Merging global and local planners: real-time replanning algorithm of redundant robots within a task-priority framework

Paolo Di Lillo, *Member, IEEE,* Daniele Di Vito, *Member, IEEE,* Gianluca Antonelli, *Fellow, IEEE*

***Abstract*—Task-priority inverse kinematics is a popular motion control algorithm which efficiently handles redundancy in robot manipulators. It has been recently extended in order to handle also set-based control objectives or inequality constraints. As any local motion planner it is prone to the occurrence of local minima. This work further extends set-based inverse kinematics by adding a motion planner in order to avoid such occurrence. Motion planners are usually computationally heavy especially in their eventual implementation with a task-priority architecture. To reduce this issue, the planner is implemented as a sampling-based algorithm which works in the reduced-dimensionality of the robot workspace applying Cartesian constraints only. The output trajectory is then checked against the inverse kinematics algorithm exploiting the redundancy and verifying the fulfillment of the joint-based task constraints. During the motion, inverse kinematics is then used also in real-time to ensure a reactive behavior to address, e.g., mismatch between the a-priori information and real-time perception acquisition. Also, the motion planner runs in background to adapt to changes in the environment or to accommodate incremental mapping. Comparison with alternative approaches are investigated and discussed. The most promising method is validated first in hundreds of numerical simulations to provide a solid statistical analysis and then experimentally with a Kinova Jaco2 7 DOFs manipulator equipped with an RGB-D sensor.**

***Note to Practitioners*—In this work we propose a motion planning algorithm that allows to effectively deploy robot manipulators in partially unstructured environments. It is structured in two layers: a Cartesian space global planner guarantees the feasibility of the trajectory with respect to potential obstacles in the environment, while a reactive local planner checks the feasibility at joint level. This architecture allows to define all the safety constraints both at Cartesian and joint space needed to operate in unstructured environments, while keeping the computation time lower with respect to standard approaches that define all the constraints in an offline global planner. The reduced computation time allows to effectively perform real-time replanning operations when needed, making the robotic system capable of adapting to a dynamic environment and suitable for sharing its workspace with human operators.**

***Index Terms*—Path planning, motion planning, set-based task priority inverse kinematics, redundant robots.**

I. Introduction

Nowadays robots moved out from the cells that separate them from human operators, to reach the desks and share their workspace with humans. Collaborative robots (cobots) [1] can be used in several scenarios, i.e., industrial, assistive and medical applications [2], [3], [4], [5], [6]. However, differently from the *standard* industrial robotics where any action is properly pre-planned in a highly-structured environment, moving in a dynamic or only partially known environment requires real-time trajectory generation. In addition, the actions of the human operators are not known in advance and the robotic system has to take into account its own constraints such as joint limits, while guaranteeing simultaneously the human safety which is obviously a critical requirement.

As well-known in robotics, robot control objectives such as a pick and place command, can be embedded in an inverse kinematics algorithm [7], [8]. The latter is a local motion controller that can be extended through the task-priority inverse kinematics framework allowing to manage multiple tasks on redundant robots [9], [10], [11], [12], [13]. Further extensions of this local controller have been recently proposed for controlling set-based tasks, also known in literature as inequality constraints [14], [15], [16], [17], [18], [19] with also recent practical implementations such as spray painting [20].

Inverse kinematics control algorithms take into account both system and environment constraints reacting and managing environmental changes in real-time. Nevertheless, being a local method, during the execution of the desired task, the robot can get trapped in local minima.

Global planners based on task-constrained motion planning [21] have been proposed in literature also to overcome such issue. Some proposed global planners work in the joint space [22] requiring, then, particular projection approaches [23], [24] to find a system configuration satisfying all the constraints. Indeed, a proper task constraints representation is not straightforward since the mapping between the joint and task space is not linear. Thus, other global planners working directly in the task space have been proposed [25].

Planning and control tasks are usually treated as two independent problems. An improvement can be obtained by implementing a control-aware motion planning algorithm as in [26]. This is based on a strict coupling between planning and control which takes into account the closed-loop system dynamics. The aim is to close as much as possible the gap between the planning and the real trajectory execution.

The mentioned global planners own the drawback of usually exhibiting a high computational load. This is critical when the problem is addressed by taking into account all the constraints in a complex environment. Indeed, global planners are usually used offline since they are not able to perform real-time replanning in a dynamic environment which is mandatory to guarantee human safety in a shared workspace.

Some methods in literature have been designed to perform replanning operations in real-time, in particular within dynamic environments shared with human operators. The work in [27] discusses a real-time Model Predictive Control able to deal with dynamic and unknown scenarios including motion constraints, i.e., acceleration, velocity and position constraints.

P. Di Lillo, D. Di Vito and G. Antonelli are with the Department of Electrical and Information Engineering of the University of Cassino and Southern Lazio, Via Di Biasio 43, 03043 Cassino (FR), Italy {pa.dilillo, d.divito, antonelli}@unicas.it

The work in [28] proposes a control-based motion planning that, given a desired task path, exploits the robot redundancy planning in the task-constrained configuration space. The same control-based planner is extended in [29] in order to plan safe cyclic motions under repetitive task constraints. It is further extended in in [30] for planning the motion along the known task path in the presence of obstacles moving along known trajectories.

In addition to algorithmic methods, great efforts have been made at the hardware level as well. Indeed, the work in [31] describes the process to build a robot-specific circuitry for motion planning. In particular, the circuits are implemented on a Field-Programmable Gate Array (FPGA), which is able to plan for a 6-DOF Jaco robot manufactured by Kinova in 1 ms.

In this work the extension of a task-priority-based local planner by properly merging it with a global one is presented. The global planner generates a feasible trajectory being aware of the constraints included in the local planner. Both of them run also in real-time, the local one to properly handle a dynamic environment and incremental maps, the global one in background to always optimize the path. More in detail, the global planner is structured in two steps. The motion planner is the first one, in which a sampling-based algorithm that plans in the more intuitive and low-dimensional Cartesian space takes into account potential obstacles in the scene. When a feasible Cartesian path is found, a trajectory is generated and then checked in the second step of the global planner. Indeed, the second step consists of simulating the local motion control by tracking the candidate trajectory in output from the motion planner. In particular, the Set-based Task-Priority Inverse Kinematics (STPIK) algorithm is used as local motion control. This inverse kinematics method handles the equality and set-based, also known in literature as inequality constraints, verifying the trajectory feasibility both in Cartesian and joint space. The feasible trajectory is then sent to the local motion controller which, being based on the same inverse kinematics algorithm used for the trajectory check, assures its tracking in case of a static and perfectly known environment. Nevertheless, during the robot motion, the global planner continues to run in background trying to re-plan in case of a dynamic environment.

The proposed method, preliminary presented in [32], being based on the combination of a sampling-based planning algorithm as global planner and the STPIK algorithm as local planner, is able to perform real-time replanning for redundant robots. Indeed, it keeps the key-properties of both approaches. On one hand, the typical local minima problem of local methods is avoided thanks to the global planner. On the other hand, the typical high computational load of global planners is lightened by planning in the low dimensional Cartesian space instead of the joint space. Furthermore, the local planner, being a reactive method, guarantees the constraints fulfillment during the real-time global replanning.

The developed method is validated first in hundreds of numerical simulations to provide a solid statistical analysis and then experimentally with a Kinova Jaco2 7 DOFs manipulator equipped with an RGB-D camera.

The rest of the paper is organized as follows: in Sect. II the control architecture is described and details on the global/local planners are given; Section III shows the results of an extensive experimental campaign aimed at evaluating the performances of the proposed algorithm in the presence of static and dynamic obstacles in the scene; in Sec IV we discuss algorithmic insights and alternative implementations, while Sec. V presents conclusions and future works.

II. Control architecture

The proposed control architecture is based on a combination of a global and local planner. In detail, the global planner generates a trajectory in position and orientation $\boldsymbol{\eta}_{\text{ee,r}}(t)$, fulfilling all constraints both in the Cartesian and joint space, given a desired target constant $\boldsymbol{\eta}_{\text{ee,d}}$ in the Cartesian space for the manipulator end-effector and taking into consideration the environmental a-priori knowledge, the robot constraints and the feedback from the perception module. The generated trajectory is sent to the local planner which computes the reference system velocity $\dot{\boldsymbol{q}}$ needed for tracking the desired trajectory, handling the information from the perception module through a specific inverse kinematics framework able to manage both equality and set-based tasks. It is worth noticing that the same inverse kinematics algorithm is used inside the global planner for checking the feasibility of $\boldsymbol{\eta}_{\text{ee,r}}(t)$. This guarantees that the local motion control tracks it correctly in case of static and known environment.

The global planner continues running in background during the motion. Indeed, in case of mismatch between the a-priori environmental information and the one acquired by the perception module, it re-plans the trajectory generating a new $\boldsymbol{\eta}_{\text{ee,r}}(t)$ compatible with the changes in the configuration space, e.g., moving obstacles, human activity or incremental map building. The global planner is completely asynchronous with respect to the rest of the control algorithm architecture. In this way, the global planner is able to adapt to changes while the trajectory tracking fulfilling all the constraints is guaranteed by the local controller running in real-time with a fixed sampling time.

The overall control architecture is shown in Fig. 1 whereas the algorithmic approach is reported in Algorithm 1.

Algorithm 1: `Planning`$(\boldsymbol{\eta}_{\text{ee,d}}, \boldsymbol{q}, \text{knowledge}, \text{perception})$

The Global Planner generates a feasible trajectory, that is sent to the Local Planner. At each timestep, the latter computes the joint velocities that makes the end-effector follow the reference trajectory.

$[\textit{SUCCESS}, \boldsymbol{\eta}_{\text{ee,r}}(t)] \leftarrow$ `GlobalPlanner`$(\boldsymbol{\eta}_{\text{ee,d}}, \text{knowledge}, \text{perception})$;
if *SUCCESS* **then**
  $k = 0$;
  while $k < \textit{length}(\boldsymbol{\eta}_{ee,r}(t))$ **do**
    $\dot{\boldsymbol{q}} \leftarrow$ `LocalPlanner`$(\boldsymbol{\eta}_{\text{ee,r}}(t_k), \boldsymbol{q}, \text{perception})$;
    $k = k + 1$;

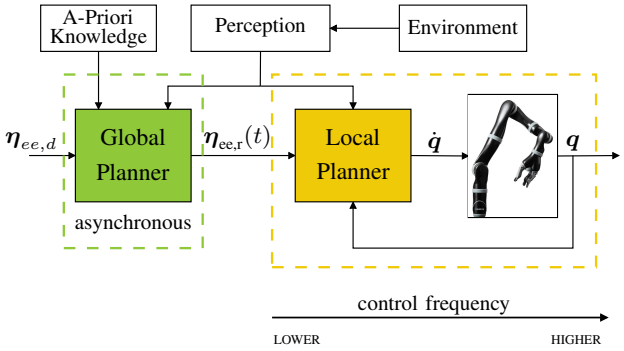

Fig. 1: Control architecture: given a desired target $\eta_{ee,d}$ in Cartesian space, the *global planner*, considering the feedback from the *perception* module, generates a trajectory $\eta_{ee,r}(t)$ each time a re-planning is performed, fulfilling all tasks constraints both in Cartesian and joint space; then, the *local planner* computes the reference velocity $\dot{q}$ for the robotic system. In particular, the local planner (yellow dashed box) is synchronous with the low level control while the global planner (green dashed box) is asynchonous.

A. Local Planner

The local planner receives the reference trajectory $\eta_{ee,r}(t)$ for the end-effector,specified both in translation and orientation, from the global planner. Its output is the reference system velocity $\dot{q}$ needed for tracking $\eta_{ee,r}(t)$ while taking into account Cartesian and joint space constraints. To this aim, the Set-based Task-Priority Inverse Kinematics (STPIK) framework is used in this step of the proposed approach.

Indeed, it is the core of the local planner as shown in Fig. 2 and in Algorithm 2.

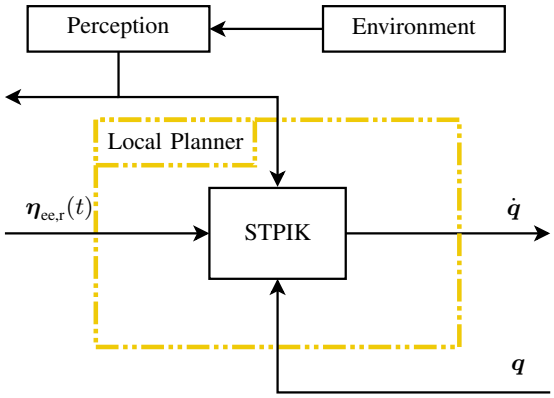

Fig. 2: Local Planner: it is represented by the Set-based Inverse Kinematics (STPIK) framework which computes the robotic system velocity $\dot{q}$ necessary for the reference trajectory $\eta_{ee,r}(t)$ tracking simultaneously fulfilling the Cartesian/joint-space constraints.

Algorithm 2: `LocalPlanner`$(\eta_{ee,r}(t_k), q,$ perception$)$

At each timestep, the Local Planner receives the current pose sample to follow and computes the related joint velocities.

1 $\dot{q}, q \leftarrow$ `STPIK`$(\eta_{ee,r}(t_k), q,$ perception$)$;
2 $\eta_{ee} \leftarrow$ `computeState`(q);

For a generic robotic manipulator system with n DOFs and, therefore, being $q = [q_1 \dots q_n]^T \in \mathbb{R}^n$ its joint space, in addition to its end-effector pose $\eta_{ee} = [p_{ee} \quad \mathcal{Q}_{ee}]^T \in \mathbb{R}^7$, where p_{ee} is the vector stacking its 3D position $\mathcal{Q}_{ee}$ is the unit quaternion representing its orientation [33], there are several tasks that can be controlled, e.g., obstacle avoidance, joint limits and virtual walls [8]. More in detail, the relation between a generic task function $\sigma_x \in \mathbb{R}^m$ and the manipulator joints can be described by the function

$$\sigma_x(q) = f_x(q) \tag{1}$$

whose time derivative is

$$\dot{\sigma}_x(q) = J_x(q)\dot{q} \ , \tag{2}$$

where $J_x(q) \in \mathbb{R}^{m \times n}$ is the task Jacobian matrix and $\dot{q}$ is the vector of joint velocities. In order to achieve a desired task value $\sigma_{x,d}$, the joint velocities can be computed through the Closed Loop Inverse Kinematics (CLIK) algorithm [10]:

$$\dot{q} = J_x^{\dagger}(\dot{\sigma}_{x,d} + K_x\tilde{\sigma}_x) \ , \tag{3}$$

where $J_x^{\dagger}$ is the Moore-Penrose pseudoinverse of the task Jacobian matrix, $\dot{\sigma}_{x,d} \in \mathbb{R}^m$ is the time derivative of $\sigma_{x,d}$, $K_x \in \mathbb{R}^{m \times m}$ is a positive-definite gain matrix and $\tilde{\sigma}_x = \sigma_{x,d} - \sigma_x$ is the task error.

When the manipulator system is redundant, namely the system DOFs number n is greater than the task dimension m, multiple tasks can be accomplished simultaneously by exploiting the orthogonal complement $\mathcal{N}(J) \neq \emptyset$ of the subspace $\mathcal{R}(J)$. Given h tasks, they can be combined in a hierarchy $\mathcal{H}$ with h levels of priority, sorted from the highest priority task ($i = 1$) to the lowest one ($i = h$). In particular, the velocity components corresponding to a low-priority task are projected onto the null space of higher priority ones, i.e., the primary task fulfillment is always guaranteed while the lower priority ones are accomplished *at best* [34]. Therefore, given h tasks, the manipulator system velocity $\dot{q}$ can be recursively computed as [35]:

$$\dot{q}_h = \sum_{i=1}^{h} (J_i N_{i-1}^A)^{\dagger}(\dot{\sigma}_{i,d} K_i \tilde{\sigma}_i - J_i \dot{q}_{i-1}) \ , \tag{4}$$

where

$$N_i^A = I - (J_i^A)^{\dagger} J_i^A \ , \tag{5}$$

is the null space projector of the augmented Jacobian matrix

$$J_i^A = \begin{bmatrix} J_1^T & J_2^T & \dots & J_i^T \end{bmatrix}^T \ , \tag{6}$$

obtained by stacking all the task Jacobian matrices from task 1 to i.

Projecting any joint velocity $\dot{q}_l$, with $l > i$, onto the null-space of the augmented Jacobian matrix guarantees that the fulfillment of all the tasks from 1 to i is not affected by $\dot{q}_l$.

Equation (4) represents one of the implementations of the task-priority inverse kinematics algorithm that can handle multiple tasks simultaneously. However, the latter can be only equality-based tasks meaning that the control objectives are characterized by specific desired values. In addition, there exist control objectives the values of which can vary inside a range

such as the virtual walls, the obstacle avoidance, the joint limits and self-collision avoidance. Such control objectives are defined as set-based tasks and need to be properly handled within the inverse kinematics framework [36]. This is obtained by considering set-based tasks as equality-based ones with the addition of lower and upper thresholds. The task gets added or removed from the hierarchy following a dedicated algorithm, depending on the value that the function takes during the motion. More details about the specific employed algorithm can be found in [37].

It is worth remarking that, in general, inverse kinematics approaches are influenced by the kinematic singularities [7]. Nevertheless, in order to overcome the singularity problem, the damped pseudoinverse [38] is used in Eq. (4) instead of the standard one:

$$\boldsymbol{J}^{\dagger^{\lambda}} = \boldsymbol{J}(\boldsymbol{q})^{\mathrm{T}}\Big(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^{\mathrm{T}} + \lambda^2 \boldsymbol{I}\Big)^{-1}, \tag{7}$$

where λ is the damping factor weighting the accuracy of tracking the task error with respect to the norm of the joint velocities and $\boldsymbol{I} \in \mathbb{R}^{m\times m}$ is an identity matrix.

B. Global Planner

The global planner receives the desired Cartesian pose $\boldsymbol{\eta}_{\text{ee,d}}$ for the end-effector and it generates a feasible trajectory $\boldsymbol{\eta}_{\text{ee,r}}(t)$ taking into consideration the a-priori knowledge and the information on the space configuration coming from the perception module. In particular, it is structured in two steps, as shown in Fig. 3: 1) the motion planner that is in charge of generating $\boldsymbol{\eta}_{\text{ee,r}}(t)$ satisfying the sole Cartesian constraints (e.g. obstacle avoidance); 2) the STPIK-check which verifies if $\boldsymbol{\eta}_{\text{ee,r}}(t)$ fullfills both Cartesian and joint space constraints (e.g. joint limits, self-collisions, etc.) guaranteeing, then, the local control tracking, being the latter based on the same inverse kinematics algorithm. It is worth noticing that the motion planner computation load is lightened, due to the fact that it has to take into account only Cartesian constraints, while all the joint constraints are handled by the STPIK-check. This task separation allows to considerably reduce the overall execution time of the planner, making it suitable also for online replanning operations. The motion planner is implemented with a sampling-based algorithm [39] whose general template is shown in Algorithm 3. More in detail, the single algorithmic steps are:

1) **Init:** let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a non-directed search graph where V is the set of vertexes containing the start $\boldsymbol{\eta}_{\text{ee,0}}$ and desired pose $\boldsymbol{\eta}_{\text{ee,d}}$, respectively, and E the set of edges which is empty.
2) **Vertex selection:** a vertex $\boldsymbol{\eta}_{\text{ee}} \in V$ is selected for the graph expansion.
3) **Vertexes connection:** let $\mathcal{C}_{\text{free}}$ be the Cartesian space free from obstacles and human cooperators, then, taking into consideration a $\boldsymbol{\eta}_{\text{ee,new}} \in \mathcal{C}_{\text{free}}$, an attempt to implement a collision-free path segment p is done; if the attempt fails then it is necessary to re-do the step 2).
4) **Edge insertion into the graph:** the path segment p from the previous step is added to E as an edge from $\boldsymbol{\eta}_{\text{ee}}$ to $\boldsymbol{\eta}_{\text{ee,new}}$; the latter is added to V if not already present.

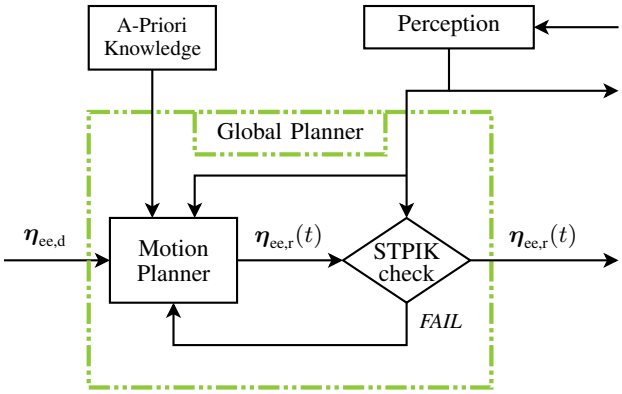

Fig. 3: Global Planner structured in two steps: 1) the motion planner is a sampling-based algorithm which computes the trajectory $\boldsymbol{\eta}_{\text{ee,r}}(t)$ to reach the desired target $\boldsymbol{\eta}_{\text{ee,d}}$ taking into consideration the a-priori knowledge and the information from the perception module; 2) the STPIK-check, through the set-based task-priority inverse kinematics framework, verifies the trajectory tracking and the constraints fulfillment

Algorithm 3: `SamplingAlgorithm`$(\boldsymbol{\eta}_{\text{ee,0}}, \boldsymbol{\eta}_{\text{ee,d}}, \text{knowledge})$

The sampling algorithm generates an obstacle-free path for the end-effector based on the a-priori knowledge of the obstacles position.

$\mathcal{G}$.`init`$(\boldsymbol{\eta}_{\text{ee,0}}, \boldsymbol{\eta}_{\text{ee,d}})$;
while *FAIL* **do**
  $\boldsymbol{\eta}_{\text{ee}} \leftarrow$ `vertexSelection`$(\mathcal{G})$;
  if `vertexesConnections`$(\boldsymbol{\eta}_{ee})$ **then**
    $[(\boldsymbol{\eta}_{\text{new}}, \boldsymbol{\eta}_{\text{ee}})] \leftarrow$ `vertexesConnections`$(\boldsymbol{\eta}_{\text{ee}})$;
  else
    FAIL $\leftarrow$ `vertexesConnections`$(\boldsymbol{\eta}_{\text{ee}})$;
    continue `// restart the loop`;
  $\mathcal{G}$.`edgeInsertion`$(\boldsymbol{\eta}_{\text{new}}, \boldsymbol{\eta}_{\text{ee}})$;
  $[\boldsymbol{s}_{\text{path}}, \textit{FAIL / SUCCESS}] \leftarrow$ $\mathcal{G}$.`checkSolutionPath`$()$;
return $\boldsymbol{s}_{\text{path}}$;

5) **Check for a solution path:** a check on the graph $\mathcal{G}$ is performed for finding a global path connecting the start point $\boldsymbol{\eta}_{\text{ee,0}}$ to the desired target $\boldsymbol{\eta}_{\text{ee,d}}$; if a solution path is not found, the procedure is iterated from step 2) unless a global path has been found or some termination condition is satisfied; in the latter case the algorithm reports a failure.

The sampling is performed in the low-dimensional Cartesian space. Then, the computational load is lower and the constraint representation, e.g., obstacles, self-collisions, virtual and physical walls, is simplified.

Several sampling-based planning algorithms can be found in literature. In the specific case, the Rapidly-exploring Random Tree (RRT) [39] is used.

At each iteration a configuration $\boldsymbol{\eta}_{\text{rand}}$ is sampled at random. Then, $\boldsymbol{\eta}_{\text{near}}$, the nearest configuration to $\boldsymbol{\eta}_{\text{rand}}$ in the graph, is found and extended in the direction of $\boldsymbol{\eta}_{\text{rand}}$ to a new configu-

ration $\boldsymbol{\eta}_{\text{new}}$. If the path between $\boldsymbol{\eta}_{\text{near}}$ and $\boldsymbol{\eta}_{\text{new}}$ is collision-free, the vertex $\boldsymbol{\eta}_{\text{new}}$ is added to the graph, together with the edge connecting $\boldsymbol{\eta}_{\text{near}}$ and $\boldsymbol{\eta}_{\text{new}}$. Nevertheless, alternative sampling-based planning algorithms, such as RRT Connect [40] may be implemented.

The collision-free check during the graph construction is performed taking into consideration the a-priori knowledge of the space configuration and the information coming from the perception module. When a collision-free global path $\boldsymbol{s}_{\text{path}}$ from $\boldsymbol{\eta}_{\text{ee,0}}$ to $\boldsymbol{\eta}_{\text{ee,d}}$ is found by the sampling-based exploring algorithm, it is processed for generating a smoothed reference trajectory $\boldsymbol{\eta}_{\text{ee,r}}(t)$. Within this aim, first an attempt to reduce the way-points of $\boldsymbol{s}_{\text{path}}$ is done through a short-cutting iterative operation.

The short-cutting operation attempts to remove vertices from a path while keeping it path valid. A connection is attempted between non-consecutive way-points on the path and, if the connection is successful, it is shortened by removing the in-between way-points. In case that the Cartesian space constraints are not satisfied anymore, the previous way points are restored.

After the short-cutting operation, the path is smoothed applying the B-splines approach [41], having better approximation capabilities compared to polynomial methods. The resulting smoothed path is finally sampled with a specific desired sampling time T_s to generate the reference trajectory $\boldsymbol{\eta}_{\text{ee,r}}(t)$ for the local planner.

The trajectory $\boldsymbol{\eta}_{\text{ee,r}}(t)$ computed by the motion planner in the first step is then sent to the second step of the Global Planner that is the STPIK-check. In particular, during this phase, the reference trajectory tracking is simulated through the STPIK algorithm. This simulation, which takes into consideration the information from the perception module as well as the motion planner, allows to check if the manipulator is able to respect both Cartesian and joint space constraints while guaranteeing the trajectory tracking. A failure condition that depends on the tracking error $\boldsymbol{e}(t) = \boldsymbol{\eta}_{\text{ee,r}}(t) - \boldsymbol{\eta}_{\text{ee}}(t)$ has been defined. Indeed, if during the simulation the end effector deviates from $\boldsymbol{\eta}_{\text{ee,r}}(t)$ of a predefined *failure threshold* M, i.e. if $||\boldsymbol{e}(t_i)|| > M$ for some i, then the STPIK-check process stops and sends a *fail* signal to the motion planner which generates a new $\boldsymbol{\eta}_{\text{ee,r}}(t)$. The loop terminates when a candidate trajectory $\boldsymbol{\eta}_{\text{ee,r}}(t)$ overcomes the STPIK-check or when the number of failures exceeds a predefined threshold $F_{\max}$.

The pseudo-code is reported in Algorithm 4.

As already mentioned, the use of the same IK algorithm both in the global planner and in the local one ensures that the local controller can actually accomplish the constrained trajectory in real-time in case of static and known environment. However, replanning is necessary if space configuration changes, e.g., due to the human presence or moving obstacles.

C. Online re-planning

The global planner works also online during the trajectory tracking through the local planner as a parallel process. More in detail, the motion planner runs in background acquiring the information coming from the perception module and

Algorithm 4: `GlobalPlanner`($\boldsymbol{\eta}_{\text{ee,d}}$, knowledge, perception)

A motion planner generates a tentative trajectory that is then checked by the inverse kinematics. This loop ends when a candidate trajectory overcomes the STPIK-check or when the number of failures exceeds a predefined threshold.

```
failCount=0;
while η_ee ! = η_ee,d do
    while FAIL do
        η_ee,r(t) ←
          MotionPlanner(η_ee,0, η_ee,d, knowledge)
        [η_ee,r(t), FAIL / SUCCESS] ←
          STPIKcheck(η_ee,r(t))
        if FAIL then
            failCount = failCount + 1
        if failCount == F_max then
            return FAIL;
        η_ee ← computeState(q);
return [SUCCESS, η_ee,r(t)];
```

continuously checks if the current trajectory $\boldsymbol{\eta}_{\text{ee,r}}(t)$ still fulfills Cartesian constraints. A replanning operation is performed in case of a mismatch between the environmental information acquired at the time of the generation of $\boldsymbol{\eta}_{\text{ee,r}}(t)$ and the current one. In particular, let us assume that the planning operation takes in average ΔT s, depending especially on the robotic system, including the hardware where the algorithm runs, and the environments conditions, e.g., how much it is cluttered. Then, if a constraint is violated (e.g. an obstacle moves along the planned path) at, e.g., $t = t_1$, the motion planner uses the reference robot state at $t_2 = t_1 + \Delta T$ in order to replan a path which smoothly deviates from the current trajectory. Indeed, the planned state at t_2 is consistent with the robot dynamic capability by construction since it belongs to the current trajectory which has been checked through the STPIK-check.

It is worth remarking that no mathematical guarantee exists on the capability of the system to replan in time. Let us take into consideration the worst scenario where an obstacle so far unknown to the perception module suddenly appears along the trajectory at a low distance from the end-effector. In this case, the robot may not be able to replan a new feasible trajectory in time. To overcome this issue, we have envisioned an *emergency state* in which the planner stops sending the trajectory to the local planner if an obstacle is suddenly seen by the perception system too close to the end-effector. In this condition, the local planner keeps running in order to ensure the fulfillment of all the constraints, e.g. it reactively avoids the dynamic obstacle, while the global planner computes a new feasible trajectory. Once generated, it is given as reference to the local planner that starts following it again.

The effective avoidance of the dynamic obstacles by the local planner is embedded in the chosen task hierarchy and the null-space projection of the velocities computed to fulfill all the tasks. When the system is in *emergency state* the desired

end-effector pose coming from the planner is a constant value, and the related joint velocities are projected onto the null space of the obstacle avoidance task, which is chosen at a higher priority with respect to the end-effector pose one. When a dynamic obstacle gets too close to the robot, the corresponding task gets activated and the joint velocities that makes it avoid the obstacle are applied. For more details about the handling of safety tasks such as obstacle avoidance, joint limits and virtual walls within the set-based task-priority inverse kinematics framework, the reader is referred to [16].

Algorithm 5 shows the pseudo-code of the online planner.

Algorithm 5: `OnlinePlanner`$(\boldsymbol{\eta}_{\text{ee,r}}(t), \text{knowledge}, \text{perception})$

The reference trajectory generated by the Global Planner is sent to the Local Planner. When there is a change in the environment, the Global Planner replans a new feasible trajectory. If an obstacle is seen too close to the robot, it enters in the *emergency* state until a new feasible trajectory is replanned.

$k = 0$;
while $k <$ *length*$(\boldsymbol{\eta}_{ee,r}(t))$ **do**
  $[$*EMERGENCY*, *envChanged*$] \leftarrow$ `UpdateEnvironment`$(\boldsymbol{\eta}_{\text{ee}}(t), \text{knowledge}, \text{perception})$
  `// updates the obstacles position`
  if *envChanged* **then**
    $[$*SUCCESS*$, \boldsymbol{\eta}_{\text{ee,r}}(t)] \leftarrow$ `GlobalPlanner`$(\boldsymbol{\eta}_{\text{ee,d}}, \text{knowledge}, \text{perception})$
    `// runs in background`
  if *SUCCESS* **then**
    EMERGENCY $=$ **false**;
  `LocalPlanner`$(\boldsymbol{\eta}_{\text{ee,r}}(t_k), \boldsymbol{q}, \text{perception})$;
  if *EMERGENCY* **then**
    continue;
  else
    $k = k + 1$;

III. Algorithm Results

In this section we first show a preliminary simulation aimed at comparing the proposed control architecture with the instantaneous inverse kinematics algorithm, in order to demonstrate its superiority regarding the local minima problem. Then, we show the results of an intensive simulation campaign aimed at evaluating the performances of the proposed control architecture in the presence of static and dynamic obstacles in the environment in which the robot operates. Several scenarios have been taken into account and statistical data regarding the number of failures and the execution time needed by the planner have been gathered in order to validate the performances of the proposed algorithm. Finally, we present an experimental case study in which the robot shares its workspace with a human operator.

While the motion planner step of the Global Planner takes into account only the obstacle avoidance at the end-effector, the STPIK-check and the Local Planner include a *rich* hierarchy of Cartesian and joint-space safety tasks, composed by: 1) joint limits at 2nd, 4th and 6th joints (which are the ones having actual mechanical limits on the manipulator taken into account for the validation); 2) obstacle avoidance at end-effector; 3) obstacle avoidance at elbow; 3) obstacle avoidance at 3rd joint; 4) self-collisions avoidance controlling the distance between the end-effector and the 2nd and 3rd joints.

Table I reports all the tasks included in the hierarchy for the presented simulations and the experimental case study, while a video showing some of the runs is provided as supplementary material for the present paper.

A. Experimental setup

The control architecture has been developed within a framework based on ROS (Robot Operating System) [42]. The latter is a middleware widely used in the robotic research field since it is based on a publish-subscribe paradigm which allows to easily test and integrate software modules.

The robotic system taken into consideration for the simulation campaign and the experimental case study is the Kinova Jaco^2 manipulator that, being a 7 DOFs robot, is kinematically redundant with respect to any end-effector motion. Nevertheless, it is worth remarking that the proposed replanning algorithm is independent of the robotic system. Indeed, any other manipulator could be used.

Global and local planners have been implemented in C++ within the ROS-based framework. More in detail, the motion planner step inside the global planner has been developed using the Open Motion Planning Library (OMPL) [43], which is an open source library providing many state-of-the-art sampling-based motion planning algorithms. Hence, it has been customized in order to achieve the own motion planner used in the first step of the global planner. The STPIK algorithm, used both in the global and local planners, has been developed in C++ as well. In particular, a library of control objectives, namely elementary tasks, has been implemented in order to consider several Cartesian and joint space constraints. The entire simulation campaign has been performed on a laptop with a 6-core Intel i7-8750H processor clocked at 2.20 GHz and 16 GB of RAM. On the same machine has been run RViz for having a visual feedback of the manipulator motion.

B. Comparison with instantaneous inverse kinematics

In this subsection, we present a preliminary simulation that shows the behavior of the proposed control architecture with respect to the local minima problem compared with an instantaneous inverse kinematics control.

The manipulator starts from a certain initial joint configuration and it has to reach the desired end-effector pose $\boldsymbol{\eta}_{\text{ee,d}} = \begin{bmatrix}\boldsymbol{p}_{\text{ee,d}} & \boldsymbol{\mathcal{Q}}_{\text{ee,d}}\end{bmatrix}^{\text{T}}$, with desired position $\boldsymbol{p}_{\text{ee,d}} = \begin{bmatrix}0.6 & -0.4 & 0.13\end{bmatrix}^{\text{T}}$m, and desired quaternion of $\boldsymbol{\mathcal{Q}}_{\text{ee,d}} = \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}^{\text{T}}$. In the scenario taken into consideration, three static obstacles, very close to each other, have been placed between the initial and the desired and-effector pose

TABLE I: Implemented task hierarchy for the IK-check step of the global planner in the simulation campaigns. In addition to the position/orientation tasks, a number of safety tasks both in Cartesian and joint spaces are included at higher priority levels.

Task name	Task value $\boldsymbol{\sigma}_{\text{x}}$	Task dimension m	Description
Arm joint limits	$\sigma_{\text{joint}} = q_{\text{i}}$	$1 \times n$ joints	Control of the i-th arm joint value to make it lie within a defined set
Obstacle avoidance at the end-effector	$\sigma_{\text{ee,obst}} = \sqrt{(\boldsymbol{p}_{\text{o,i}} - \boldsymbol{p}_{\text{ee}})^{\text{T}}(\boldsymbol{p}_{\text{o,i}} - \boldsymbol{p}_{\text{ee}})}$	$1 \times k$ obstacles	Control of the distance between the arm end-effector $\boldsymbol{p}_{\text{ee}}$ and the i-th obstacle position $\boldsymbol{p}_{\text{o,i}}$
Obstacle avoidance at joint 3	$\sigma_{q_3,\text{obst}} = \sqrt{(\boldsymbol{p}_{\text{o,i}} - \boldsymbol{p}_{\text{j3}})^{\text{T}}(\boldsymbol{p}_{\text{o,i}} - \boldsymbol{p}_{q_3})}$	$1 \times k$ obstacles	Control of the distance between the Cartesian third joint position $\boldsymbol{p}_{j_3}$ and the i-th obstacle position $\boldsymbol{p}_{\text{o,i}}$
Obstacle avoidance at the elbow	$\sigma_{\text{el,obst}} = \sqrt{(\boldsymbol{p}_{\text{o,i}} - \boldsymbol{p}_{\text{el}})^{\text{T}}(\boldsymbol{p}_{\text{o,i}} - \boldsymbol{p}_{\text{el}})}$	$1 \times k$ obstacles	Control of the distance between the arm elbow $\boldsymbol{p}_{\text{el}}$ and the i-th obstacle position $\boldsymbol{p}_{\text{o,i}}$
Self-collisions avoidance	$\sigma_{\text{ee},q_i} = \sqrt{(\boldsymbol{p}_{q_i} - \boldsymbol{p}_{\text{ee}})^{\text{T}}(\boldsymbol{p}_{q_i} - \boldsymbol{p}_{\text{ee}})}$	$1 \times n$ joints	Control of the distance between the arm end-effector $\boldsymbol{p}_{\text{ee}}$ and the i-th Cartesian joint position $\boldsymbol{p}_{q_i}$
End-effector position	$\boldsymbol{\sigma}_{\text{ee,pos}} = \boldsymbol{p}_{\text{ee}}$	3	Control of the arm end-effector position $\boldsymbol{p}_{\text{ee}}$
End-effector orientation	$\boldsymbol{\sigma}_{\text{ee,ori}} = \boldsymbol{\Phi}_{\text{ee}}$	3	Control of the arm end-effector orientation $\boldsymbol{\Phi}_{\text{ee}}$

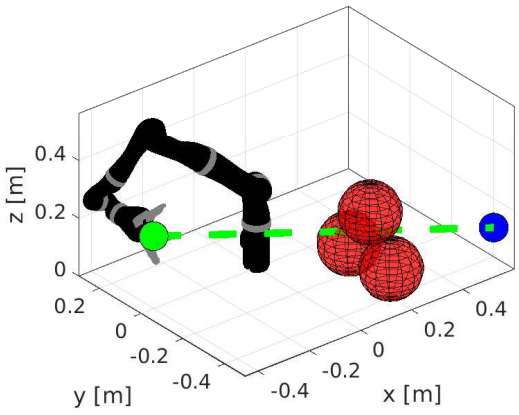

Fig. 4: Graphical representation of the scenario taken into account in the preliminary simulation. The green circle represents the initial end-effector position, while the blue one represents the final position. The green-dashed line represents the desired path for the instantaneous inverse kinematics simulation. It is clear that the end-effector would encounter the three obstacles while following the desired path. Given the chosen obstacles positions, the instantaneous inverse kinematics will inevitably fall into a local minimum problem, preventing the end-effector to reach the final position.

in order to intentionally obtain a local minimum problem. More specifically the position of the three obstacles with respect to the base frame of the manipulator has been set to $\boldsymbol{p}_{o,1} = [0.2 \quad -0.4 \quad 0.1]^T$m, $\boldsymbol{p}_{o,2} = [0.2 \quad -0.2 \quad 0.1]^T$m, $\boldsymbol{p}_{o,3} = [0.2 \quad -0.3 \quad 0.25]^T$m. Figure 4 shows the considered scenario.

In the following, we show the results of a simulation in which the manipulator motion is controlled by resorting only to instantaneous inverse kinematics, i.e. the same algorithm used in the Local Planner block, without using the Global Planner one. The desired trajectory for the end-effector $\boldsymbol{\eta}_{\text{ee,r}}(t)$ is a segment connecting the initial and desired final position to be followed with a simple trapezoidal velocity profile. Figure 5 shows the simulation results. In particular, looking at Fig 5.(a) it is possible to understand that the end-effector does not reach the final desired pose, as it reaches the minimum distance from all the three obstacles (Figs 5.(c), 5.(d), 5.(e)). Additionally, also two of the joint limit tasks (5.(b)) reached their thresholds at steady state. The concurrent activation of all these safety tasks, being at a higher priority level with respect to the end-effector pose one, eventually forced the end-effector to completely stop the motion. In other words, the projection of the joint velocities related to the achievement of the end-effector pose task onto the null space of all the active safety tasks gives the null vector, resulting in a local minimum.

In a second simulation, we instead used the entire proposed control architecture, with the same initial joint configuration and the same final end-effector desired pose of the previous one. Looking at the results in Fig. 6, it is clear that in this case the manipulator successfully accomplishes the task avoiding the local minimum. Indeed, all the end-effector trajectories that would end up in a local minimum are discarded by the ik-check block, and the reference end-effector trajectory in output of the planner successfully makes the robot reach the desired final pose, while respecting all the safety constraints during the motion.

C. Simulations with static obstacles

In this subsection we show the results of the proposed algorithm in the presence of static obstacles, whose positions are considered known a-priori by the robot. Four different scenarios of increasing complexity have been considered, differentiated by the number of static obstacles included in the scene. In all of them the manipulator starts from the same initial joint configuration and it has to reach the desired end-effector pose $\boldsymbol{\eta}_{\text{ee,d}} = \begin{bmatrix}\boldsymbol{p}_{\text{ee,d}} & \boldsymbol{\mathcal{Q}}_{\text{ee,d}}\end{bmatrix}^{\text{T}}$, with desired position $\boldsymbol{p}_{\text{ee,d}} = [0.6 \quad 0.0 \quad 0.13]^{\text{T}}$m, and desired quaternion of $\boldsymbol{\mathcal{Q}}_{\text{ee,d}} = [0 \quad 1 \quad 0 \quad 0]^{\text{T}}$. In a first scenario, the end-effector of the robot is asked to reach the desired configuration without any obstacle in the scene. The other three scenarios are

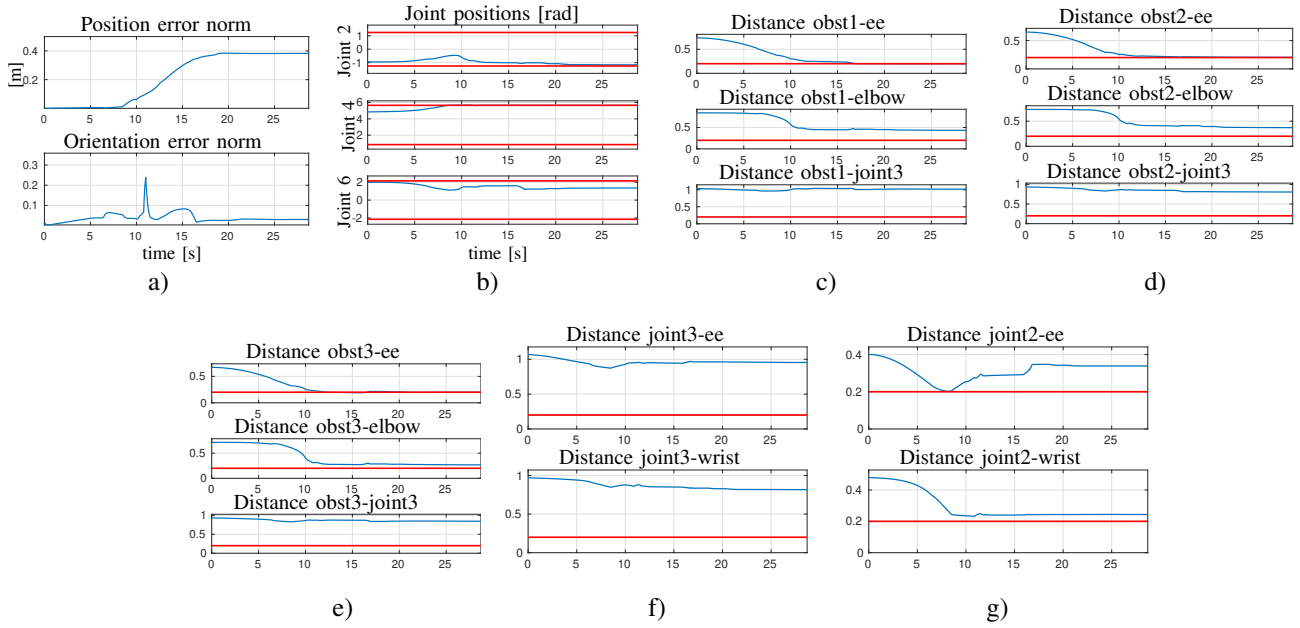

Fig. 5: Task values during the execution of the desired trajectory using only the instantaneous inverse kinematics algorithm (Local Planner block). a) Position and orientation error norms. The end-effector could not reach the desired pose due to the concurrent activation of three obstacle avoidance tasks that forced it to stop in a local minimum; b) Joint positions; c), d), e) Distance between obstacle 1, 2, 3 and end-effector, elbow and joint3; f), g) Distance between joint3, joint2 and end-effector and wrist. Solid red horizontal lines represent the safety thresholds imposed for the set-based tasks.

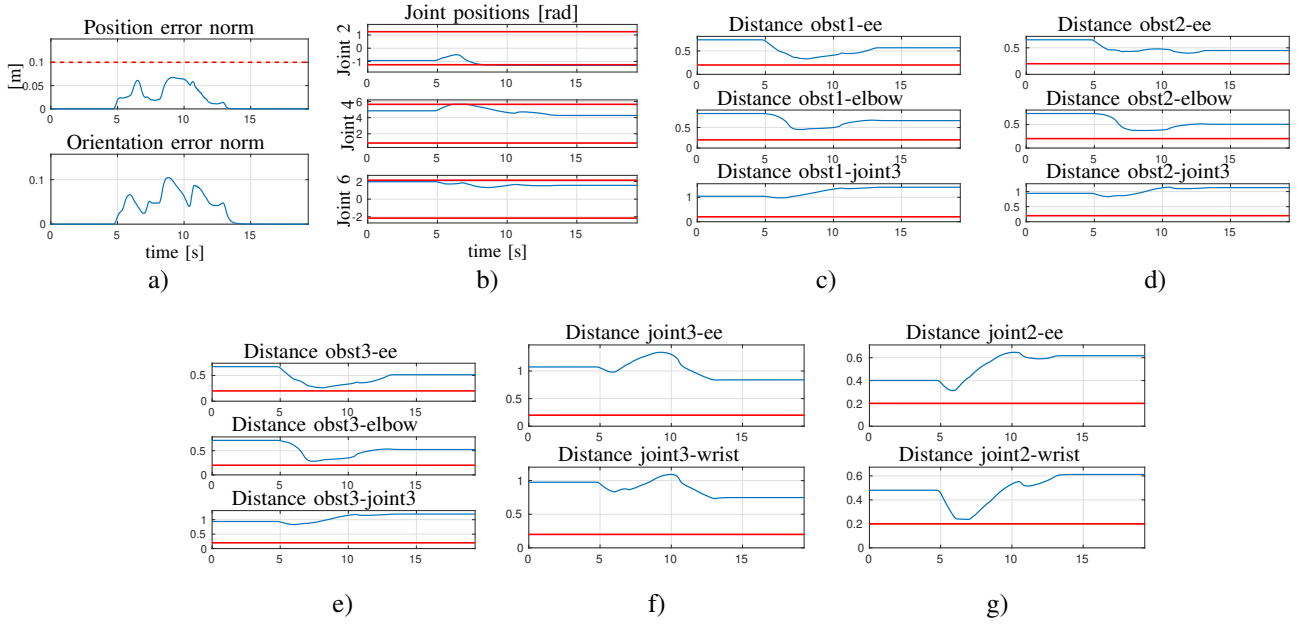

Fig. 6: Task values during the execution of the desired trajectory using both the Local Planner and Global Planner blocks. a) Position and orientation error norms. This time all the candidate trajectories that would end up in a local minimum are discarded by the Global Planner block, thanks to the IK-check. The position error stays below the value set as *failure threshold* (dashed red horizontal line) for the planner during all the motion; b) Joint positions; c), d), e) Distance between obstacle 1, 2, 3 and end-effector, elbow and joint3; f), g) Distance between joint3, joint2 and end-effector and wrist. Solid red horizontal lines represent the safety thresholds imposed for the set-based tasks.

characterized by the presence of one, two and three obstacles respectively, intentionally placed along the path executed by the end-effector, the wrist or the elbow of the manipulator in order to increase the complexity for the planning operation. More specifically the position of the three obstacles with respect to the base frame of the manipulator has been set to $\boldsymbol{p}_{o,1} = [0.1 \quad -0.3 \quad 0.6]^T$m, $\boldsymbol{p}_{o,2} = [0.2 \quad -0.4 \quad 0.4]^T$m, $\boldsymbol{p}_{o,3} = [0.0 \quad 0.45 \quad 0.4]^T$m. For each one of the scenarios, 120 simulation have been carried out in order to gather statistical data. Figure 7 shows a graphical representation of the four scenarios, together with the path in output of the planner for each one of the simulations, while Fig. 8 shows the box charts related to the number of failures and the overall execution time of the planner for each one of the scenarios. It is possible to see that the number of failures and consequently the overall execution time increases together with the complexity of the scenario taken into account. It is worth noticing that also in the first scenario, despite the absence of obstacles, there is a number of failures in the path generation due to the constraints imposed by the joint limits and the self-collisions tasks. Indeed, the initial joint configuration of the robot and the desired pose of the end-effector have been intentionally chosen in order to trigger those tasks in case of a *too straight* path in output from the motion planner. Overall, the total execution time stays below 1.5s even for the most complex scenario.

Figure 9 shows the task values and the position/orientation errors during one of the simulations of the fourth scenario. It is possible to see that the position error stays below the value chosen as *failure threshold* for the global planner (10cm in these simulations), while all the set-based task values (obstacle avoidance, joint limits and self hits) stay within the imposed safety thresholds.

D. Simulations with dynamic obstacles

In this subsection we evaluate the performances of the proposed algorithm in the presence of dynamic obstacles, focusing in particular on the replanning capability. Three scenarios have been considered, each one with three obstacles differentiated by the initial positions and the amplitude and the frequency of a sinusoidal motion along one of the axis. Also for this case study, 120 simulations have been carried out for each scenario.

In these sets of simulations, the Global Planner initially generates the desired trajectory not considering the obstacles. Once the trajectory is generated, the obstacles spawn in the scene and start their motion. At that point, the robot replans the trajectory following the replanning logic described in Sec II-C. Figure 10 shows the gathered statistical data regarding the number of failures and the overall execution time of the replanning algorithm. It is possible to see that the average time needed for the replanning operations is around 0.5s, independently of the scenario taken into account, successfully achieving a real-time replanning of the path in all the simulations. Obviously the complexity of the scenario affects the execution time of the planning and it is not necessarily related to the number of obstacles itself, but rather on the number of safety tasks included in the IK-check that get active during the motion. A formal analysis on its impact of is thus not straightforward, but the obtained results on the considered scenarios show the effectiveness and the general suitability of the approach for real-time application.

Figure 11 shows the task values during one of the simulations of the first scenario. Even in this case, it is clear that the end-effector reaches the desired final pose, taking the norm of the position error always below the value chosen as *failure theshold*. Regarding the other safety tasks, several of them gets active during the motion, effectively preventing any collision with the dynamic obstacles in the scene.

E. Experimental case study

The proposed control architecture has been tested and validated in an experimental case study in which the robot shares its workspace with a human operator [1]. He is monitored in real-time with an Intel RealSense RGB-D sensor. OpenPose [44] detects and tracks the points of interest of the skeleton of the operator within the image plane, whose positions in 3D-space are obtained by the superimposition of the depth image with the RGB one, using the ROS package [45]. In particular, only the eight points of interest belonging to the arms, the neck and the head of the operator are considered as obstacles by the robot, for lightening the computation load. Figure 12 shows the point cloud taken from the RealSense sensor and the extracted obstacles. The initial joint configuration and the desired final end-effector position/orientation are the same as the simulation campaign, while the operator sits next to the robot and extends an arm along the planned path triggering a replan operation. It is worth noticing that the implemented task hierarchy is the same as the one chosen for the simulation campaign. Figure 13 shows some screenshots of the two runs of the experiment: in the first sequence the robot successfully manage to replan a collision-free path in time; in the second sequence, the operator intentionally takes more time to extend his arm, in order to test the *emergency state*. The robot detects the obstacle and stops its motion, trying to replan a new path. The experimental results highlight the effectiveness of the proposed control architecture in avoiding dangerous collisions also in real scenarios, with actual real-time monitoring of a human operator that shares the workspace with the robot.

IV. Discussion

A. Node-to-node check, an alternative implementation

Referring to Fig. 3 it is possible to notice that an alternative implementation can be employed. In detail, the STPIK check may be run during the motion planning when each node is added during the graph building. It is worth noticing that a similar approach is implemented in the control aware paradigm [26].

As explained above, the proposed method first plans the entire trajectory connecting the start and target points in Cartesian space and then it performs a check through the STPIK algorithm including also the joint-space constraints.

[1]The human operator is the first author of the present paper

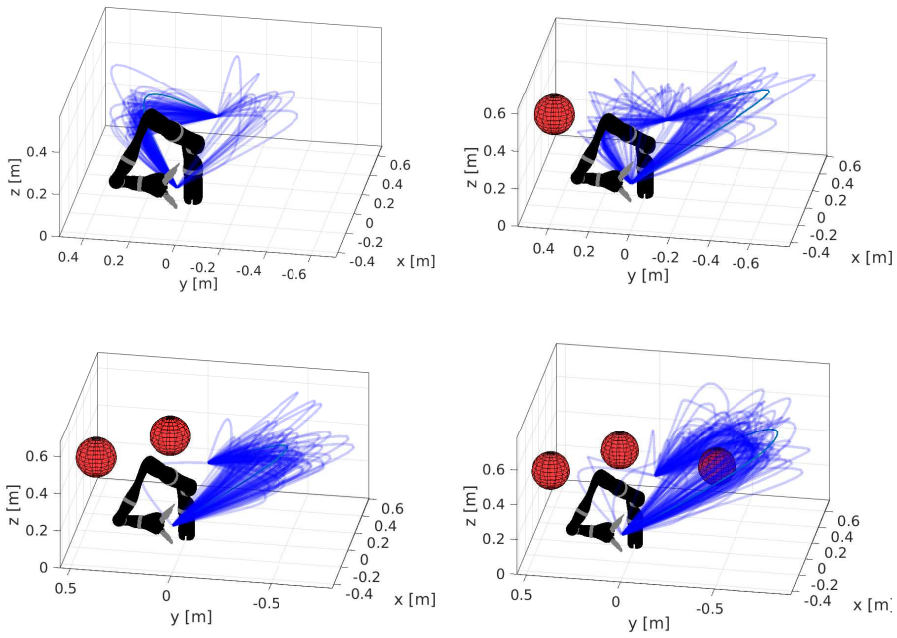

Fig. 7: Graphical representation of the four scenarios taken into account in the simulations. The initial configuration of the robot and the desired position and orientation are the same for all the scenarios, while the number of obstacles varies (red spheres). Blue lines represent the path in output from the planner for each one of the 120 simulations carried out for gathering statistical data.

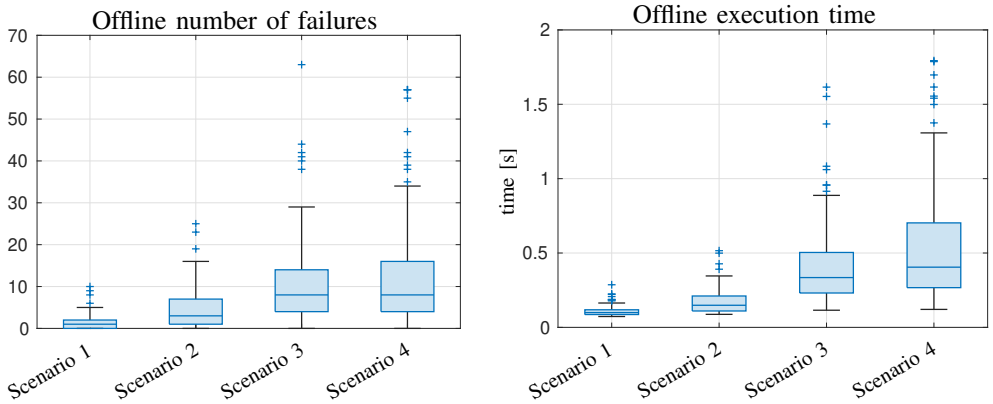

Fig. 8: Box charts of the number of failures and execution time related to the static obstacles simulations (120 simulations for each scenario)

From one hand, the node-to-node implementation allows to erase the unfeasible branches as soon as a joint-space constraint is violated. From the other hand, due to the Boundary Value Problem [46], some of the motion planning algorithm can not be implemented such as the RRT Connect.

Another difference is in the execution time. In order to compare the two approaches definitions of critical times is given in Table II.

The total time needed to compute a trajectory for the two algorithms is then given by

$$T_a = k_{p,a}(T_{p,a} + T_{\text{IK},p}) \,, \tag{8}$$

$$T_b = k_{p,b}(T_{p,b} + T_{\text{IK},p}) \,, \tag{9}$$

where

$$T_{p,b} = n(T_n + T_{\text{IK},n}) \tag{10}$$

TABLE II: Significant times for proposed and *node-to-node check* algorithms

variable	average time to compute…
$T_{p,a}$	Cartesian plan from A to B for alg. a
$T_{p,b}$	Cartesian plan from A to B for alg. b
$T_{\text{IK},p}$	run the STPIK on the A-to-B traj
$T_{\text{IK},n}$	run the STPIK between two consecutive nodes
T_a	total traj for alg. a (see eq. (8))
T_b	total traj for alg. b (see eq. (9))

with n the number of sampled nodes, T_n the time for sampling one node and $k_{p,a}/k_{p,b}$ the average number of trials necessary

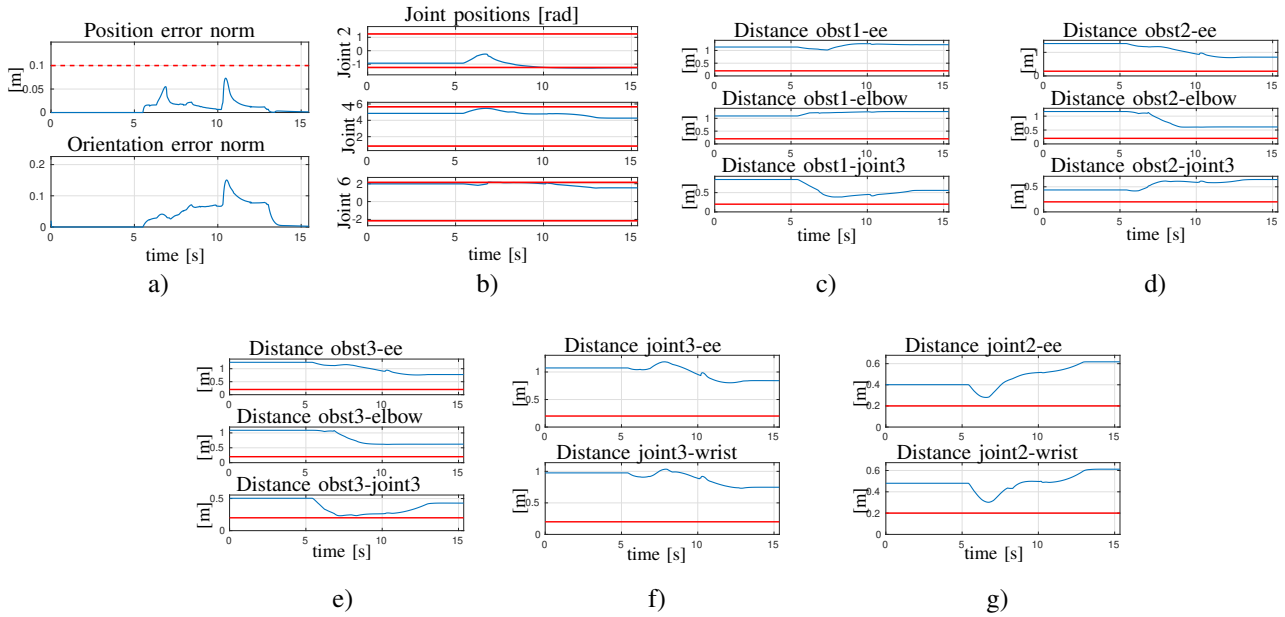

Fig. 9: Task values during the execution of the desired path in output from the planner. a) Position and orientation error norms. The position error stays below the value set as *failure threshold* (dashed red horizontal line) for the planner during all the motion; b) Joint positions; c), d), e) Distance between obstacle 1, 2, 3 and end-effector, elbow and joint3; f), g) Distance between joint3, joint2 and end-effector and wrist. Solid red horizontal lines represent the safety thresholds imposed for the set-based tasks. It is clear that several of them get active during the motion, making the manipulator respect all the imposed constraints both in joint space and Cartesian space.

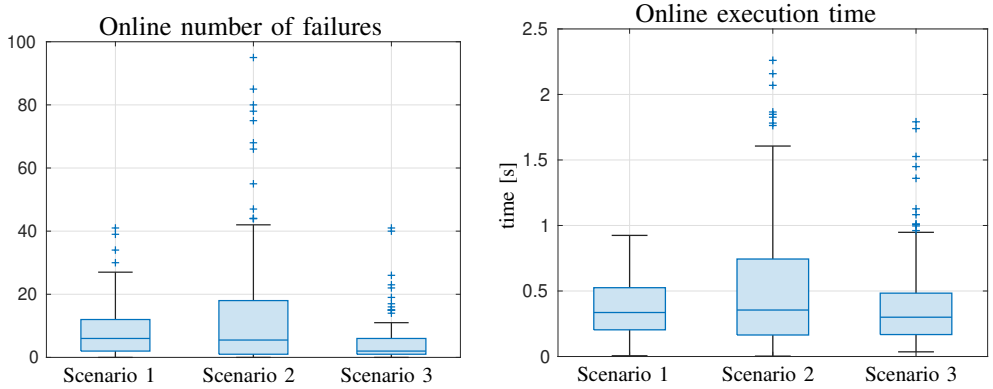

Fig. 10: Box charts of the number of failures and execution time related to the dynamic obstacles simulations (120 simulations for each scenario)

for the planner to find a feasible trajectory for the two algorithms.

It is clearly very difficult to compare those times for the general case. However, few comments can be made:

- $T_{\text{IK},n}$ depends on the robotic DoF, the specific IK algorithm implemented and the distance d_{node} between two consecutive nodes. The latter is a parameter of the planning algorithm, equal for a and b;
- Given an average path length l_{path} and d_{node}, the IK-check time relation can be approximated as

$$T_{\text{IK},p} \approx \frac{l_{\text{path}}}{d_{\text{node}}} T_{\text{IK},n} \ , \tag{11}$$

- $T_{p,a}$ is *simply* the planning time in Cartesian space. It is function of the environment size, its crowding and the algorithm used together with the corresponding parameters;
- $T_{p,b}$ is *similar* to $T_{p,a}$. However, as shown in (10) it needs to run n times $T_{\text{IK},n}$ where n exhibits the same dependency of $T_{p,a}$ but, due to the BVP, a smaller set of planning algorithms can be used;
- The number of needed run $k_{p,a}$ and $k_{p,b}$ is also function of the environment size and its crowding, clearly the same for both the algorithms but, being strongly influenced also by the planning algorithm may exhibit differences due to the BVP arising in algorithm b.

A numerical analysis has been run but is not presented for the case at hand for seek of space since it is clearly in favour of the algorithm a, i.e., that $T_a < T_b$ for this robotic system varying significantly all the other parameters.

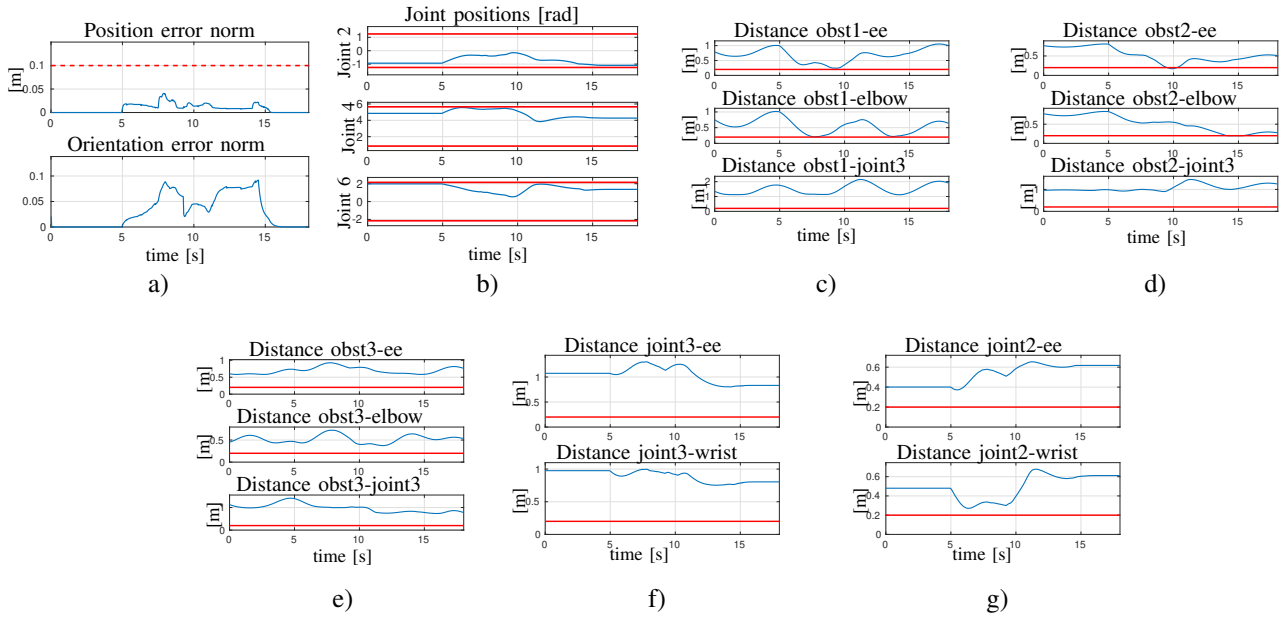

Fig. 11: Task values during the execution of the desired path in output from the planner. a) Position and orientation error norms. The position error stays below the value set as *failure threshold* (dashed red horizontal line) for the planner during all the motion; b) Joint positions; c), d), e) Distance between obstacle 1, 2, 3 and end-effector, elbow and joint3; f), g) Distance between joint3, joint2 and end-effector and wrist. Solid red horizontal lines represent the safety thresholds imposed for the set-based tasks. It is clear that several of them get active during the motion, making the manipulator respect all the imposed constraints both in joint space and Cartesian space also in the presence of moving obstacles.

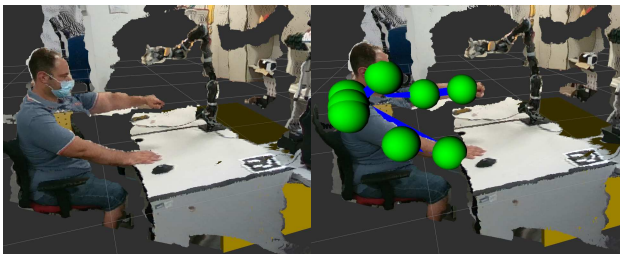

Fig. 12: Left: point cloud taken from the RealSense sensor; right: point cloud and detected obstacles superimposed

B. Delegation vs exploration

In this work we assumed that the robot is given an initial joint configuration, a list of tasks and a target end-effector pose to reach. The implemented algorithm, thus, owns the probabilistic features of the global planner of finding a solution.

In [26] the delegation-exploration compromise is discussed. The *delegation* aspect means that the sampling phase is run in Cartesian space, thus in a space with lower dimension with respect to the system's DoFs, and the redundancy exploitation is in charge at the inverse kinematics, the STPIK also in that case. This approach has the advantage to lower the complexity for the random search of the path in Cartesian space. The *exploration* feature is achieved when the redundancy is not exploited by the local controller and left, at least partially, to the global planner. The price is to modify the assigned control problem, i.e., to reach a certain pose with a list of prioritized tasks.

In [30] the null space of a motion planning problem assigned to a robot is exploited in a different manner, i.e., it is used to sample self-motions allowing to track the assigned end-effector trajectory.

It is worth noticing that it is not always possible to relax the tasks, in fact, a common way to prioritize is to cluster them in 3 groups [27]:

- high priority. safety tasks, most of the time set-based such as virtual walls, joint limits, self-collision, obstacle and human avoidance;
- medium priority. mission task, i.e., the reason why the robot is moving. Most of the time end-effector pose or end-effector position, rarely the sole end-effector orientation;
- lower priority. optimization tasks such as maximization of robot manipulability.

While it is easy to get rid of the lower priority tasks, related to optimization, it is undesirable to release the safety ones. Concerning the medium-priority it may be possible to control the sole position instead of the whole end-effector pose. This allow to *gain* 3-4 DoFs and run again the global planning algorithm *on a different problem*, which may be allowed or not.

In alternative, it is possible to search for a different solution with an approach that we defined as *reconfiguration motion* whose aim is to change the joint configuration by restarting the motion from the same end-effector pose. In practice this is done by keeping, as always, the safety tasks, modifying the end-effector equality tasks in set-based as well allowing *small* deviation from the initial value and ignoring the optimization tasks. The robot is the reconfigured randomly with an *enriched* internal motion and from this new joint configuration the

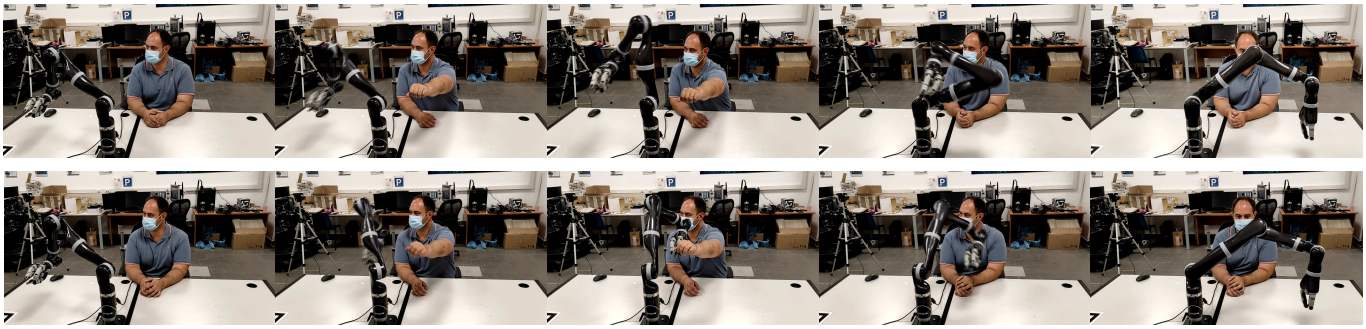

Fig. 13: Top: screenshots of the video taken during one of the experiments. The human operator extends his arm along the planned path, triggering a replan operation. Bottom: screenshots of an experimental run in which the robot enters in *emergency state*. The robot stops the motion before the collision with the operator's arm, trying to replan the path. Once the operator removes the arm, the robot resumes the previously planned path.

global planner works again.

V. Conclusions and Future Work

In this paper we have presented a control architecture that merges a Global Planner and a Local Planner, aiming at combining the advantages of a sampling-based motion planning algorithm with the real-time and reactive capabilities of a task-priority inverse kinematics algorithm. Hundreds of simulations on environments characterized by static and dynamic obstacles have been presented, and the gathered statistical data show the effectiveness of the proposed algorithm in replanning online the motion of a 7 DoF manipulator guaranteeing the safety of the system while overcoming the local minima problems that usually affects gradient-based local planners. An experimental case study using a Kinova Jaco^2 and an Intel Realsense sensor for monitoring a human operator that shares the workspace with the robot has been additionally presented, proving the suitability of the proposed architecture in real scenarios.

Future works will concern a further exploitation of the redundancy of the system, e.g. by implementing the null-space sampling discussed in Sec. IV-B in order to improve the planning capability and optimize the efficiency of the proposed motion planner. Further efforts will be also devoted to the extension of the proposed algorithm to more complex structure, such as dual-arm systems with a mobile base, in which their intrinsic redundancy can be exploited to design more efficient behaviors of the robot.

Finally, the proposed method might be further improved by driving the search towards the constraint manifold, like done, e.g., in [47]. This would significantly decrease the number of failures in the Global Planner, speeding up the trajectory planning process.

Acknowledgments

This work was supported by H2020-ICT project CANOPIES-A Collaborative Paradigm for Human Workers and Multi-Robot Teams in Precision Agriculture Systems (Grant Agreement N. 101016906), by POR FSE LAZIO 2014-2020, Project DE G06374/2021 and by the MIUR program "Dipartimenti di Eccellenza 2018-2022" granted to Department DIEI of the University of Cassino and Southern Lazio.

References

[1] J. Goetz, S. Kiesler, and A. Powers, "Matching robot appearance and behavior to tasks to improve human-robot cooperation," in *Proceedings of the 12th IEEE international workshop on robot and human interactive communication*. IEEE Press Piscataway, NJ, 2003, pp. 55–60.

[2] F. Arrichiello, P. Di Lillo, D. Di Vito, G. Antonelli, and S. Chiaverini, "Assistive robot operated via P300-based brain computer interface," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6032–6037.

[3] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[4] G. Dudek and M. Jenkin, *Computational principles of mobile robotics*. Cambridge university press, 2010.

[5] X. Yu, B. Li, W. He, Y. Feng, L. Cheng, and C. Silvestre, "Adaptive-constrained impedance control for human-robot co-transportation," *IEEE Transactions on Cybernetics*, pp. 1–13, 2021.

[6] X. Yu, W. He, H. Li, and J. Sun, "Adaptive fuzzy full-state and output-feedback control for uncertain robots with output constraint," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6994–7007, 2021.

[7] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Journal of dynamic systems, measurement, and control*, vol. 108, no. 3, pp. 163–171, 1986.

[8] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[9] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proceedings 5th International Conference on Advanced Robotics*, Pisa, I, 1991, pp. 1211–1216.

[10] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[11] N. Mansard and F. Chaumette, "Task sequencing for high-level sensor-based control," *IEEE Transactions on Robotics and Automation*, vol. 23, no. 1, pp. 60–72, 2007.

[12] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The Null-Space-based Behavioral control for autonomous robotic systems," *Journal of Intelligent Service Robotics*, vol. 1, no. 1, pp. 27–39, Jan. 2008.

[13] F. Flacco and A. D. Luca, "A reverse priority approach to multi-task control fo redundant robots," in *Proceedings 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, Illinois, Sept. 2014, pp. 2421–2427.

[14] E. Simetti, G. Casalino, S. Torelli, A. Sperindé, and A. Turetta, "Floating underwater manipulation: Developed control methodology and experimental validation within the TRIDENT project," *Journal of Field Robotics*, vol. 31(3), pp. 364–385, 2013.

[15] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[16] P. D. Lillo, F. Arrichiello, G. Antonelli, and S. Chiaverini, "Safety-related tasks within the set-based task-priority inverse kinematics framework," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[17] G. Notomista, S. Mayya, M. Selvaggio, M. Santos, and C. Secchi, "A set-theoretic approach to multi-task execution and prioritization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9873–9879.
[18] E. A. Basso and K. Y. Pettersen, "Task-priority control of redundant robotic systems using control lyapunov and control barrier function based quadratic programs**this research was partly funded by the research council of norway through the centres of excellence funding scheme, project no. 223254 ntnu amos." *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9037–9044, 2020, 21th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320326598
[19] E. Cataldi, F. Real, A. Suarez, P. Di Lillo, F. Pierri, G. Antonelli, F. Caccavale, G. Heredia, and A. Ollero, "Set-based inverse kinematics control of an anthropomorphic dual arm aerial manipulator," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2960–2966.
[20] S. Moe, J. T. Gravdahl, and K. Y. Pettersen, "Set-based control for autonomous spray painting," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1785–1796, 2018.
[21] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.
[22] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, June 2010.
[23] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2009, pp. 625–632.
[24] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.
[25] A. Shkolnik and R. Tedrake, "High-dimensional underactuated motion planning via task space control," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 3762–3768.
[26] M. Tognon, E. Cataldi, H. Tello Chávez, G. Antonelli, J. Cortes, and A. Franchi, "Control-aware motion planning for task-constrained aerial manipulation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2478–2484, 2018.
[27] G. Buizza Avanzini, A. Zanchettin, and P. Rocco, "Constrained model predictive control for mobile robotic manipulators," *Robotica*, vol. 36, no. 1, pp. 19–38, 2018.
[28] G. Oriolo and M. Vendittelli, "A control-based approach to task-constrained motion planning," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 297–302.
[29] G. Oriolo, M. Cefalo, and M. Vendittelli, "Repeatable motion planning for redundant robots over cyclic tasks," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1170–1183, Oct 2017.
[30] M. Cefalo and G. Oriolo, "A general framework for task-constrained motion planning with moving obstacles," *Robotica*, vol. 37, no. 3, pp. 575–598, 2019.
[31] S. Murray, W. Floyd-Jones, , Y. Qi, D. Sorin, and G. Konidaris, "Robot motion planning on a chip," in *Robotics: Science and Systems*, 2016.
[32] D. Di Vito, M. Bergeron, D. Meger, G. Dudek, and G. Antonelli, "Dynamic planning of redundant robots within a framework of set-based task-priority inverse kinematics," in *2020 IEEE Conference on Control Technology and Applications*, Montreal, CA, August 2029.
[33] R. E. Roberson and R. Schwertassek, *P Dynamics of Multibody Systems*. Berlin, D: Springer-Verlag, 1988.
[34] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The NSB control: a behavior-based approach for multi-robot systems," *Paladyn Journal of Behavioral Robotics*, vol. 1, no. 1, pp. 48–56, 2010.
[35] B. Siciliano and J.-J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. Fifth International Conference on Advanced Robotics (ICAR)*. Pisa, Italy: IEEE, 1991, pp. 1211–1216.
[36] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. Schrimpf, "Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results," *Frontiers in Robotics and AI*, vol. 3, p. 16, 2016.
[37] P. Di Lillo, F. Arrichiello, D. Di Vito, and G. Antonelli, "Bci-controlled assistive manipulator: Developed architecture and experimental results," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 1, pp. 91–104, 2021.
[38] D. D. Vito, C. Natale, and G. Antonelli, "A comparison of damped least squares algorithms for inverse kinematics of robot manipulators," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6869–6874, 2017.
[39] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: https://books.google.ca/books?id=Clg8SWNMSRAC
[40] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, April 2000, pp. 995–1001 vol.2.
[41] V. Braibant and C. Fleury, "Shape optimal design using b-splines," *Computer methods in applied mechanics and engineering*, vol. 44, no. 3, pp. 247–267, 1984.
[42] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.
[43] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012.
[44] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
[45] R. P. Joshi, M. K. Broek, X. Z. Tan, A. Choi, and R. Luo, "ROS OpenPose," https://github.com/ravijo/ros_openpose, 2019.
[46] C. Xie, J. van den Berg, S. Patil, and P. Abbeel, "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4187–4194.
[47] P. Englert, I. M. R. Fernandez, R. K. Ramachandran, and G. S. Sukhatme, "Sampling-based motion planning on sequenced manifolds," *arXiv preprint arXiv:2006.02027*, 2020.

Paolo Di Lillo received the Ph.D. degree in robotics from the University of Cassino and Southern Lazio in 2018. He is currently Assistant Professor at the University of Cassino and Southern Lazio. His research interests include dynamic and kinematic control methods for redundant base-fixed manipulators, as well as mobile dual arm systems, assistive robotics and control underwater vehicle-manipulator systems. He has been involved in the H2020 project DexROV on autonomous underwater intervention with remote supervision via satellite communication.

Daniele Di Vito is a Postdoctoral Fellow at the University of Cassino and Southern Lazio. His research interests are about motion planning, kinematic and dynamic control of mobile and base-fixed manipulator systems both in assistive and underwater robotics. He has been also involved in the H2020 project ROBUST on seabed mining through an underwater vehicle-manipulator system.

Gianluca Antonelli is Full Professor at the "University of Cassino and Southern Lazio". His research interests include marine and industrial robotics, multi-agent systems, identification. He has published 50 international journal papers and more than 120 conference papers, he is author of the book "Underwater Robots" (Springer, 2003, 2006, 2014, 2018) and co-authored the chapter "Underwater Robotics" for the Springer Handbook of Robotics, (Springer, 2008, 2016). He is member of the "IEEE Robotics & Automation Society" (RAS) Administrative Committee, he is coordinator of the EuRobotics Topic Group in Marine Robotics, he has been secretary of the IEEE-Italy section, he has been chair of the IEEE RAS Italian Chapter, he has been Chair of the IEEE RAS Technical Committee in Marine Robotics. He served in the Editorial Board of the IEEE Transactions on Robotics, IEEE Transactions on Control Systems Technology, Springer Journal of Intelligent Service Robotics