2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)
Beijing, China, November 6-9, 2018

Planning in Time-Configuration Space for Efficient Pick-and-Place in Non-Static Environments with Temporal Constraints

Yiming Yang, Wolfgang Merkt, Vladimir Ivan, and Sethu Vijayakumar

***Abstract*— This paper presents a novel sampling-based motion planning method using bidirectional search with a time-configuration space representation that is able to efficiently generate collision-free trajectories in complex and non-static environments. Our approach exploits time indexing to separate a complex problem with mixed constraints into multiple sub-problems with simpler constraints that can be solved efficiently. We further introduce a planning framework by incorporating the proposed planning method enabling efficient pick-and-place of large objects in various scenarios. Simulation as well as hardware experiments show that the method also scales from redundant robot arms to mobile manipulators and humanoids. In particular, we have demonstrated that the proposed method is able to plan collision-free motion for a humanoid robot to pick up a large object placed inside a moving storage box while walking.**

I. Introduction

The goal in motion planning using trajectory optimization is to compute state-space trajectories that minimize an objective function while satisfying a set of constraints. The cost function usually minimizes the control effort or maximizes a smoothness criterion while the task is often defined by constraints. Hereby, we distinguish between two types of constraints: 1) differentiable constraints, e.g., equality and inequality constraints such as position and orientation of the gripper, 2) binary constraints, e.g., state validity such as collision checks. The latter type is often necessary when the constraint is discontinuous, non-smooth, highly non-linear, and cannot be replaced or represented by a proxy constraint. While type 1 constraints can be efficiently satisfied via optimization exploiting the gradient (derivative) of the constraint function, type 2 constraints commonly require stochastic optimization or sampling-based approaches.

Finding a feasible and/or optimal solution while the two types of constraints are active at the same time is non-trivial. However, there exists a class of problems, such as pick-and-place, for which the differentiable and binary constraints are temporally separable. A pick-and-place task is composed of a reaching motion (primarily avoiding obstacles), grasping motion (dominated by accurate gripper positioning), and placing motion (avoiding obstacles again, however, with changed collision geometry due to attached/picked up object). We propose to decompose this task into the three respective sub-problems which will enable us to apply efficient solvers for each constraint type respectively and adjoin them through constraints across the transitions. To do this, we require that all variables describing the problem are indexed on time, e.g. goal positions, obstacle motion, etc.

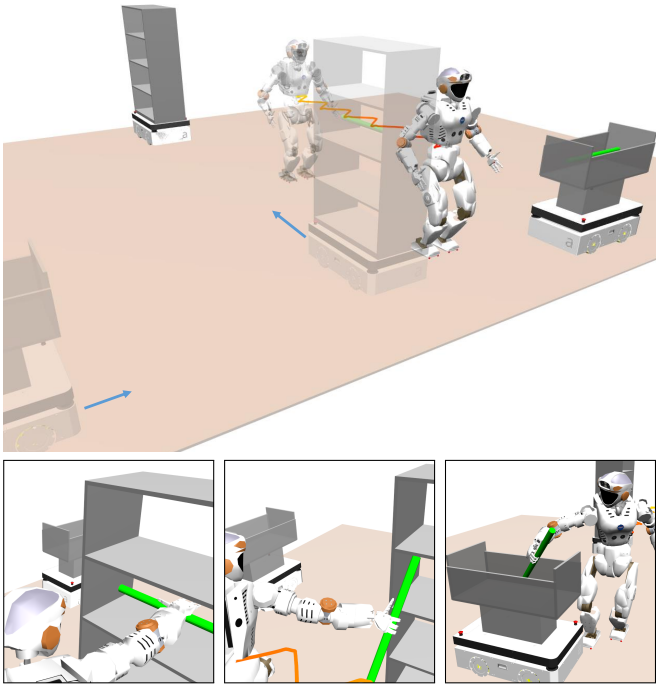

Fig. 1: The NASA Valkyrie humanoid robot picking-up a $0.8m$ long target from a moving shelf and placing it into a moving storage box while walking. The top figure shows the environment setup and start (transparent) and goal (solid) states. The bottom figures highlight some key frames during the pick-and-place task.

Many industrial approaches delivering efficient pick-and-place solutions focus on perception and grasp planning, while having the environment manually designed to be simple and static, such that motion planning has to only account for static obstacles and moving targets [1]. However, collision avoidance and interaction with the environment are key for collaborative as well as humanoid robots that operate in unstructured environments. On the other hand, collision-free motion planning in static environments is a much easier problem even in very complex scenarios by using sampling-based planning algorithms such as Rapidly Ex-

This research is supported by the EPSRC UK RAI Hub in Future AI and Robotics for Space (FAIR-SPACE, project ID: EP/R026092/1) and EU H2020 project Memory of Motion (MEMMO, project ID: 780684).

All authors are with the Institute of Perception, Action and Behaviour, School of Informatics, University of Edinburgh, Edinburgh EH8 9AB U.K. Email: {yiming.yang, wolfgang.merkt, v.ivan, sethu.vijayakumar}@ed.ac.uk.

ploring Random Tree (RRT), RRT-Connect [2], Probabilistic Roadmap (PRM [3]), Dynamic Roadmap [4], and many others [5]. However, sampling-based methods in general can not handle time-indexed constraints. Time-configuration space was proposed in [6], [7] for avoiding moving obstacles using a unidirectional search algorithm. As extending it to bidirectional search is non-trivial, the algorithm requires long planning times, e.g., several minutes for a 7-DoF system. Approaches such as velocity obstacles [8] are commonly used for generating collision-free trajectories in the presence of moving obstacles, but normally do not scale to complex environments. Similarly, sampling-based methods such as Anytime RRT* [9] address path planning problems in changing environments but do not scale to high-dimensional motion planning problems. An alternate approach is to use a swept volume approximation for continuous collision checking—this, however, is a highly restrictive approximation of occupied workspace disregarding the temporal aspect of the workspace occupation of moving objects. It further prohibits grasping objects that are contained within a moving box or shelf as common in real-world applications. Methods that require preprocessing of the environment or collision models further face the difficulty of changing collision geometry from picking up large objects.

Derivative-free, stochastic optimization-based approaches such as STOMP [10] which perform noisy roll-outs over trajectories that may be in collision deal with binary constraints. However, these are not time-indexed and finding a solution may be comparatively slow. CHOMP [11] uses covariant gradient descent and only takes differentiable constraints into account while replacing binary constraints with an approximation. T-CHOMP [12] is an extension of CHOMP to include time indexing for the trajectory functional; yet, it does not consider moving obstacles as the robot and environment require preprocessing to create collision representations suitable for optimization, which would have to be recreated for every time-index. Efficient trajectory optimization approaches such as KOMO [13] (using an approximated Hessian and exploiting its band-diagonal properties with a second-degree Newton method) and TrajOpt [14] (using sequential quadratic programming and convex decompositions of collision objects) only deal with differentiable, time-indexed constraints. In summary, trajectory optimization methods which utilize derivative information are suitable for type 1 constraints, while sampling-based algorithms are preferable for type 2.

Pick-and-place tasks in non-static environments, e.g., from inside of a box on a conveyor belt or moving vehicle is a type of problem that includes both differentiable and binary constraints with time-indexing, e.g., as shown in Fig. 1 for the case of a humanoid robot. In order to address such problem and enable pick-and-place of large objects in non-static environments, in this paper we propose a planning framework using a novel time-configuration space sampling-based planning algorithm that is able to efficiently find collision-free trajectories in complex and non-static environments. As part of this, we present a formal solution for time monotonicity for reverse queries in bidirectional sampling-based algorithms while adhering to first-order transition (velocity) constraints.

II. Problem Formulation

Incorporating the two types of constraints, we consider two types of problems: a *reaching problem*, i.e., reach to a desired configuration at a given time (II-A) where only type 2 binary constraints are considered; and a *generic pick-and-place problem* (II-B) where both type 1 and type 2 constraints are considered but activated at different parts along the trajectory. It is obvious that the first problem is a prerequisite for the latter. Note that the key challenge is that the environment contains large and complex obstacles and the task needs to be accomplished while both obstacles and targets are moving.

A. Reaching Problem in Time-Configuration Space

For a N Degree-of-Freedom (DoF) robot, let $\mathbf{q} \in \mathcal{C}$ be a state in the configuration space $\mathcal{C} \subseteq \mathbb{R}^N$. In the rest of the paper, vectors are denoted in bold font and absolute values are computed element-wise. Let $\mathcal{R}(\mathbf{q}) \subset \mathcal{W}$ denote the robot's rigid body posture in the workspace $\mathcal{W} \subset \mathbb{R}^3$ at configuration $\mathbf{q}$, and $\mathcal{O}(t) \subset \mathcal{W}$ the workspace region that is occupied by obstacles at time t. This implies that trajectories for the obstacles need to be specified in advance. The obstacle region in configuration space is defined as $\mathcal{C}_{obs}(t) = \{\mathbf{q} \in \mathcal{C} | \mathcal{R}(\mathbf{q}) \cap \mathcal{O}(t) \neq \emptyset\}$, and the collision-free region is defined as $\mathcal{C}_{free}(t) = \mathcal{C} \backslash \mathcal{C}_{obs}(t)$. For humanoid robots, we also need to consider balance constraint, thus let $\mathcal{C}_{valid}(t) = \mathcal{C}_{free} \cap \mathcal{C}_{balance}(t)$, where $\mathcal{C}_{balance}(t)$ is the manifold of statically balanced configurations at time t. For convenience, let $\mathbf{s} = \langle \mathbf{q}, t \rangle \in \mathcal{S}$ be the state in the time-configuration space $\mathcal{S} = \mathcal{C} \times \mathbb{R}_{\geq 0}$. The valid regions in the time-configuration space are defined as

$$\mathcal{S}_{valid} = \{\langle \mathbf{q}, t \rangle \in \mathcal{S} | \mathbf{q} \in \mathcal{C}_{valid}(t)\}. \tag{1}$$

Given start state $\mathbf{s}_0 \in \mathcal{S}_{valid}$ and goal state $\mathbf{s}_T \in \mathcal{S}_{valid}$ in the time-configuration space, together with the obstacles region over time $\mathcal{O}(t), t \in \mathbb{R}_{\geq 0}$, a reaching motion planning is defined as

$$\begin{aligned} \mathbf{q}_{[0:T]} &= \text{Reaching}(\mathbf{s}_0, \mathbf{s}_T, \mathcal{O}(t)) \\ s.t. \quad & \forall t \in [0, T] : \mathbf{q}_t \in \mathcal{C}_{valid}(t) \end{aligned} \tag{2}$$

B. Pick-and-Place Problem in Time-Configuration Space

The plan generated by (2), however, only works if the task is to simply reach the configuration $\mathbf{q}_T$ at t_T without other constraints. Now consider another scenario where the robot needs to not only move to $\mathbf{q}_T$, but also accomplish an extra grasping task during the movement. Let $\mathbf{y} = \Phi(\mathbf{q}) \in \mathcal{W}$ be the end-effector pose, where $\Phi(\cdot)$ is the forward kinematic mapping. Let $\mathbf{y}^*_{[t_s:t_g]} \in \mathcal{Y}$ denote the desired end-effector trajectory from t_s to t_g, where t_s and t_g are the start and end time of grasping phase, and $\mathcal{Y}$ is the set of all valid end-effector trajectories for accomplishing a certain grasping task. For example, Fig. 2 (right) shows a simple end-effector

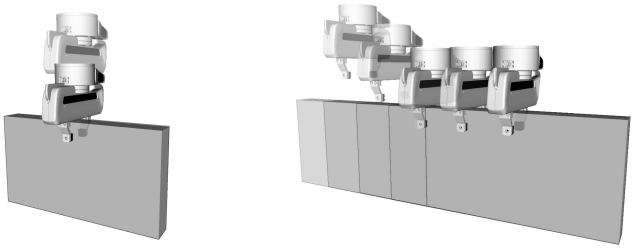

Fig. 2: *Left:* grasping trajectory in target frame; *right:* grasping trajectory in world frame while the target is moving. The relative pose between the target and gripper must be maintained during the grasping process.

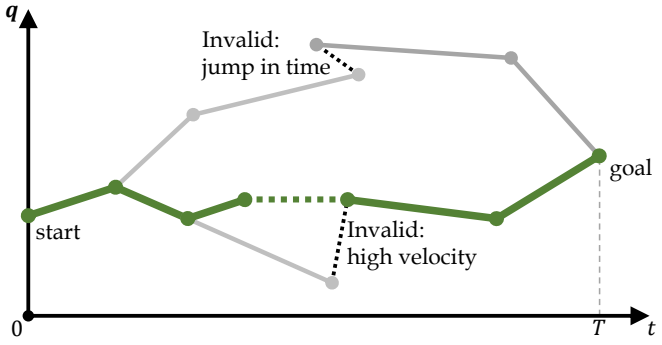

Fig. 3: Tree search in time-configuration space.

trajectory $\mathbf{y}^*_{[t_s:t_g]}$ for grasping a box target. We define the pick-and-place planning as

$$\begin{aligned} \mathbf{q}_{[0:T]} &= \text{PickAndPlace}(\mathbf{s}_0, \mathbf{s}_T, \mathcal{O}(t)) \\ s.t. \quad & \forall t \in [0, T] : \mathbf{q}_t \in \mathcal{C}_{valid}(t) \\ & \{\mathbf{y}_t = \Phi(\mathbf{q}_t) \mid t \in [t_s, t_g]\} \in \mathcal{Y} \end{aligned} \tag{3}$$

Note that the focus of this paper is motion planning, where $\mathbf{y}^*$ and $\mathcal{Y}$ are assumed given by a high-level grasp or task planner.

III. Methodology

While avoiding collision is usually the main constraint during reaching and placing phases, the grasping phase is dominated by dexterous and precise positioning of the end-effector, especially when the target itself is moving, as shown in Fig. 2. This is difficult to achieve using sampling-based approaches. Although task-constrained sampling methods have been proposed to address this issue [15], those techniques do not scale well to tasks that also include time-indexing. Furthermore, for redundant systems, the same object can be grasped by different end-effector trajectories in $\mathcal{Y}$ further complicating the problem, making it intractable for sampling-based methods. Thus, in this work, we split the problem into three sub planning problems: reaching, grasping, and placing, which is a common approach for solving pick-and-place tasks [16]. The reaching and placing problems can be formulated as two individual reach planning problems with time-indexing as formulated by (2); and the grasping problem is formulated as follows

$$\begin{aligned} \mathbf{q}_{[t_s:t_g]} &= \text{Grasping}(\mathbf{y}^*_{[t_s,t_g]}, \mathcal{O}(t)) \\ s.t. \quad & \forall t \in [t_s, t_g] : \Phi(\mathbf{q}_t) = \mathbf{y}^*_t \end{aligned} \tag{4}$$

which can be solved by efficient optimization solvers such as [17] and [18]. In this work, we focus on solving the reaching and placing problems in a non-static environment with time indexing, and using existing trajectory optimization solvers to generate the grasping trajectory. In the rest of this section, we explain first how to automatically find the grasping time window $[t_s, t_g]$ in III-A, and then solve the motion planning problem in time-configuration space in III-B, and finally solve the whole pick-and-place problem in III-C.

A. Grasping Time Selection and Completeness

For the pick-and-place framework to work, the grasping time window $[t_s, t_g]$ needs to be given as a prerequisite for other actions to proceed. Let $\mathbf{p}_{mid}$ be the location along the target object trajectory that is closest to the robot base at t_{mid}, which is set to be exactly at the middle of the grasping process. The closing/grasping time $t_{closing}$ is hardware dependent and assumed known for a given end-effector, e.g. two-finger gripper, thus the grasping time window can be obtained,

$$t_s = t_{mid} - \frac{1}{2} t_{closing}, \tag{5}$$

and

$$t_g = t_{mid} + \frac{1}{2} t_{closing}. \tag{6}$$

Note that this is a simple approach for automatically determining the grasping window that works in most scenarios but does not guarantee that the calculated $[t_s, t_g]$ will be always valid, i.e. there exists scenarios where no valid pick-and-place trajectory can be found for the given $[t_s, t_g]$. In other words, the pick-and-place framework is incomplete.[1] However, we want to emphasis here that, while the whole pick-and-place framework is incomplete, the motion planning algorithm proposed in III-B is probabilistically complete as it preserves such property from the original RRT-Connect. The completeness of the pick-and-place framework can be recovered by assuming a complete grasping planner that can provide all possible grasping trajectories $\mathcal{Y}$ and the corresponding grasping windows.

B. Reaching in a Non-Static Environment

In general, geometric sampling-based planners such as RRT, RRT-Connect, and PRM are independent of the search space, i.e., they have the ability to solve problems with different search spaces without the need for changing the method. However, this generality does not apply to the time-configuration space. By adding time into the search space, velocity has been added implicitly. This is in contrast

[1] An algorithm is considered complete if for any input it correctly reports whether there is a solution or not. If a solution exists, it must return one in finite time. A weaker notion called probabilistic completeness is used for sampling-based algorithms. It posits that as the number of samples goes to infinity, the probability of finding a solution converges to one.

with kino-dynamic planning where the velocity is handled explicitly. As shown in Fig. 3, the velocity can be represented by the slope of the line which connects two neighboring states. There are two major features when sampling in time-configuration space: (1) if two states are very close in time dimension but far in configuration, the edge is invalid as such an edge generates very large velocities; (2) the path can only advance in one direction in the time dimension (time monotonicity), otherwise the path would jump in time which is invalid. The sampling procedure needs to be adjusted in order to avoid these two artifacts. As these adjustments may vary across different methods, we have chosen to focus on RRT-Connect in this work as it has been shown empirically as one of the most generic and efficient sampling-based methods [4].

The uniform joint velocity required to transit from state $\mathbf{s}_a$ to state $\mathbf{s}_b$ is defined as

$$\mathbf{v}_{ab} = \begin{cases} \frac{|\mathbf{q}_b - \mathbf{q}_a|}{t_b - t_a}, & t_a \neq t_b \\ \infty, & t_a = t_b, \mathbf{q}_a \neq \mathbf{q}_b \\ 0, & t_a = t_b, \mathbf{q}_a = \mathbf{q}_b \end{cases} \tag{7}$$

Although this is based on the assumption that joints can accelerate instantaneously, it is a suitable simplification of the problem as the proposed method constitutes a geometric planner. We deploy two distance functions: the *forward time distance* and the *reverse time distance*. Note that the sign of $\mathbf{v}_{ab}$ indicates the direction in time dimension, i.e., positive velocity means moving forward in time (valid during forward search) and negative velocity means moving backward in time (valid during reverse search). We require two metrics to facilitate the bi-directional search executed by the RRT-Connect algorithm. Given two states $\mathbf{s}_a$ and $\mathbf{s}_b$, the forward time distance is defined as

$$d_{forward}(\mathbf{s}_a, \mathbf{s}_b) = \begin{cases} \infty, & \mathbf{v}_{ab} \leq 0 \text{ or } \mathbf{v}_{ab} \geq \mathbf{v}_{max} \\ w_d \|\mathbf{q}_a - \mathbf{q}_b\| + w_v \|\mathbf{v}_{ab}\|, & \text{otherwise} \end{cases} \tag{8}$$

and the reverse time distance is defined as

$$d_{reverse}(\mathbf{s}_a, \mathbf{s}_b) = \begin{cases} \infty, & \mathbf{v}_{ab} \leq -\mathbf{v}_{max} \text{ or } \mathbf{v}_{ab} \geq 0 \\ w_d \|\mathbf{q}_a - \mathbf{q}_b\| + w_v \|\mathbf{v}_{ab}\|, & \text{otherwise} \end{cases} \tag{9}$$

where w_d and w_v are constant weighting factors. In simple scenarios, one can set $w_d = 1$ and $w_v = 0$—in this case, the above functions represent Euclidean distances.

The distance functions are then integrated into two different nearest neighbor tree structures: The *forward time tree* and *reverse time tree*, respectively. Given a set of existing states on the tree and a random state $\mathbf{s}_{rand}$, the nearest neighbor function should return the state with the minimum forward/reverse space-time distance to $\mathbf{s}_{rand}$. In cases where the nearest neighbor function returns a state with infinite distance, another random state needs to be sampled until the nearest neighbor function returns a neighboring state with a valid distance. So far, by changing the distance function and nearest neighbor structure, the default RRT-Connect planner should be able to solve planning problems in time-configuration space. However, such a naïve treatment would lead to a high probability for returned states to have an infinite distance resulting in high rejection rates and long planning times. Thus, if a new random state has no valid neighbor on the tree, we adjust the time component of the time-configuration space such that a given neighboring state can be reached within the velocity limits by passing the new random state into a `CorrectTime` function. Given a random state $\mathbf{s}_{rand} = \langle \mathbf{q}_{rand}, t_{rand} \rangle$ and the nearest neighboring state $\mathbf{s}_{near} = \langle \mathbf{q}_{near}, t_{near} \rangle$, the adjusted time t_{rand} can be obtained by

$$t_{rand} = \begin{cases} t_{rand}, & |t_{rand} - t_{near}| \geq t_{min} \\ t_{rand} + sign(t_{rand} - t_{near}) t_{min}, & \text{other} \end{cases}, \tag{10}$$

where

$$t_{min} = \max \left\{ \left| \frac{\mathbf{q}_{near} - \mathbf{q}_{rand}}{\mathbf{v}_{max}} \right| \right\} \tag{11}$$

is the minimum time required for the robot to transit from $\mathbf{s}_{near}$ to $\mathbf{s}_{rand}$ at the maximum allowed velocity $\mathbf{v}_{max}$. The RRT-Connect algorithm for solving time-configuration space problems is highlighted in Algorithm 1 (cf. [2] for more details on the general RRT-Connect algorithm). Note that in line 6, in scenarios where all existing states have infinity distance to the new random state, the nearest tree search returns the state that is closest (but not equal) to the random state in time dimension.

Algorithm 1 Time-Configuration Space RRT-Connect

Require: $\mathbf{s}_0$, $\mathbf{s}_T$, $\mathcal{O}(t)$
Ensure: Collision-free trajectory $\mathbf{q}_{[0:T]}$
1: $\mathcal{T}_{forward}$.Insert($\mathbf{s}_0$), $\mathcal{T}_{reverse}$.Insert($\mathbf{s}_T$)
2: $\mathcal{T}_{current} = \mathcal{T}_{forward}$
3: $\mathcal{T}_{other} = \mathcal{T}_{reverse}$
4: **while not** Terminate **do**
5: $\quad \mathbf{s}_{rand} = \text{SampleRandom}()$
6: $\quad \mathbf{s}_{near}, d = \mathcal{T}_{current}.\text{Nearest}(\mathbf{s}_{\text{rand}})$
7: $\quad$ **if** $d = \infty$ **then**
8: $\quad\quad \mathbf{s}_{rand} = \text{CorrectTime}(\mathbf{s}_{near}, \mathbf{s}_{rand})$
9: $\quad \mathbf{s}_{new}, \text{status} = \text{Extend}(\mathbf{s}_{near}, \mathbf{s}_{rand})$
10: $\quad$ **if** status **is not** Trapped **then**
11: $\quad\quad \mathbf{s}_{near} = \mathcal{T}_{other}.\text{Nearest}(\mathbf{s}_{new})$
12: $\quad\quad$ **if** $\mathbf{s}_{near}$ **is not** null **then**
13: $\quad\quad\quad \text{status} = \text{Connect}(\mathbf{s}_{near}, \mathbf{s}_{new})$
14: $\quad\quad\quad$ **if** status = Reached **then**
15: $\quad\quad\quad\quad$ **return** Path($\mathcal{T}_{forward}, \mathcal{T}_{reverse}$)
16: $\quad$ swap($\mathcal{T}_{current}, \mathcal{T}_{other}$)
17: **return** Failure

C. Pick-and-Place Planning in Non-Static Environments

The motion plans generated in III-B only work for transiting the robot to a desired goal configuration, but are unable to accomplish additional tasks during the movement. In the following section, we explain how to perform a grasping task during the movement as defined by (3). As illustrated

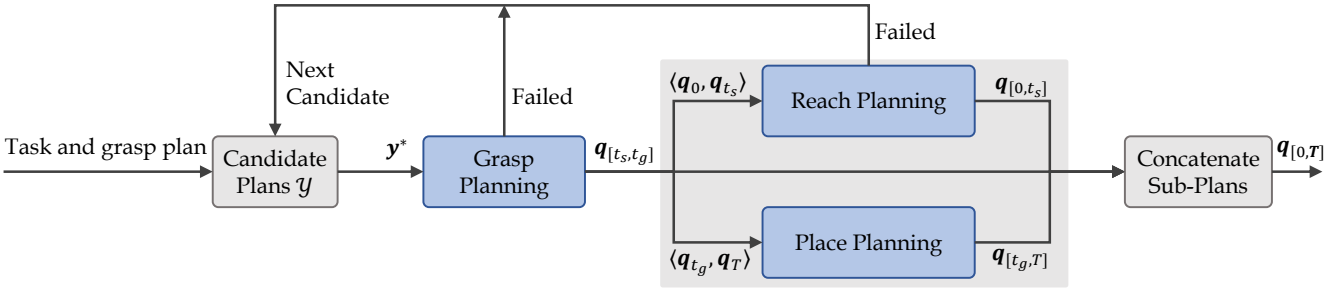

Fig. 4: Overview of the proposed pick-and-place pipeline.

in Fig. 4, we first select one candidate trajectory $\mathbf{y}^* \in \mathcal{Y}$ as the input to (4) to generate a grasping trajectory. This step is formulated as a trajectory optimization problem and can be solved by various non-linear programming solvers such as SNOPT [17] or AICO [18]. Including collision-avoidance into the optimization is non-trivial and the solver can easily get trapped in local minima when facing complex collision models even in static environments, not to mention moving obstacles. Thus, we do not consider collision-avoidance in the objective or constraint functions. The optimization solver is initialized with a random seed trajectory, followed by a binary collision-free validity check on the output grasping trajectory $\mathbf{q}_{[t_s,t_g]}$. A new $\mathbf{y}^*$ will be selected and tested if the optimizer returns failure or leads to a colliding trajectory.

After having found an optimal grasping trajectory $\mathbf{q}_{[t_s,t_g]}$, the reaching and placing problems are invoked with $\{\mathbf{q}_0, \mathbf{q}_{t_s}\}$ and $\{\mathbf{q}_{t_g}, \mathbf{q}_T\}$ as the start and goal states, respectively. A whole pick-and-place trajectory $\mathbf{q}_{[0,T]}$ can be created by concatenating $\mathbf{q}_{[0,t_s]}$,$\mathbf{q}_{[t_s,t_g]}$ and $\mathbf{q}_{[t_g,T]}$. Similarly, the process should restart with a different $\mathbf{y}^*$ if either reaching or place planning returns failure. Note that the grasped object is attached to the robot's end-effector and considered for collision checking during placing—this further complicates the planning problem especially when grasping large objects.

Finally, a post-planning simplification process can be applied to improve the smoothness of $\mathbf{q}_{[0,T]}$. We have used the standard path simplifier from the OMPL in our implementation [19]. Note that although the path simplifier was designed for configuration spaces without time indexing, it is still valid when used for time-configuration space. When trying to simplify a path, for instance as the scenario shown in Fig. 3, short-cutting any two non-neighboring states results in a new edge with a slope that is smaller than the maximum slopes of the original edges between the neighboring states, thus the maximum velocity constraint will not be validated during path simplification.

D. Pick-and-Place in Non-Static Environments while Walking

The proposed method is able to directly plan pick-and-place motion for fixed-base robotic arms as well as mobile manipulators. However, for more complex systems such as bipedal humanoids, it is unrealistic based on current methodology to directly plan a whole pick-and-place motion for the full-body, as such a system requires extra balance constraint which is difficult for sampling-based planners. While we have demonstrate in our previous work that we can efficiently plan collision-free end-poses with floating-base [20] and full-body motion with fixed-base [21], planning collision-free pick-and-place motion in non-static environments while the robot is walking is still a challenging problem.

We split the walking and pick-place into two subproblems instead of trying to solve them together. We assume that a lower-body trajectory $\mathbf{q}_{lower[0,T]}$ can be first planned using existing footstep or walking planners with the upper-body set to a default posture. We can then extract the floating base link (the pelvis) trajectory $\mathbf{p}_{pelvis[0,T]}$ using forward kinematics. During planning phase, for each sampled state $\mathbf{s} = \langle \mathbf{q}, t \rangle$, the lower-body and the pelvis are first set to the correct posture according to $\mathbf{q}_{lower[0,T]}$ and $\mathbf{p}_{pelvis[0,T]}$. The support polygon at t can be then calculated and a upper-body sample is valid only if it is collision-free and the centre of mass projection lies within the support polygon.

IV. Experiments

The proposed pick-and-place framework has been implemented within the Extensible Optimization Toolset (EXOTica) framework [22].[2] The time-configuration space RRT-Connect solver is implemented as an extension on top of the Open Motion Planning Library (OMPL), which is wrapped as part of EXOTica. Simulation evaluation was performed using an Intel Core i7-6700K 4.0 GHz CPU with 32GB 2133MHz RAM. Two different robot platforms were tested during the experiments: a 7-DoF KUKA LWR industrial manipulator, an omni-directional mobile manipulator with a 7-DoF Franka Emika Panda robotic arm (10-DoF in total) and a 38-DoF NASA Valkyrie humanoid robots. We have also conducted hardware experiments on the KUKA LWR robot showing that the proposed method is applicable to solve real-world problems.

A. Evaluation of the pick-and-place planning framework

We first conducted two experiments in simulation to evaluate the effectiveness and efficiency of the proposed

[2]The EXOTica library is an open source project for easy prototyping and benchmarking of planning and control algorithms. The source code of the proposed method can be obtained from `https://github.com/ipab-slmc/exotica`.

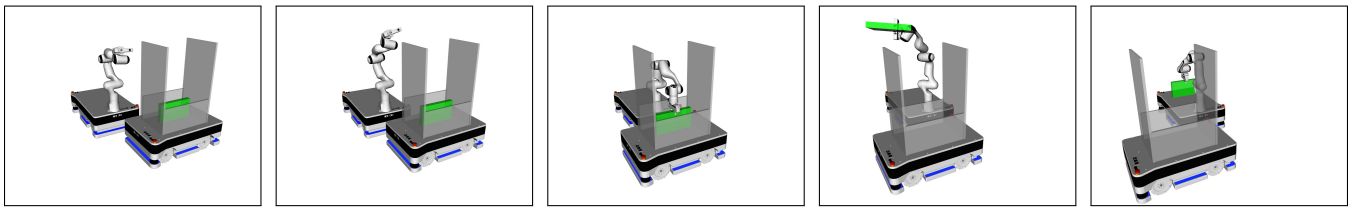

Fig. 5: Experiment R_1: reaching into moving box.

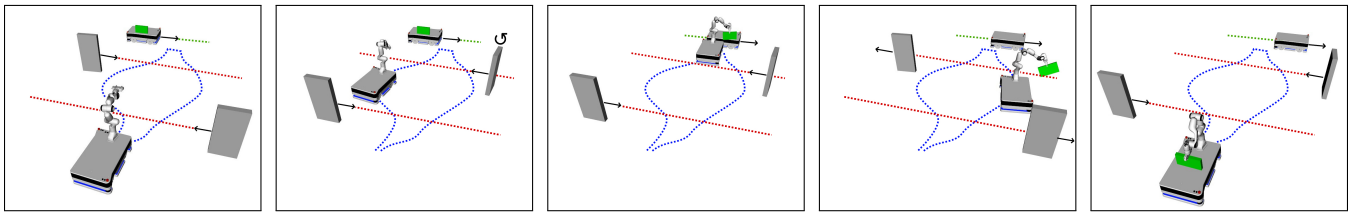

Fig. 6: Experiment R_2: fetching distant object in the presence of large moving obstacles. The red lines indicate the obstacles' movement and the blue lines show the base trajectories.

method. In experiment $R1$, as shown in Fig. 5, the robot arm fitted on the mobile base needs to pick-up a large green object placed inside a box on another mobile base which is passing by at 0.1 m/s. The robot base is fixed in this scenario, thus $N = 7$. In the second scenario $R2$, as shown in Fig. 6, the robot needs to coordinate the arm and base, i.e. $N = 10$, in order to fetch a distant object while the environment is populated with several large moving obstacles. The whole pick-and-place duration is set to 10 seconds, i.e. $T = 10$ and $t_{mid} = 5$. The grasping phase is then set to $[t_s = 4, t_g = 6]$ according to (5) and (6), resulting a $[0, t_s = 4]$ reaching phase and $[t_g = 6, T = 10]$ placing phase. The joint velocity limits are set to the maximum allowed velocity on the robot hardware (π rad/s for each arm joint and 1.5 m/s for the base). The average planning time over 100 trials is highlighted in Table I. The results show that the proposed planning method is capable of finding collision-free trajectories in time-configuration space very efficiently. As we have mentioned, the reaching and placing trajectories can be generated using the proposed time-configuration space planner, while the grasping is formulated as a trajectory optimization problem. The planning time of the grasping trajectory is subject to different algorithms and implementations of the optimization methods. In our experiments we use the Approximate Inference Control (AICO, [18]) solver implemented in the EXOTica framework. Note that the placing motion planning takes much longer to compute than the reaching motion as the large target object is attached to the robot end-effector thus further complicating the collision avoidance task.

We have further carried out a set of experiments evaluating how different joint velocity limits and the `CorrectTime` function can affect the planning time. The experiment R_1, i.e. reaching into the box, has been extended with four different joint velocity limits $\mathbf{v}_{max}$: π rad/s, $\frac{3}{4}\pi$ rad/s, $\frac{\pi}{2}$ rad/s and $\frac{\pi}{4}$ rad/s. The planning time result is highlighted in Table II, which suggests that, within a certain margin, e.g. π and $\frac{3}{4}\pi$, the planning time does not necessarily get affected by the joint velocity limits. When the maximum allowable joint velocity decreases more significantly, e.g. $\frac{\pi}{2}$ or $\frac{\pi}{4}$ m/s, the planning time increases with the decrease of joint velocity limits. In this case, the velocity limits create a state space with significant bottlenecks that are more costly to explore using randomized sampling. The result also shows that, especially with a limited maximum allowable joint velocity, the `CorrectTime` function does make a difference and thus improves the planning efficiency.

TABLE I: Pick-and-place motion planning time with standard deviations (in millisecond). The result is averaged over 100 runs for each scenario.

Experiment	Reaching	Grasping	Placing	Total
R_1, Fig. 5	9.2 ± 4.6	105.9 ± 5.8	28.2 ± 24.4	143.3
R_2, Fig. 6:	36.0 ± 23.6	142.9 ± 13	44.5 ± 37.2	223.4

TABLE II: Reach motion planning time with different joint velocity limits (in milliseconds), the last column shows the improvement factor by using the `CorrectTime` function. The result is averaged over 100 runs for each scenario.

Joint velocity limits (rad/s)	with `CorrectTime`	without `CorrectTime`	without/with
3.142 (π)	9.2	10.3	1.117
2.335 ($3\pi/4$)	9.5	10.6	1.119
1.571 ($\pi/2$)	13.3	15.1	1.138
0.785 ($\pi/4$)	29.6	48.9	1.653

B. Experiments on robot arm hardware

We have also conducted hardware experiments on the KUKA LWR robot to demonstrate that the proposed method is applicable for solving real-world problems, as shown in Fig. 7 and Fig. 8. The tested tasks include picking up objects that are placed inside of a box (H_1) or a shelf (H_2) on a

Fig. 7: Experiment on robot hardware H_1: picking up an object placed inside of a box on a moving conveyor belt.

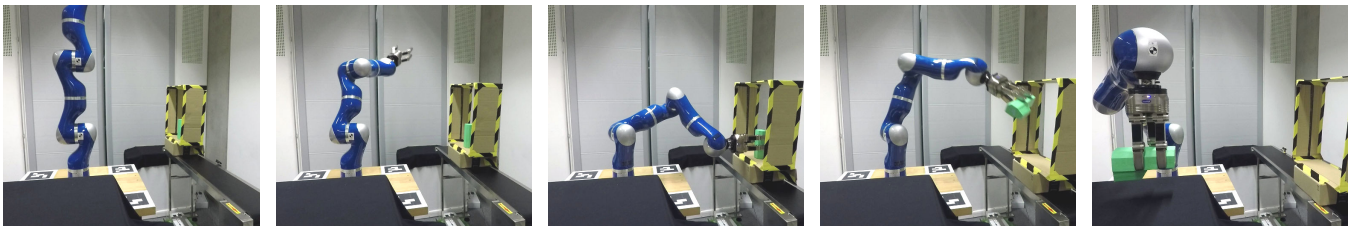

Fig. 8: Experiment on robot hardware H_2: fetching an object placed inside of a shelf on a moving conveyor belt.

moving conveyor belt. The whole pick-and-place process is set to $T = 10$ seconds with the conveyor belt moving at 0.1 m/s. We assume the model of the collision environment and $\mathcal{Y}$ are given. The robot is fitted with a Schunk Dexterous Hand which takes approximately 3 seconds to close, thus, the grasping phase is set to $[t_s = 4, t_g = 7]$. The main challenge here is that the robot hand and target objects are intentionally large for these highly constrained environments, where the motion needs to be carefully planned to avoid any possible collisions.

C. Pick-and-Place in Non-Static Environment while Walking

Last but not least, we have also deployed the proposed method on the NASA Valkyrie humanoid robot to accomplish pick-and-place tasks during walking in simulation. As described in III-D, we used an existing walking planner to generate a 20 second walking trajectory on a flat ground with a velocity of 0.13m/s. One mobile base is moving towards the robot at a velocity of 0.2m/s with a large target object placed inside a storage box, another mobile base with an empty storage box is crossing the robot's walking path also traveling at a speed of 0.2m/s. The task is to pick-up the target from the first storage box and put it into the other one. The key challenge here is that the robot will be walking along the planned pelvis trajectory during the whole pick-and-place phase. The grasping window is set to $7s - 9s$, the placing window is set to $18s - 20s$ and the final upper-body stopping time is set to $28s$, i.e., the walking stops at $20s$ but the upper-body takes an extra $8s$ to safely go to default posture. Snapshots of the motion is highlighted in Fig. 1 and Fig. 9. A supplementary video of the simulation and hardware experiments can be found at

`https://youtu.be/jhht2H8Dgqk`.

V. Conclusion

In this work, we have proposed a novel time-configuration space bidirectional sampling-based planning algorithm which is capable of efficiently searching for collision-free trajectories in complex and changing environments. By using the planning algorithm, we also introduced a pick-and-place planning framework with simulation and hardware experiments showing that the method is able to plan pick-and-place trajectories for grasping large objects in challenging scenarios. The proposed method is very efficient and scales to redundant robot arms as well as mobile manipulators with more than 10-DoF.

Meanwhile, the proposed method has certain limitations. First, the grasping trajectory $\mathbf{y}^*_{[t_s,t_g]}$, the grasping duration $[t_s, t_g]$, and the total time T are either pre-specified, or calculated by naïve assumption as in (5) and (6). While the grasping trajectories can be solved by interfacing to existing grasp planning methods, finding an appropriate grasping duration $[t_s, t_g]$ is non-trivial. Thus, future work includes automatic planning of the grasping duration, which is similar to end-pose planning [23] with the difference that a reachable trajectory, rather than only a reachable pose, needs to be found. Another limitation is that, although the proposed method limits the maximum velocity, it does not guarantee smoothness and optimality which may generate trajectories with non-smooth acceleration or jerk profiles. Such artifacts which are common to geometric sampling-based planning methods point out possible future work in extending the proposed method to kinodynamic planning or to incorporate it with trajectory optimization for generating smooth and optimal motion. Finally, an integration with a shared autonomy system such as [24] opens up the possibility for collaborative mobile manipulation applications and deployments.

References

[1] V. Nabat, M. de la O Rodriguez, O. Company, S. Krut and F. Pierrot, "Part4: very high speed parallel robot for pick-and-place," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 553-558.

[2] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 995-1001.

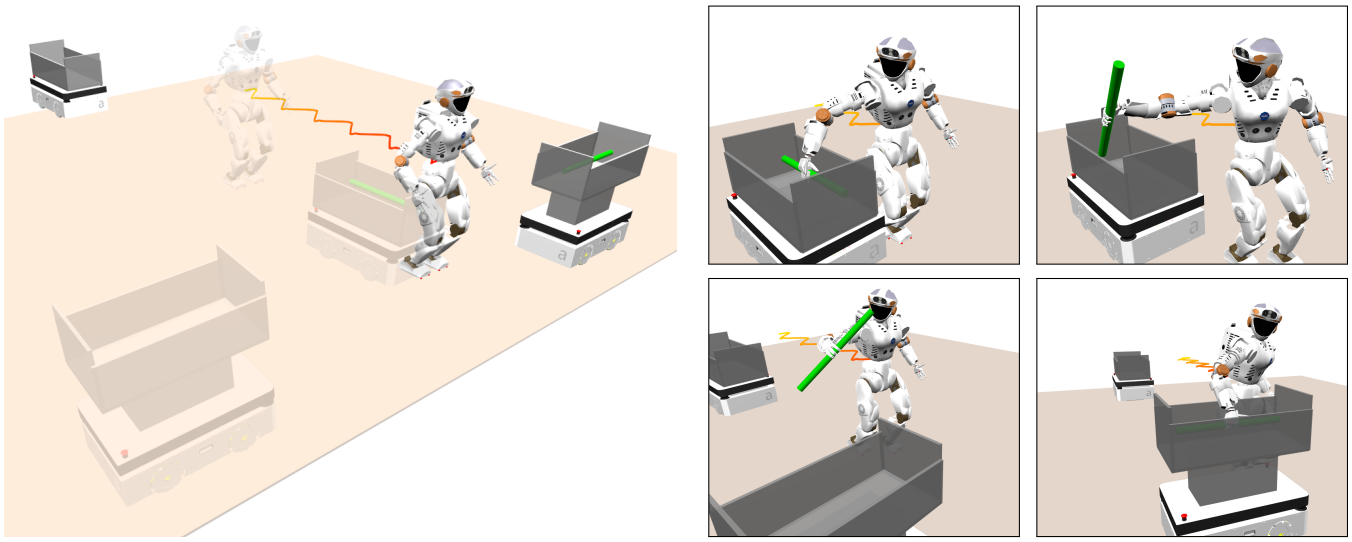

Fig. 9: Experiment on NASA Valkyrie humanoid robot picking up an object from inside a moving storage box and placing it into another moving storage box while walking. The first figure shows the environment setup and the right four figures highlight the key frames during pick-and-place. The colored line shows the pelvis trajectory during walking.

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transaction on Robotics and Automation*, vol. 12, no. 4, pp. 566-580, 1996.

[4] Y. Yang, W. Merkt, V. Ivan, Z. Li and S. Vijayakumar, "HDRM: A Resolution Complete Dynamic Roadmap for Real-Time Motion Planning in Complex Scenes," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 1, pp. 551-558, 2018.

[5] M. Elbanhawi and M. Simic, "Sampling-Based Robot Motion Planning: A Review," in *IEEE Access*, vol. 2, pp. 56-77, 2014.

[6] T. Fraichard, "Dynamic trajectory planning with dynamic constraints: A 'state-time space' approach," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1993, pp. 1393-1400.

[7] M. Cefalo, G. Oriolo, and M. Vendittelli, "Task-constrained motion planning with moving obstacles," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 5758-5763.

[8] P. Fiorini and Z. Shiller, "Motion Planning in Dynamic Environments Using Velocity Obstacles," *The International Journal of Robotics Research (IJRR)*, vol. 17, no. 7, pp. 760-772, 1998.

[9] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli and S. Teller, "Anytime Motion Planning using the RRT*," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1478-1483.

[10] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proceedings of IEEE International Conference on Robotics and Automation, Shanghai*, 2011, pp. 4569-4574.

[11] N. Ratliff, M. Zucker, J.A. Bagnell, S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 489-494.

[12] A. Byravan, B. Boots, S. Srinivasa, and D. Fox, "Space-time functional gradient optimization for motion planning," In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6499-6506.

[13] M. Toussaint, "Newton methods for k-order Markov constrained motion problems," *arXiv preprint*, arXiv:1407.0414, 2014.

[14] J. Schulman, J. Ho, A.X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization," *Robotics: Science and Ssystems (RSS)*, vol. 9, No. 1, pp. 1-10, 2013.

[15] D. Berenson, S. Srinivasa, and J. Kuffner,"Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research (IJRR)*, vol. 30 no. 12, pp. 1435-1460, 2011.

[16] T. Lozano-Perez, J. L. Jones, E. Mazer and P. A. O'Donnell, "Task-level planning of pick-and-place robot motions," in *Computer*, vol. 22, no. 3, pp. 21-29, 1989.

[17] P. Gill, W. Murray, and M. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Review*, vol. 47, no.1, pp. 99-131, 2005.

[18] K. Rawlik, M. Toussaint, S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," *Robotics: science and systems (RSS)*, vol. 13, no. 2, pp. 3052-3056, 2012.

[19] I. A. Sucan, M. Moll, L. E. Kavraki, "The Open Motion Planning Library", *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 7282, 2012.

[20] Y. Yang, V. Ivan, Z. Li, M. Fallon, S. Vijayakumar, "iDRM: Humanoid Motion Planning with Real-Time End-Pose Selection in Complex Environments", *Proceedings of IEEE RAS International Conference on Humanoid Robots (Humanoids)*, 2016.

[21] Y. Yang, V. Ivan, W. Merkt, S. Vijayakumar, "Scaling Sampling-based Motion Planning to Humanoid Robots", *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2016.

[22] V. Ivan, Y. Yang, W. Merkt, M. Camilleri and S. Vijayakumar, "EXOTica: An Extensible Optimization Toolset for Prototyping and Benchmarking Motion Planning and Control," *Robot Operating System Volume 3*, Springer, 2018.

[23] Y. Yang, W. Merkt, H. Ferrolho, V. Ivan and S. Vijayakumar, "Efficient Humanoid Motion Planning on Uneven Terrain Using Paired Forward-Inverse Dynamic Reachability Maps," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2279-2286, 2017.

[24] W. Merkt, Y. Yang, T. Stouraitis, C. E. Mower, M. Fallon, and S. Vijayakumar, "Robust Shared Autonomy for Mobile Manipulation with Continuous Scene Monitoring," In *Proceedings of IEEE Conference on Automation Science and Engineering (CASE)*, 2017, pp. 130-137.