan is needed and the goal is reached, the follower finishes the execution. Otherwise, it continues sending actuation commands in accordance to the already generated motion plan, and starts again the sequence.

In case one of the states drifts from the planned motion, the behaviour of the replanning capability is exemplified in Fig. 3. Starting with a planned motion (dark blue) from t_0 to t_N, with N number of time steps of Δt size, and given the time evolution of a controlled state $x(n)$ (dark green) in accordance to a given actuation plan $u(n)$ (dark red), this evolution may differ from the plan, increasingly accumulating error. When this error surpasses a certain threshold at time step t_n, the predicted behaviour of $x(n)$ is completely

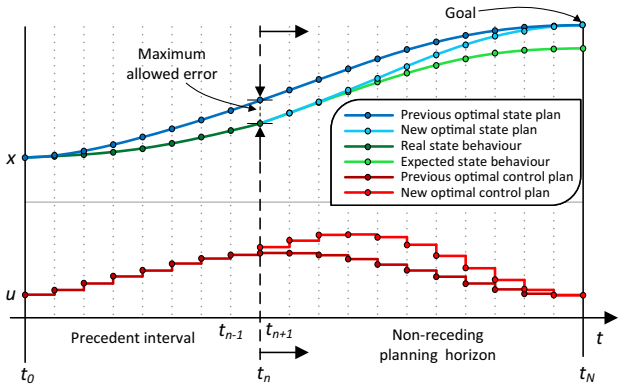

Fig. 3 Graph exemplifying the replanning capability. The behaviour of a controlled state $x(n)$ is continuously checked (dark green). If a considerable deviation from the previous plan (dark blue) is detected, a new global motion plan (light blue) is computed from the time step t_n onwards. The new state and actuation plans (light red) allow the system to reach the goal smoothly correcting the previously accumulated error

undesired (light green), thus, a replan is launched using the previous motion plan, from t_n onwards (dark blue), as a warm start. Thus, the replanned motion (light blue) compensates the accumulated drift in $x(n)$ by slightly modifying the previous optimal actuation plan $u(n)$ (dark red), generating a new one (light red) in the neighbourhood of the previous solution. In this way, the state $x(n)$ will still reach the goal as long as the new motion plan is properly followed.

4 Results

The proposed method for motion planning was validated by means of several tests with the sample tube retrieval use case. On one hand, a deep performance analysis of the motion planner was performed with a benchmark between different layouts of the approach, i.e. using different combination of the already explained stages. This comparison confirmed that the proposed warm-start sequence is the most convenient, with a path planning warm start, a first unconstrained stage and a final constrained stage. On the other hand, several laboratory tests were performed with the Exomars Testing Rover (ExoTeR) in the Martian Analogue Testbed of the Planetary Robotics Laboratory (PRL) of the European Space Agency (ESA). Using MWMP and the proposed replanning procedure, ExoTeR was capable of successfully reach a martian sample tube and retrieve it with its manipulator. The source code of the MWMP library used within these tests is available in MatLab[1] and C++,[2] under MIT open source license.

Note that ExoTeR has a series of system constraints, as explained in Sect. 3, which arise the necessity of using MWMP. First, a coupled arm-base motion solves the arm joints velocity issues, generating optimal motions where the

[1] https://github.com/spaceuma/MWMP-MatLab.

[2] https://github.com/spaceuma/MWMP-Cpp.

arm is already prepared to perform the desired task once the base has reached the objective. Second, the rover DoFs can be used to place the manipulator in a certain manner to effectively retrieve the sample, i.e. aligning the manipulator first joint and the sample tube, performing completely perpendicular retrieval operations. This is achieved by properly tuning the costs associated to the goal pose of the end effector, including its orientation, as will be clarified below.

Thereupon, this section is divided into different subsections. First, the experimental setup, the scenario and the motion planner configuration are thoroughly detailed. Second, the performance benchmark is exposed. Third, the laboratory tests are presented.

4.1 Experimental setup

The goal of the performed laboratory tests was to demonstrate that ExoTeR can reach and retrieve a martian sample tube in a completely autonomous way. For that purpose, these tests were carried out in the Martian Analogue Testbed at the ESA-PRL, which can be observed in Fig. 2. This is a 9x9 m experimental terrain which is highly representative of a real martian environment, including different types of soil (sandy, rocky), rocks or small slopes.

The tests include real martian autonomous navigation restrictions in order to perform an illustrative emulation of a sample tube retrieval mission. Therefore, two additional subsystems were integrated in the platform, apart from the presented MWMP and replanning algorithms. First, an autonomous sample detection and localization subsystem based on Convolutional Neural Networks (CNNs), which uses the LocCam stereo images to locate the sample tube with an average under 5 cm position and 5° orientation errors [28]. Second, a visual odometry algorithm for the platform localization, using the LocCam stereo camera and the IMU, with 7.5 % average localization drift in position and less than 2° orientation error [29]. The Vicon markers were also used to obtain the ground-truth localization, not online but for data logging and post-processing purposes.

It was necessary to properly configure the motion planner for the tests. First of all, the cost map of the scenario was generated by processing a 2 cm resolution Digital Elevation Map (DEM) of the PRL. This cost map considers obstacles, slopes and roughness, for more information about the cost map generation see [30]. The tests time horizon t_N was 160 s, with a time step Δt of 0.8 s. For the motion planner to converge, the maximum allowed position error was set-up to 1 cm and the orientation error to 10°. It was considered that the algorithm had converged if the norm of the stepped actuation $\bar{u}(n)$ was lower than 1 % of the norm of the whole actuation vector $u(n)$. The particular costs which configured the LQR cost matrices $Q(n)$, $R(n)$ are defined in Table 3. Note that the cost of modifying the input gravity disturbance g in $R(n)$ is the largest, to ensure that it remains as a constant gravity acceleration of $9.81 m/s^2$ precisely following the reference u^0. This gravity disturbance is ignored for the manipulator, as aforementioned. Additionally, to ensure that the sample was retrieved perpendicularly to the ground, the goal pose orientation ${}^w\phi_2(N)$ was filled with roll ${}^w\varphi_2(N) = 0$, and the pitch ${}^w\vartheta_2(N)$ and yaw ${}^w\psi_2(N)$ were computed depending on the estimated sample orientation.

Table 3 Quadratic costs configuration

Type	Variable	Cost
Goal state $Q(N)$	EE pose ${}^w P_2$	10^{11}
	Platform speed ${}^w \dot{P}_1$	10^6
	End effector speed ${}^1 \dot{P}_2$	10^6
State full motion $Q(n)$	Platform pose ${}^w P_1$	20
	Driving wheels speed $\dot{\theta}_d$	100
	Driving wheels acc. $\ddot{\theta}_d$	10^4
	Arm joints speed $\dot{\theta}_m$	$3 \cdot 10^5$
	Arm joints acceleration $\ddot{\theta}_m$	$3 \cdot 10^5$
Input full motion $R(n)$	Wheels driving torque τ_d	10^5
	Steering joints torque τ_s	$8 \cdot 10^4$
	Arm joints torque τ_m	10^{11}
	Gravity g	10^{15}

Lastly, the replanning was launched in accordance to certain errors when following the planned motions. In the first place, if the platform drifted more than 4 cm from the planned path. In the second place, if any of the controlled joints deviated more than 2.29° from the plan. Additionally, every time the goal sample pose differed more than 3 cm or 17.19° from the previous estimation, a complete motion replan was launched.

4.2 Motion planner performance analysis

The performance of the proposed motion planning approach was analyzed using the aforementioned use case and setup, to showcase its advantages w.r.t. any possible layout of the stages, for instance, cold started or single-staged versions of the motion planner. For that purpose, six different layouts of the motion planner were defined, which include every possible combination of the three stages (USLQ, PPWS+USLQ, CSLQ, PPWS+CSLQ, USLQ+CSLQ) and the complete approach (PPWS+USLQ+CSLQ = MWMP). For every layout, the same 21 motion plans were launched, using the PRL scenario with the ExoTeR model and the sample tube retrieval use case, changing the initial rover pose and the goal sample pose.

Three main parameters were measured within the tests. First, the success rate, as a percentage. This represents the ratio of finding a successful motion plan in the 21 tests, i.e.

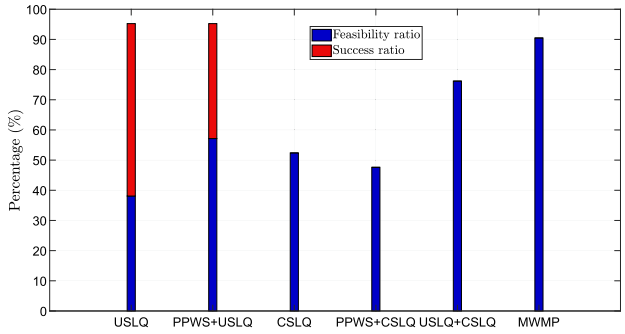

Fig. 4 Measured success and feasibility ratios on the performance tests. For every layout, 21 different motion plans were launched. Note that the layouts that include the constrained stage (CSLQ) have the same percentage of successful and feasible motion plans, since this stage ensures feasibility if the algorithm converges

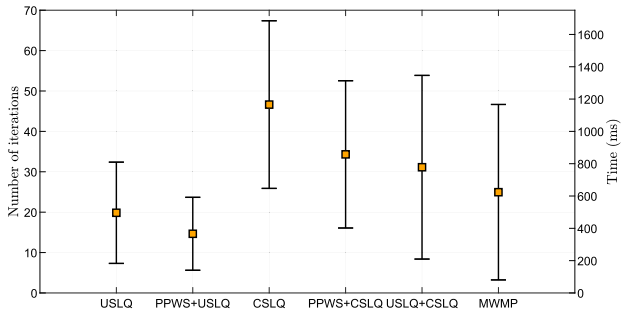

Fig. 5 Measured iterations and execution time on the performance tests. For every layout, 21 different motion plans were launched. Average measurements are shown, including the standard deviation on the samples

when the algorithm converges in less than 100 iterations, which is shown in Fig. 4. Second, the feasibility rate, which represents the percentage of constraint compliant motion plans in the 21 tests, which is also shown in Fig. 4. Clearly, the feasibility rate encompasses the success rate, given that a motion plan can only be feasible if the solver converges, i.e. if the motion plan is successful. Hence, every layout that makes use of the CSLQ stage has equal success and feasibility rates, since the algorithm only converges if the constraints are fulfilled. Third, the average number of iterations until convergence. If several stages are established, then the total number of iterations is used. The extracted results regarding the number of iterations are shown in Fig. 5. Note that the number of iterations is employed in the following as a measure of the convergence speed, since the computational time spent is not representative due to its dependency on external factors, such as the hardware, the quality of the software implementation or the CPU usage. Besides, the computational time spent is nearly proportional to the number of iterations, as an example, spending approximately $25ms$ per iteration in these tests, run on a single core of an Intel(R) Core(TM) i7-10750 H CPU (2.60 GHz).

4.3 Lab tests

Regarding the laboratory tests campaign, four of the most representative tests are analyzed in this paper, and one of them is shown in a summarizing video[3] of the lab tests campaign. Each test starts from a different rover location, also with a different pose of the sample tube. Additionally, an example of the evolution of the tests is also shown in Fig. 6.

The experiments were run as follows. First, it was assumed that the sample was inside ExoTeR's LocCam Field of View (FoV). Therefore, the sample detection and localization subsystem was launched at the beginning to provide the initial estimation of the sample pose. Then, the sample pose was translated into an end effector goal pose, just above the sample and approaching the ground perpendicularly, and this goal pose was fed to MWMP to compute a global initial motion plan. This was sent to the motion plan follower, which started to send the control commands at each time step, and receive the robot measured state. As explained in Sect. 3, the follower continuously checked if any of the controlled states was accumulating too much drift, given the defined thresholds. Then, if necessary, the motion was replanned using the last motion plan as a warm start to accelerate the computation. Additionally, the sample detection and localization subsystem was launched repeatedly with a frequency of 0.1 Hz, in order to keep improving the sample pose estimation and filtering the localization error due to the use of visual odometry, eventually triggering additional motion replans. Once the end effector had reached its goal pose, the execution was finished, and a separate sample retrieval component was launched just to perform the final sample grasping movement.

5 Discussion

The carried out experiments, both performance and laboratory tests, are examined in detail in the following. On the one hand, concerning the performance tests, their results are summarized in Figs. 4 and 5. As can be observed, the path planner warm start (PPWS+USLQ, PPWS+CSLQ, PPWS+USLQ+CSLQ) always reduces the average number of iterations w.r.t. the cold started versions (USLQ, CSLQ, USLQ+CSLQ), reducing also the convergence speed variability. This means that the motion planner behaviour is more predictable. Additionally, the fastest layouts are the unconstrained ones (USLQ, 19.85 it; PPWS+USLQ, 14.65 it), as expected and aforementioned. Although these layouts have really high success ratios (USLQ, 95.24%; PPWS+USLQ, 95.24%), they can not guarantee constraints compliance, thus they also have the lowest feasibility ratios (USLQ, 38.10%; PPWS+USLQ, 57.14%). The slowest layouts are the con-

[3] https://youtu.be/xDFv4Ho4KZs.

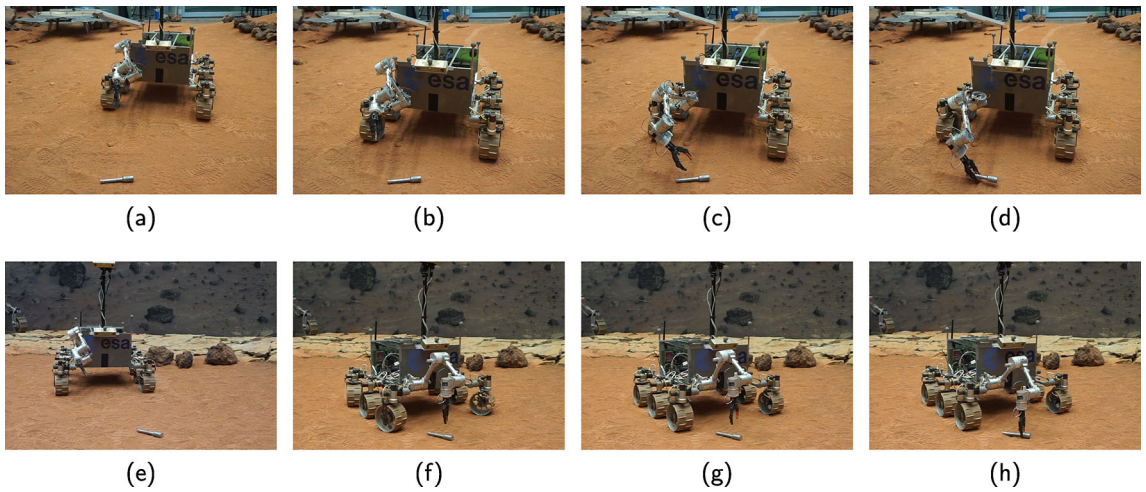

Fig. 6 Motion evolution of ExoTeR during two different sample tube retrieval tests, using a decoupled motion planning approach (**a**–**d**) or MWMP (**e**–**h**). The decoupled solution is not prepared to retrieve the sample once it reaches it (**b**), and cannot retrieve the sample perpendicularly to the ground (**c**), with the result of a defective grasp (**d**). On the other hand, MWMP generates an optimal motion, leaving the arm prepared to the retrieval operation as soon as the base stops (**f**), being it placed in a certain pose which allows the manipulator to retrieve the sample perpendicularly (**g**), with a high quality grasp (**h**)

strained ones (CSLQ, 46.645 it; PPWS+CSLQ, 34.31 it), and they still do not reach high feasibility ratios (CSLQ, 52.38%; PPWS+CSLQ, 47.62%), due to convergence difficulties considering the high over-actuation and number of constraints. Finally, the unconstrained-constrained layouts have an intermediate convergence speed (USLQ+CSLQ, 31.13 it; MWMP, 24.95 it), having the complete approach (MWMP) a comparable convergence speed to the unconstrained layouts. Besides, these layouts also have the highest feasibility ratios (USLQ+CSLQ, 76.19%; MWMP, 90.48%), thanks to the successive warm start procedure.

Summarizing, the performance tests demonstrate that the proposed multi-staged approach (MWMP) improves the behaviour of the optimal motion planner, increasing noticeably the average number of feasible motion plans (90.48%) maintaining a considerably low average number of iterations until convergence (24.95 it).

On the other hand, the results of four of the lab tests are summarized in Table 4. First, the initial motion plan number of iterations (20 it. avg) matches what it is expected, seeing the results of the performance tests (25 it. avg). Furthermore, the average number of iterations in each test case, i.e. mean of iterations required by the planner to converge in each particular test case considering the initial plan and the replans, is generally lower than the first plan iterations. This confirms that using the last motion plan as a warm start accelerates the motion planning procedure, although it did not happen in case 4, since a sharp turn was required and the steering joints were repeatedly reaching their limits. It is remarkable that the average errors w.r.t. the planned motion of the controlled states, i.e. the arm joints (0.3896° avg) and the steering joints (0.7105° avg), are negligible, which means that the predicted motion was accurate and the linearization errors do not severely impact the system behaviour, thanks to a sufficiently small time step Δt. Besides, most of the required replans were performed due to the low accuracy of the sample pose estimator (16 out of 25), which changed the goal pose substantially several times during the tests, as it can be observed in Table 4. Regarding the non directly controlled states, the errors of the rover base (0.0236 m, 0.6417° avg) and the end effector (0.0484 m, 13.3556° avg) final poses are also small, therefore, it is confirmed that the system model is representative despite of the linearization, and that the motion planner is accurate enough to ensure a successful sample retrieval, considering the 7 cm full opened gripper width. Note also that the end effector final pose error is caused mainly by the sample pose estimator, being minimal the errors induced by the motion planner itself.

Finally, the evolution of the Test Case 2 is shown in Fig. 6a–h, in comparison to another sample retrieval test in Fig. 6a–d, performed with a standard non-optimal and decoupled motion planning approach [31]. As can be observed, although both approaches start from similar situations (Fig. 6a and e), MWMP leaves the manipulator prepared for the retrieval operation as soon as the rover reaches the sample (Fig. 6f), meanwhile the decoupled solution yet requires to move the arm once the rover stops (Fig. 6b). Additionally, the decoupled solution does not place the rover base in a good position considering the posterior retrieval operation, thus, the gripper orientation is not perpendicular to the ground and does not match the sample yaw (Fig. 6c), which, in the end, generates a defective grasp (Fig. 6d). Conversely, the optimal motion planner uses all the system joints (rover and manipulator), placing the base to leave the arm in

Table 4 Lab test results

Sample tube retrieval test case	1	2	3	4
First plan number of iterations	17	20	29	13
Number of replans (after goal changed)	6 (6)	5 (3)	7 (2)	7 (5)
Average number of iterations in the test case	13.43	16.60	25.62	60.33
Average arm joints position error (°)	0.2521	0.2235	0.9167	0.1719
Average steering joints position error (°)	2.6471	0.0521	0.0997	0.0555
Rover base final pose error (m, °)	0.0023, 2.0798	0.036, 0.4183	0.02, 0.0	0.0361, 0.0630
End effector final pose error (m, °)	0.0942, 15.8996	0.0221, 12.4332	0.0441, 0.1833	0.0332, 24.9007
Sample pose estimation error (m, °)	0.0908, 15.7964	0.0150, 12.0035	0.0220, 0.6303	0.0030, 24.8893

a perfectly perpendicular pose w.r.t. the sample. Therefore, the gripper is orientated perpendicularly to the ground and matching the sample yaw (Fig. 6g), being the quality of the grasp, thus, much higher (Fig. 6h).

6 Conclusion

In this paper MWMP is presented, a motion planner for over-actuated mobile platforms capable of dealing with system dynamics and constraints, such as non-holonomic constraints or joints limits, without severely impacting the computational resources of the system. This is achieved by means of a multi-staged warm start approach, which initializes in several steps the optimal solver, SLQ. In particular, a novel pipeline with three different stages is used: first, a FMM-based path planner; second, an unconstrained SLQ motion planner; third, a constrained SLQ motion planner. This complete approach has been demonstrated to improve the performance of the motion planner in comparison with any other combination of the stages, since the algorithm converges faster and the probability of finding a feasible solution is the highest (up to twice as fast and 40% more feasible solutions in comparison with the standard Constrained SLQ).

Furthermore, a generic state space model for over-actuated mobile platforms has been presented, which can model platforms composed of several kinematic chains in a straightforward way. This model is particularized for the ExoTeR rover, composed by a mobile base and a robotic arm, which is later used to perform some laboratory tests to validate the motion planner. For that purpose, a tailored event-triggered replanning capability has been included, which allows the system to precisely follow the generated motion plans. The performed laboratory tests emulate a martian sample tube retrieval mission, showcasing the advantages of the presented motion planner to generate accurate motions for over-actuated and constrained platforms, in this case, allowing ExoTeR to retrieve a sample tube with a high quality grasp, even considering the rover and sample localization errors.

The proposed multi-staged warm start sequence has demonstrated to improve the motion planner convergence speed and the feasibility of its solutions, nevertheless, it has only been tested with the SLQ solver. Testing if the proposed sequence also boosts other optimization solvers is planned as future work. Furthermore, as done in related works, the inclusion of the constraints sequentially inside the CSLQ stage can boost even further the convergence speed specially for highly constrained and cluttered use cases, which remains to be tested. Finally, it is expected the use of this motion planner in further use cases, including 3D platforms with faster dynamics and multiple external disturbances. Thus, a benchmark of the proposed motion planner and the replanning capability with other methodologies, such as existing MPC controllers, is also planned as future work.

Funding This work has been partially funded by the EU-H2020 project entitled "Cooperative Robots for Extreme Environments" (CoRob-X) under grant agreement: 101004130. Funding for open access charge: Universidad de Málaga / CBUA".

References

1. Latombe J-C (1991) Robot motion planning. Springer, Boston. https://doi.org/10.1007/978-1-4615-4022-9
2. Gerdes L, Azkarate M, Sánchez-Ibáñez JR, Joudrier L, Perez-del-Pulgar CJ (2020) Efficient autonomous navigation for planetary rovers with limited resources. J Field Robot 37(7):1153–1170. https://doi.org/10.1002/rob.21981

3. Araguz C, Bou-Balust E, Alarcón E (2018) Applying autonomy to distributed satellite systems: trends, challenges, and future prospects. Syst Eng 21(5):401–416. https://doi.org/10.1002/SYS.21428
4. Kratky V, Alcantara A, Capitan J, Stepan P, Saska M, Ollero A (2021) Autonomous aerial filming with distributed lighting by a team of unmanned aerial vehicles. IEEE Robot Autom Lett 6(4):7580–7587. https://doi.org/10.1109/LRA.2021.3098811
5. Huang SW, Chen E, Guo J (2018) Efficient seafloor classification and submarine cable route design using an autonomous underwater vehicle. IEEE J Ocean Eng 43(1):7–18. https://doi.org/10.1109/JOE.2017.2686558
6. St Pierre R, Bergbreiter S (2019) Toward autonomy in sub-gram terrestrial robots. Ann Rev Control Robot Auton Syst 2:231–52
7. Li W, Todorov E (2004) Iterative linear quadratic regulator design for nonlinear biological movement systems. In: 1st international conference on informatics in control, automation and robotics (ICINCO), pp 222–229
8. Sideris A, Bobrow JE (2005) An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems. IEEE Trans Autom Control 50(12):2043–2047. https://doi.org/10.1109/TAC.2005.860248
9. Todorov E, Li W (2005) A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In: Proceedings—American control conference, vol 1, pp 300–306. https://doi.org/10.1109/ACC.2005.1469949
10. Sideris A, Rodriguez LA (2011) A Riccati approach for constrained linear quadratic optimal control. Int J Control 84(2):370–380. https://doi.org/10.1080/00207179.2011.555883
11. Shirai Y, Lin X, Mehta A, Hong D (2021) LTO: lazy trajectory optimization with graph-search planning for high dof robots in cluttered environments. In: Proceedings—IEEE international conference on robotics and automation, pp 7533–7539 . https://doi.org/10.1109/ICRA48506.2021.9561502
12. Jiao Z, Zhang Z, Jiang X, Han D, Zhu SC, Zhu Y, Liu H (2021) Consolidating kinematic models to promote coordinated mobile manipulations. In: Proceedings—IEEE International conference on intelligent robots and systems, pp 979–985. https://doi.org/10.1109/IROS51168.2021.9636351
13. Giftthaler M, Farshidian F, Sandy T, Stadelmann L, Buchli J (2017) Efficient kinematic planning for mobile manipulators with non-holonomic constraints using optimal control. In: Proceedings—IEEE international conference on robotics and automation, pp 3411–3417 . https://doi.org/10.1109/ICRA.2017.7989388
14. Plancher B, Neuman SM, Bourgeat T, Kuindersma S, Devadas S, Reddi VJ (2021) Accelerating robot dynamics gradients on a CPU, GPU, and FPGA. IEEE Robot Autom Lett 6(2):2335–2342. https://doi.org/10.1109/LRA.2021.3057845
15. Lembono TS, Paolillo A, Pignat E, Calinon S (2020) Memory of motion for warm-starting trajectory optimization. IEEE Robot Autom Lett 5(2):2594–2601. https://doi.org/10.1109/LRA.2020.2972893
16. Thakar S, Rajendran P, Annem V, Kabir A, Gupta S (2019) Accounting for part pose estimation uncertainties during trajectory generation for part pick-up using mobile manipulators. In: Proceedings—IEEE international conference on robotics and automation, pp 1329–1336 . https://doi.org/10.1109/ICRA.2019.8793501
17. Kabir AM, Thakar S, Malhan RK, Shembekar AV, Shah BC, Gupta SK (2021) Generation of synchronized configuration space trajectories with workspace path constraints for an ensemble of robots. Int J Robot Res 40:651–678. https://doi.org/10.1177/0278364920988087/ASSET/IMAGES/LARGE/10.1177_0278364920988087-FIG2.JPEG
18. Thakar S, Malhan RK, Bhatt PM, Gupta SK (2022) Area-coverage planning for spray-based surface disinfection with a mobile manipulator. Robot Auton Syst. https://doi.org/10.1016/J.ROBOT.2021.103920
19. Luis CE, Vukosavljev M, Schoellig AP (2020) Online trajectory generation with distributed model predictive control for multi-robot motion planning. IEEE Robot Autom Lett 5(2):604–611. https://doi.org/10.1109/LRA.2020.2964159
20. Sethian JA (1999) Fast marching methods. SIAM Rev 41(2):199–235. https://doi.org/10.1137/S0036144598347059
21. Valero-Gomez A, Gomez JV, Garrido S, Moreno L (2013) The path to efficiency: fast marching method for safer, more efficient mobile robot trajectories. IEEE Robot Autom Mag 20(4):111–120. https://doi.org/10.1109/MRA.2013.2248309
22. Liu Y, Bucknall R (2015) Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment. Ocean Eng 97:126–144. https://doi.org/10.1016/j.oceaneng.2015.01.008
23. Sánchez-Ibánez JR, Pérez-del-Pulgar CJ, Azkarate M, Gerdes L, García-Cerezo A (2019) Dynamic path planning for reconfigurable rovers using a multi-layered grid. Eng Appl Artif Intell 86:32–42. https://doi.org/10.1016/j.engappai.2019.08.011
24. Bajracharya M, Maimone MW, Helmick D (2008) Autonomy for Mars Rovers: past, present, and future. Computer 41(12):44–50. https://doi.org/10.1109/MC.2008.479
25. Merlo A, Larranaga J, Falkner P (2013) Sample fetching Rover (SFR) for MSR. In: ASTRA 2013—12h symposium on advanced space technologies for robotics and automation. European Space Agency (ESA)
26. Muirhead BK, Karp A (2019) Mars sample return lander mission concepts. In: Proceedings—IEEE aerospace conference. https://doi.org/10.1109/AERO.2019.8742215
27. Azkarate M, Gerdes L, Wiese T, Zwick M, Pagnamenta M, Hidalgo Carrio J, Poulakis P, Perez-del-Pulgar C (2022) Design, testing, and evolution of Mars Rover testbeds: European space agency planetary exploration. IEEE Robot Autom Mag. https://doi.org/10.1109/MRA.2021.3134875
28. Castilla-Arquillo R, Perez-del-Pulgar CJ, Paz-Delgado GJ, Gerdes L (2022) Hardware-accelerated mars sample localization via deep transfer learning from photorealistic simulations. IEEE Robot Autom Lett 100:1–8. https://doi.org/10.1109/LRA.2022.3219306
29. Geiger A, Ziegler J, Stiller C (2011) StereoScan: dense 3d reconstruction in real-time. In: Proceedings—IEEE intelligent vehicles symposium, pp 963–968. https://doi.org/10.1109/IVS.2011.5940405
30. Paz-Delgado GJ, Azkarate M, Sanchez-Ibanez JR, Perez-Del-Pulgar CJ, Gerdes L, Garcia-Cerezo AJ (2020) Improving autonomous rover guidance in round-trip missions using a dynamic cost map. In: Proceedings—IEEE international conference on intelligent robots and systems.https://doi.org/10.1109/IROS45743.2020.9340912
31. Mantoani LM, Castilla-Arquillo R, Paz-Delgado GJ, Pérez-Del-Pulgar CJ, Azkarate M (2022) Samples detection and retrieval for a sample fetch Rover. In: 16th symposium on advanced space technologies in robotics and automation (ASTRA), Noordwijk, The Netherlands