BI^{2}RRT*: An Efficient Sampling-Based Path Planning Framework for Task-Constrained Mobile Manipulation

Felix Burget   Maren Bennewitz   Wolfram Burgard

***Abstract*— Mobile manipulators installed in warehouses and factories for conveying goods between working stations need to meet the requirements of time-critical workflows. Moreover, the systems are expected to deal with changing tasks, cluttered environments and constraints imposed by the goods to be delivered. In this paper, we present a novel planning framework for generating asymptotically optimal paths for mobile manipulators subject to task constraints. Our approach introduces the Bidirectional Informed RRT* (BI^{2}RRT*) that extends the Informed RRT* [1] towards bidirectional search and satisfaction of end-effector task constraints. In various experiments, we demonstrate the efficiency of BI^{2}RRT* for both unconstrained and constrained mobile manipulation planning problems. As the results show, our planning framework finds better solutions than Informed RRT* and Bidirectional RRT* in less planning.**

I. Introduction

Mobile manipulators are fast, dexterous robotic service systems with the ability of reliably and repetitively performing pick and place operations in warehouses and production facilities. Here, the time required by such systems to transfer an object from one location to another is a common decision criteria for their deployment. Often, the robotic systems handle pick and place tasks by decomposing them into a sequence of motion planning subproblems, *pick-transition-place* [2]. This decomposition allows to solve a set of independent low-dimensional planning problems, for the manipulator and the mobile base respectively, instead of solving a complex high-dimensional planning problem for the entire task at once. In particular, assuming a safe rest pose for the manipulator above the mobile base, the *transition* planning phase for the mobile manipulation task simplifies to a planar motion planning problem for a rigid box. Several asymptotically-optimal motion planning algorithms have been presented, among which rapidly-exploring random trees (RRT*) and probabilistic roadmaps (PRM*) [3] are the most popular. Advanced variants, such as Informed RRT* [1] achieve an enhanced convergence rate using intelligent sampling techniques. Using one of these planners, optimal solutions can, in principle, be generated for the individual subtask. The global optimal solution, however, does not equal the sum of the local optimal solutions found for the subtasks. In fact, each terminal robot pose resulting from a motion plan generated for a subtask directly affects the set of solutions available for the subsequent motion planning problems. In the worst case, it even leaves this set empty, hence preventing them to find a solution at all. The results are globally suboptimal paths or a high number of unsuccessful motion planning queries. Optimal manipulation planning for the entire task, on the other hand, would result in globally optimal paths and a higher success rate, but is highly computationally demanding. Doing so, requires to plan coordinated base-manipulator motions, which respect the joint limits, avoid self-collisions as well as collisions with obstacles in the environment. Furthermore, constraints imposed by the object to be manipulated must be taken into account (see Fig. 1).

F. Burget and W. Burgard are with the Institute of Computer Science, University of Freiburg, Germany. M. Bennewitz is with the Institute of Computer Science, University of Bonn, Germany. This work has been supported by the German Research Foundation (DFG) within the Cluster of Excellence *BrainLinks-BrainTools* (EXC1086) and by the European Commission under contract number FP7-610532-SQUIRREL.

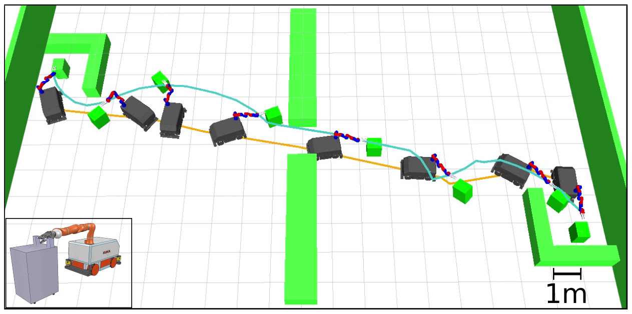

Fig. 1: Configuration sequence of a solution path generated by our BI^{2}RRT* planning framework for the complex task of maneuvering a cart out of a parking lot and into another. The orange and blue line represent the corresponding trajectory of the mobile base and end-effector.

In this paper we present a novel motion planning framework for task-constrained mobile manipulation that unifies asymptotically optimal sampling-based bidirectional path planning with informed sampling and task-constraint satisfaction. The new algorithm, *Bidirectional* Informed RRT* (BI^{2}RRT*), extends the Informed RRT* algorithm [1] towards bidirectional search, informed sampling, and satisfaction of geometric task constraints. Experiments with a mobile manipulator show that our approach is capable of generating low-cost solutions paths to complex constrained mobile manipulation tasks. We compared the performance of our planning framework to state-of-the-art path planning algorithms on several planning problems of varying complexity and demonstrate that our method generates low-cost solution paths more reliable and faster than existing methods.

II. Related Work

Karaman *et al.* introduced PRM* and RRT* [3], which are in comparison to their *probabilistically complete* counterparts also *asymptotically optimal*, guaranteeing that the cost of the returned solution approaches the optimum as the number of samples goes to infinity. These algorithms have the property of improving the solution in the available computation time, however, do not guarantee a high rate of convergence. Furthermore, Karaman *et al.* presented an extension of the RRT* algorithm toward anytime motion planning [4], interleaving planning with trajectory execution. The idea is to instantly trigger execution as soon as an initial feasible motion plan has been found whose remaining portions are improved over time. A two-tree variant of RRT* proposed by Jordan *et al.* [5] has shown to very rapidly provide solutions in planning problems with challenging regions such as narrow corridors, or high-dimensional configuration spaces with numerous obstacles. Janson *et al.* [6] proposed the FMT* algorithm to increase the efficiency of optimal path search by extending graph-search methods to sampling-based algorithms. This approach generates *resolution-optimal* solutions in an asymptotically manner. A further improvement of the solution, however, requires to restart the search from scratch at a higher resolution. Alterovitz *et al.* [7] introduced the rapidly-exploring roadmap algorithm that allows the user to explicitly control the trade-off between free-space exploration and solution path refinement. The authors show that a careful choice of this parameter enables finding optimal paths more quickly. Similarly, Akgun *et al.* [8] presented a bidirectional variant of RRT* with path biasing that quickly finds an initial solution path. Afterwards the planner predominantly spends the remaining planning time for either refinement of the current path or exploration of other homotopy classes depending on a user-defined parameter.

The claim for optimal solutions usually comes along with a high computational effort. Therefore, several heuristics have been introduced to guide the search in order to achieve a faster convergence of the path cost towards the optimal solution. Nasir *et al.* [9] developed RRT*-Smart that follows a similar principle as visibility graph techniques. To accelerate the rate of convergence, it generates nodes as close as possible to obstacle vertices instead of adopting a purely random exploration. While the approach quickly reduces the solution cost, it may also cause other homotopy classes to remain undiscovered. The Cloud RRT* algorithm, proposed by Kim *et al.* [10], allocates new samples from sampling clouds, initially generated from a generalized Voronoi diagram. During planning, these clouds are updated based on the set of configurations constituting better solutions. Additionally new clouds are added to refine the currently best solution. However, global sampling remains active throughout the planning phase in order to explore understudied homotopy classes. Otte *et al.* [11] introduced the use of parallel computing for optimal motion planning. Their C-FOREST algorithm grows configuration space trees on multiple CPUs, that propagate improvements found on the current solution among each other. By exchanging this information, the configuration space region from which samples are drawn is collaboratively shrunk to an area of configurations potentially improving the current solution. The Informed RRT* algorithm, proposed by Gammell *et al.* [1], follows a similar approach of focused asymptotically-optimal motion planning. Here, the current solution is used to define an ellipsoidal subset of the planning domain, which is used to draw only samples that potentially improve the solution. The increased rate of convergence towards the optimal solution achieved in [11] and [1], however, vanishes for planning problems where most of the planning time needs to be dedicated for finding a first solution.

The BIT* algorithm [12] tries to overcome these limitations by combining sampling-based planning with incremental graph search techniques. Instead of shrinking, the ellipsoidal subset is incrementally enlarged and batches of configurations are added, which are subsequently used to update a search graph. Typically, large batch sizes are required to quickly converge to the optimal solution which, in the presence of task constraints, would require to compute a constraint manifold approximation in advance.

Efficient optimal motion planning for mobile manipulation, requires collision-free samples not only to be drawn from particular regions of the configuration space, but also to comply with task constraints imposed on the end-effector by the object to be manipulated. Şucan *et al.* proposed to pre-compute an approximation of the constraint manifold prior planning [13]. In this way, valid samples can be directly drawn from an approximation graph for tree expansion instead of applying computationally expensive rejection sampling or sample projection techniques. The drawback is that multiple approximation graphs are needed in order to perform planning for different types of constraints. Stilman *et al.* presented three modifications to the RRT algorithm for planning with task-space constraints, namely *randomized gradient descent* (RGD), *tangent-space sampling* (TS) and *first-order retraction* (FR) [14]. The results of their comparison indicated that the FR method, which iteratively displaces a sample toward the constraint manifold, is faster and more invariant to expansion step size and error tolerance than the RGD and TS technique. Berenson *et al.* [15] described an approach for planning manipulation tasks using Task Space Regions (TSR). These regions define upper and lower bounds for constraint task coordinates and can be linked together in order to obtain more complex constraints. Here, samples are projected onto a region, as opposed to [14], where samples are required to be projected onto a single task space point. This leads to a faster and more successful projection of samples, thus generating solutions in a shorter amount of time. Alternative approaches for task constrained mobile manipulation using graph-search techniques instead of sampling-based motion planning are presented in [16] and [17]. These implementations use anytime repairing A* to generate motions for opening a door or pushing a cart. The discretization based on general motion primitives leads in many cases to dramatic performance improvements in sampling-based planning algorithms. The main drawback is that these approaches are resolution-complete and provide only resolution-optimal paths.

In this work, we combine the bidirectional RRT* [5] with informed sampling from an ellipsoidal subset [1] and the first-order retraction technique [14] for sample projection [15]. In this way, we make use of the advantages of several existing techniques to achieve an increased rate of convergence also for highly constrained planning problems.

III. Motion Planning With Bidirectional Informed RRT*

Our system builds upon the Informed RRT* algorithm by Gammell *et al.* [1], which has already demonstrated the ability to find optimal solutions to planning problems in high-dimensional domains. As the authors concluded, only the addition of further samples from an ellipsoidal subset of the planning domain can lead to an improvement of the current solution path. Thus, directly sampling from this subset can result in a faster rate of convergence. In the following, we describe our bidirectional variant of Informed RRT*, which speeds up the search for a first solution and improves the convergence rate towards the optimal solution also for highly complex planning problems.

A. The BI^{2}RRT* Algorithm

Informed RRT* achieves an increased rate of convergence once an initial solution is found. Before that, it basically coincides with the classical RRT* planner. Obviously, the earlier an initial solution is found the more planning time is left for informed path refinement. Therefore, we developed a *two-tree* variant of Informed RRT*, which quickly generates a first solution, in turn triggering informed sampling at an earlier planning stage. Furthermore, we consider *two* ellipsoidal subsets, used to perform informed sampling for the translational and rotational component of the configuration space of a mobile manipulator platform.

Alg. 1 shows the pseudocode of our BI^{2}RRT* algorithm. For simplicity, we first explain the basic functionality of our planner in the absence of end-effector task constraints as given by $\mathbf{t}_{cc}$ and $\mathbf{B}$. How those constraints are considered is afterwards described in Sec. III-B and Sec. IV. Given a start and goal end-effector pose $\mathbf{p}_s^e$, $\mathbf{p}_g^e$, our algorithm generates a start and goal configuration for the root nodes n_s, n_g of the two trees $\mathcal{T}_a$, $\mathcal{T}_b$, respectively (Lines 1, 2 in Alg. 1, details in Sec. III-B). Afterwards, the current solution path cost c_{SP} is initialized, and the algorithm starts growing the search trees until either a maximum number of iterations or time elapsed is reached (Lines 3, 4). The SAMPLE_RAND_CONFIG function (Line 5, see Alg. 2) returns a configuration $\mathbf{q}_{rand}$, randomly sampled from the entire configuration space or the informed ellipsoidal subset (the latter is detailed in Sec. III-C), depending on whether a solution path is already available or not. Afterwards, FIND_NEAREST_NEIGHBOR finds the nearest neighbor $\mathbf{q}_{nn}$ in $\mathcal{T}_a$, to the configuration $\mathbf{q}_{rand}$ (Line 6). The EXTEND function (see Alg. 3) tries to establish a connection between the two configurations by incrementally stepping from $\mathbf{q}_{nn}$ towards $\mathbf{q}_{rand}$ (Line 3 in Alg. 3) while performing collision checks for the intermediate configurations (Line 8 in Alg. 3). As a result, the EXTEND operation returns a list of segments $\mathcal{V}$, i.e.

Algorithm 1: BI^{2}RRT* ($\mathbf{p}_s^e$, $\mathbf{p}_g^e$,$\mathbf{t}_{cc}$,$\mathbf{B}$, k_{nv})

```
1  $n_s.q \leftarrow$ GEN_IK_SOLUTION($\mathbf{p}_s^e$, $\mathbf{t}_{cc}$, $\mathbf{B}$)
2  $n_g.q \leftarrow$ GEN_IK_SOLUTION($\mathbf{p}_g^e$, $\mathbf{t}_{cc}$, $\mathbf{B}$, $n_s.q$)
3  $\mathcal{T}_a.init(n_s)$; $\mathcal{T}_b.init(n_g)$; $c_{SP} \leftarrow \infty$
4  for $i = 1$ to max_time_iter do
5      $\mathbf{q}_{rand} \leftarrow$ SAMPLE_RAND_CONFIG($c_{SP}$, $n_s.q$, $n_g.q$)
6      $\mathbf{q}_{nn} \leftarrow$ FIND_NEAREST_NEIGHBOR($\mathcal{T}_a$, $\mathbf{q}_{rand}$)
7      $\mathcal{L}_{nv}$, $\mathcal{V}_a$, $\mathcal{V}_b \leftarrow \emptyset$
8      $ext\_state_{nn}$, $ext\_state_{nv}$, $ext\_state_c \leftarrow$ FAILED
9      $ext\_state_{nn}$, $\mathcal{V}_a \leftarrow$ EXTEND($\mathcal{T}_a$, $\mathbf{q}_{nn}$, $\mathbf{q}_{rand}$, $\mathbf{t}_{cc}$, $\mathbf{B}$)
10     if ($c_{SP} < \infty$ or $ext\_state_{nn}$ = FAILED) then
11         $\mathcal{L}_{nv} \leftarrow$ FIND_NEAR_NODES($\mathcal{T}_a$, $\mathbf{q}_{rand}$)
12         $ext\_state_{nv}, \mathcal{V}_a \leftarrow$ BP_SEARCH($\mathcal{T}_a$, $\mathbf{q}_{rand}$, $\mathcal{L}_{nv}^{k_{nv}}$)
13     if ($ext\_state_{nn} \neq$ FAILED or $ext\_state_{nv} \neq$ FAILED) then
14         INSERT_SEGMENTS($\mathcal{T}_a$, $\mathcal{V}_a$)
15         if $c_{SP} < \infty$ then
16             $\mathcal{T}_a \leftarrow$ REWIRE_TREE($\mathcal{L}_{nv}^{k_{nv}}$, $\mathcal{T}_a$.last.q)
17             $c_{SP} \leftarrow$ RECURSIVE_COST_UPDATE($\mathcal{T}_a$)
18         $\mathbf{q}_{nn} \leftarrow$ FIND_NEAREST_NEIGHBOR($\mathcal{T}_b$, $\mathcal{T}_a$.last.q)
19         $ext\_state_c, \mathcal{V}_b \leftarrow$ EXTEND($\mathcal{T}_b$, $\mathbf{q}_{nn}$, $\mathcal{T}_a$.last.q, $\mathbf{t}_{cc}$, $\mathbf{B}$)
20         if $ext\_state_c \neq$ FAILED then
21             INSERT_SEGMENTS($\mathcal{T}_b$, $\mathcal{V}_b$)
22             if $ext\_state_c$ = REACHED then
23                 $c_{SP,new} \leftarrow$ GET_SOLUTION_COST($\mathcal{T}_a$.last.q,
                       $\mathcal{T}_b$.last.q)
24         if $c_{SP,new} < c_{SP}$ then
25             $c_{SP} \leftarrow c_{SP,new}$
26     SWAP($\mathcal{T}_a$, $\mathcal{T}_b$)
27 if $c_{SP} < \infty$ then
28     return PATH($\mathcal{T}_a$, $\mathcal{T}_b$)
29 else
30     return FAILURE
```

Algorithm 2: SAMPLE_RAND_CONFIG (c_{SP}, $\mathbf{q}_s$, $\mathbf{q}_g$)

```
1  conf_valid $\leftarrow$ FALSE
2  while conf_valid = FALSE do
3      if $c_{SP} < \infty$ then
4          $\mathbf{q}_{rand,\,p} \leftarrow$ SAMPLE_FROM_ELLIPSE($c_{SP,p}$, $\mathbf{q}_{s,p}$, $\mathbf{q}_{g,p}$)
5          $\mathbf{q}_{rand,\,r} \leftarrow$ SAMPLE_FROM_ELLIPSE($c_{SP,r}$, $\mathbf{q}_{s,r}$, $\mathbf{q}_{g,r}$)
6          $\mathbf{q}_{rand} \leftarrow [\mathbf{q}_{rand,\,p}, \mathbf{q}_{rand,\,r}]$
7      else
8          $\mathbf{q}_{rand} \leftarrow$ GET_RAND_CONFIG()
9      conf_valid $\leftarrow$ IS_CONFIG_VALID($\mathbf{q}_{rand}$)
10 return $\mathbf{q}_{rand}$
```

pairs of nodes and edges, together with information about the expansion status, *ext_state* (Line 9 in Alg. 1). The value of *ext_state* corresponds to either *FAILED*, *PROGRESS*, or *REACHED*, indicating that tree $\mathcal{T}_a$ has not been extended at all, made some progress towards $\mathbf{q}_{rand}$, or has reached $\mathbf{q}_{rand}$, respectively. As opposed to the classical Informed RRT*, our planner performs a best parent search, defined by BP_SEARCH, not only when a solution is available, but also when the tree expansion from $\mathbf{q}_{nn}$ failed to make some progress (Line 10-12 in Alg. 1). Here, the function FIND_NEAR_NODES provides a set $\mathcal{L}_{nv}^{k_{nv}}$ of k_{nv} near nodes, stored in the order of ascending *cost-to-reach*. If either the expansion from $\mathbf{q}_{nn}$ or from one of the near nodes, stored in $\mathcal{L}_{nv}^{k_{nv}}$, has made at least some progress towards $\mathbf{q}_{rand}$, the segments $\mathcal{V}_a$ returned by EXTEND are added to the tree $\mathcal{T}_a$ (Line 14 in Alg. 1). Lines 15-17 in Alg. 1 correspond to the standard rewire operation of RRT*, considering the last node added to the tree $\mathcal{T}_a$.last.q and the k_{nv} near nodes of $\mathcal{L}_{nv}^{k_{nv}}$ with the highest *cost-to-reach*. Note, that the

Algorithm 3: EXTEND ($\mathcal{T}$, $\mathbf{q}_{nn}$, $\mathbf{q}_{rand}$, $\mathbf{t}_{cc}$, $\mathbf{B}$)

```
1  ext_state ← FAILED ; $\mathcal{V} \leftarrow \emptyset$ ; conf_valid ← TRUE
2  while conf_valid = TRUE do
3      $\mathbf{q}_{ext}, \mathbf{e}_{ext}$ ← STEP_TOWARDS_SAMPLE($\mathcal{T}$, $\mathbf{q}_{nn}$, $\mathbf{q}_{rand}$)
4      if $\mathbf{t}_{cc} = \mathbf{0}$ or $\mathbf{q}_{ext} = \mathbf{q}_{rand}$ then
5          $\mathbf{q}_{new}, \mathbf{e}_{new} \leftarrow \mathbf{q}_{ext}, \mathbf{e}_{ext}$
6      else
7          $\mathbf{q}_{new}, \mathbf{e}_{new}$ ← FR_NEW_CONFIG($\mathbf{q}_{ext}$, $\mathbf{t}_{cc}$, $\mathbf{B}$)
8      conf_valid ← IS_CONFIG_VALID($\mathbf{q}_{new}$)
9      if conf_valid = TRUE then
10         $\mathbf{q}_{nn} \leftarrow \mathbf{q}_{new}$
11         ext_state ← PROGRESS
12         $\mathcal{V}$ ← ADD_SEGMENT($\mathbf{q}_{new}, \mathbf{e}_{new}$)
13         if $\mathbf{q}_{new} = \mathbf{q}_{rand}$ then
14             ext_state ← REACHED
15             break
16 return ext_state, $\mathcal{V}$
```

reduction of the *cost-to-reach* for a single node needs to be propagated through the tree to the leaf nodes. By doing so, an improvement of the current solution path c_{SP} may be elicited (Line 17 in Alg. 1). The last part of the planner, described in Lines 18-20 of Alg. 1, basically corresponds to the classical *connect* step of bidirectional search. First, the nearest configuration $\mathbf{q}_{nn}$ in tree $\mathcal{T}_b$, to the last node added to $\mathcal{T}_a$ is determined (Line 18 in Alg. 1). Afterwards, EXTEND tries to find a new solution path by connecting $\mathbf{q}_{nn}$ to $\mathcal{T}_a$.last.q. If EXTEND returns *REACHED*, a new solution path is found. Its cost $c_{SP,new}$, however, does not necessarily constitute a better solution. Therefore, we finally compare $c_{SP,new}$ to the cost of the current best solution c_{SP}. If $c_{SP,new}$ is found to be lower than c_{SP}, c_{SP} is updated, the trees $\mathcal{T}_a$ and $\mathcal{T}_b$ are swapped and the algorithm proceeds with the next iteration. After reaching the stopping criteria, Alg. 1 returns either the final solution path or *FAILURE*.

B. Tree Initialization

We assume the initial and desired end-effector poses $\mathbf{p}_s^e$ and $\mathbf{p}_g^e$, as needed for an object manipulation task, to be given. Our algorithm allows for specifying a vector $\mathbf{t}_{cc}$ of constrained task coordinates and a matrix $\mathbf{B}$ of task coordinate bounds (see Sec. IV-A), e.g., to describe task constraints corresponding to carrying objects in an upright pose. The vector $\mathbf{t}_{cc}$ is simply constituted by binary values indicating whether a task coordinate is constrained. For those coordinates being constrained, the matrix $\mathbf{B}$ defines a lower and upper admissible displacement from $\mathbf{p}_s^e$ and $\mathbf{p}_g^e$, respectively. Considering these constraints, the GEN_IK_SOLUTION function (Line 1 in Alg. 1) finally runs a damped least-squares controller [18] using random configuration space seeds to generate a valid inverse kinematics solution for the mobile manipulator end-effector pose $\mathbf{p}_s^e$. For the generation of an inverse kinematics solution for $\mathbf{p}_g^e$, we use seeds that are generated by Gaussian sampling around the start configuration $n_s.q$ (Line 2 in Alg. 1). This approach yields two advantages. First, it is easier for the planner to find a solution path when the terminal configurations are similar. Second, we avoid initializations of the planner in mobile manipulation tasks with terminal configurations that lie in disjoint regions of the constraint manifold, thus making it impossible to find a valid solution.

C. Informed Heuristic for Mobile Manipulators

Gammell *et al.* [1] proposed to directly sample from an ellipsoidal subset, once a solution is found. Here, the uniformly distributed samples in a hyper-ellipsoid are calculated from

$$\mathbf{x}_{\hat{f}} = \mathbf{R}\mathbf{L}\mathbf{x}_{ball} + \mathbf{x}_{centre}, \tag{1}$$

where $\mathbf{x}_{ball}$ are uniformly distributed samples from the unit n-ball and $\mathbf{x}_{centre}$ is the centre of the hyper-ellipsoid, following from two focal points $\mathbf{x}_g$, $\mathbf{x}_s$. The matrices $\mathbf{R}$ and $\mathbf{L}$ are given by

$$\mathbf{R} = \mathbf{U}\,\mathsf{diag}\{1, \ldots, 1, \det(\mathbf{U})\det(\mathbf{V})\}\,\mathbf{V}^T, \tag{2}$$

where $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ follows from the singular value decomposition of a matrix $\mathbf{M}$, given by the outer product $\mathbf{a}_1\mathbf{I}_1^T$. Here, $\mathbf{I}_1$ represents the first column of the identity matrix and $\mathbf{a}_1$ is given by

$$\mathbf{a}_1 = (\mathbf{x}_g - \mathbf{x}_s)/\|\mathbf{x}_g - \mathbf{x}_s\|_2, \tag{3}$$

and

$$\mathbf{L} = \mathsf{diag}\left\{\frac{c_{SP}}{2}, \frac{\sqrt{c_{SP}^2 - c_{HS}^2}}{2}, \ldots, \frac{\sqrt{c_{SP}^2 - c_{HS}^2}}{2}\right\}, \tag{4}$$

with c_{SP} and c_{HS} describing the current and hypothetical solution path cost, respectively. Here, we defined c_{HS} as the cost of the linear interpolation between the start and goal configuration, neglecting obstacles and task constraints.

So far, informed sampling has been performed in configuration spaces solely composed of either revolute or prismatic components [1], [12]. In this work, we split the configuration space of the mobile manipulator into these components in order to perform informed sampling for each of them. The result are two components $\mathbf{q}_{rand,p}$ and $\mathbf{q}_{rand,r}$ generated from distinct hyper-ellipsoids (Lines 4, 5 in Alg. 2), which are stacked after sampling (Line 6 in Alg. 2) in order to obtain an informed configuration space sample for the entire robot.

IV. Efficient Sample Projection According to Task Constraints

Considering end-effector task constraints in mobile manipulation planning, requires an advanced sample projection method within the EXTEND function (Line 7 in Alg. 3). We therefore apply the *first-order retraction* method [14], which has already shown to be faster and more invariant to expansion step size (used in STEP_TOWARDS_SAMPLE, Line 3 in Alg. 3) and error tolerance than other projection methods. Additionally, we adopt the idea of constraining task coordinates to remain in bounded intervals instead of considering fixed values [15]. In the following, we briefly recapitulate the definition of task constraints before explaining in more detail how task constraints are satisfied within our planning framework.

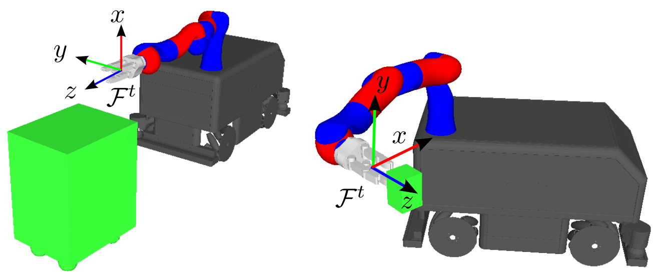

Fig. 2: Task frame for pulling a cart (left) and transporting liquids (right) used for the specification of end-effector task constraints in the first-order retraction sample projection method.

A. *Definition of Task Constraints*

For the definition of end-effector task constraints, we introduce the notions *task frame* $\mathcal{F}^t$, *coordinate constraint vector* $\mathbf{t}_{cc}$, and *task coordinate bounds* $\mathbf{B}$. The task frame represents a reference coordinate system, in which $\mathbf{t}_{cc}$ and $\mathbf{B}$ are expressed (see Fig. 2). The binary elements of $\mathbf{t}_{cc}$ are used to compose a diagonal constraint selection matrix of allowed translational and rotational displacement from $\mathcal{F}^t$

$$\mathbf{C} = \mathsf{diag}(t_{cc,1}, \dots, t_{cc,n}), \tag{5}$$

with $t_{cc,i} \in [0, 1]$ and $n = 6$. The matrix $\mathbf{B}$ additionally defines lower and upper bounds for the displacement for those task coordinates constrained by $\mathbf{C}$,

$$\mathbf{B} = \begin{bmatrix} \Delta x_{neg} & \Delta x_{pos} \\ \Delta y_{neg} & \Delta y_{pos} \\ \Delta z_{neg} & \Delta z_{pos} \\ \Delta \alpha_{neg} & \Delta \alpha_{pos} \\ \Delta \beta_{neg} & \Delta \beta_{pos} \\ \Delta \gamma_{neg} & \Delta \gamma_{pos} \end{bmatrix}, \tag{6}$$

where the first and second column refers to the admissible negative and positive deviation of the coordinates from $\mathcal{F}^t$. Here, α, β and γ are used to denote the roll, pitch, and yaw angles. Note that we initialize the task frame $\mathcal{F}^t$ globally with the desired start pose of the end-effector frame $\mathbf{p}^e_s$. For manipulation of articulated objects, a parametrized task frame would be required, instead.

B. *Satisfaction of Task Constraints*

After having sampled a random configuration $\mathbf{q}_{rand}$, the EXTEND function iteratively perform steps towards $\mathbf{q}_{rand}$ from the nearest neighbour $\mathbf{q}_{nn}$ until the random sample is reached or an invalid configuration is encountered (Line 3, 8 in Alg. 3). A premise for the extension of a tree considering end-effector task constraints is a successful projection of each intermediate configuration $\mathbf{q}_{ext}$ onto the constraint manifold (Line 7 in Alg. 3). To do so, FR_NEW_CONFIG first computes the end-effector frame $\mathcal{F}^e$ by forward kinematics for $\mathbf{q}_{ext}$. The result is given by the transformation $\mathbf{T}^0_e(\mathbf{q}_{ext})$. Afterwards, the displacement of $\mathcal{F}^e$ with respect to the task frame $\mathcal{F}^t$ is found by

$$\mathbf{T}^t_e(\mathbf{q}_{ext}) = \mathbf{T}^t_0 \mathbf{T}^0_e(\mathbf{q}_{ext}) = (\mathbf{T}^0_t)^{-1} \mathbf{T}^0_e(\mathbf{q}_{ext}), \tag{7}$$

where $\mathbf{T}^0_t$ is assumed to be a fixed transformation between the task frame $\mathcal{F}^t$ and world frame $\mathcal{F}^0$. Next, we need to represent the transform for end-effector displacement with respect to the task frame in task coordinates using the roll, pitch, yaw representation for describing the relative orientation

$$\Delta \mathbf{x} \equiv \mathbf{T}^t_e(\mathbf{q}_{ext}). \tag{8}$$

Given the matrix $\mathbf{C}$ from Eq. 5, we now select the relevant error components from $\Delta \mathbf{x}$ as follows

$$\Delta \mathbf{x}^c_{err} = [e^c_1, \dots, e^c_n] = \mathbf{C} \Delta \mathbf{x}, \tag{9}$$

$$e^c_i = \begin{cases} 0, & c_i = 0 \\ \Delta \mathbf{x}_i, & c_i = 1 \end{cases} \tag{10}$$

where c_i is the i-th element along the diagonal of matrix $\mathbf{C}$. Finally, we consider the admissible negative and possible deflection intervals for the task coordinates in order to obtain the task error components subsequently used in the sample projection method,

$$\Delta \mathbf{x}^b_{err} = [e^b_1, \dots, e^b_n], \tag{11}$$

$$e^b_i = \begin{cases} 0, & b_{i1} \le \Delta \mathbf{x}^c_{err,i} \le b_{i2} \\ \Delta \mathbf{x}^c_{err,i}, & \Delta \mathbf{x}^c_{err,i} < b_{i1} \text{ or } b_{i2} < \Delta \mathbf{x}^c_{err,i} \end{cases} \tag{12}$$

where b_{i1} and b_{i2} are the values in the first and second column of the i-th row in the task coordinate bound matrix $\mathbf{B}$ (see Eq. 6).

Once the task error is identified, we need to find a mapping that generates joint motions, iteratively reducing the error until all its components are within the bounded intervals defined by $\mathbf{B}$. This is done by a Jacobian-based method. The classical Jacobian $\mathbf{J}^0$ is a matrix of partial derivatives relating joint space velocities to end-effector linear and angular velocities expressed in the world frame $\mathcal{F}^0$. As the task space error is defined w.r.t. the frame $\mathcal{F}^t$ in our case, we need to respesent the Jacobian in this frame. The corresponding task frame Jacobian $\mathbf{J}^t$ is obtained using the inverse rotation matrix $\mathbf{R}^0_t$ as follows

$$\mathbf{J}^t = \begin{bmatrix} \mathbf{R}^t_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^t_0 \end{bmatrix} \mathbf{J}^0. \tag{13}$$

The lower three rows of $\mathbf{J}^t$ represent the mapping to angular velocities of the end-effector. However, the end-effector angular velocity with respect to the task frame is not given by the rotational velocity of a set of orientation angles. Though, it is possible to find a relationship between the angular velocity and rotational velocities for a given set of orientation angles. For RPY angles, this relationship is defined by a matrix $\mathbf{E}_{rpy}$ [14] yielding

$$\mathbf{J} = \mathbf{E}_{rpy}(\mathbf{q}_{ext}) \mathbf{J}^t(\mathbf{q}_{ext}). \tag{14}$$

Finally, we use the pseudo-inverse of the task Jacobian $\mathbf{J}^\dagger$ to map the task error $\Delta \mathbf{x}^b_{error}$ expressed in frame $\mathcal{F}^t$ to the least-norm velocities in joint space required to correct it:

$$\dot{\mathbf{q}} = \mathbf{J}^\dagger \Delta \mathbf{x}^b_{error}. \tag{15}$$

Note, that in IS_CONFIG_VALID (Line 8 in Alg. 3) a configuration $\mathbf{q}_{new}$ is found invalid not only if it is in collision, but also when it is equal to $\mathbf{q}_{nn}$ (*backprojection*) or has not made progress towards $\mathbf{q}_{rand}$ (*divergence*).

V. Implementation Details

Our planner is implemented in the *MoveIt!* framework in ROS [19] and uses FCL [20] for collision checks and and will be available open source at `https://github.com/burgetf/mobile_manipulation_planning`. When the planner checks a configuration for validity, the collision mesh model of each robot link is tested for self-collisions and collisions with the environment, considering also objects potentially attached to the end-effector. Computation of the forward kinematics for single configurations is done using the *KDL* library [21]. For computing the pseudo-inverse of the task frame Jacobian in the first-order retraction method we use the singular value decomposition implemented in the *Eigen* library [22].

VI. Experiments

For the experimental evaluation of our approach, we use the *omniRob* omnidirectional mobile manipulator platform by Kuka Robotics, which is composed of 10 degrees of freedom. Its configuration is given by

$$\mathbf{q} = (\mathbf{q}_{base}, \mathbf{q}_{manip})^T, \tag{16}$$

where

$$\mathbf{q}_{base} = (x, y, \theta), \tag{17}$$

$$\mathbf{q}_{manip} = (q_1, q_2, q_3, q_4, q_5, q_6, q_7), \tag{18}$$

denote the configuration of the base platform and manipulator chain, respectively. For planning mobile manipulation tasks, we modeled the planar motion of the base by two prismatic joints representing the translation of the robot in the x, y direction w.r.t. the world frame. For planning, we considered all DOFs of the robot, resulting in a 10-dimensional configuration space. The solution path cost c_{SP} is defined by the sum of the rotational and prismatic path length. Note that we adopted in our comparative experiments anytime variants of the standard RRT, Informed RRT and RRT-CONNECT planning algorithms in order to allow them to exploit the available planning time for finding alternative solution paths of lower cost. We derived these algorithms by selecting different combinations of the features *bidirectional search*, *informed sampling*, *tree optimization* (continuing the search after a first solution is found) in our planning framework. For the following experiments, planning was performed off-board on a single core of a standard desktop CPU (Intel Core i7, 3.4 GHz).

A. Planning Collision-Free Motions

In a first experiment, we conducted a quantitative comparison between our planning framework and other variants of RRT-based planning algorithms regarding the performance in planning collision-free motions for a complex scenario (Fig. 3, first column). Here, the robot needed to travel from an initial configuration to a final configuration while avoiding self-collisions and collisions with objects in the environment. Each planner performed 100 runs, each given a maximum planning time of 2 minutes. According to the averaged results, the bidirectional variants generally find better solutions than the unidirectional variants in less time and more often. Moreover, our planning framework finds a first solution faster compared to the Informed RRT* due to the earlier activation of informed sampling (Fig. 3, first column, fourth row), thus having more time left for the optimization of the current solution.

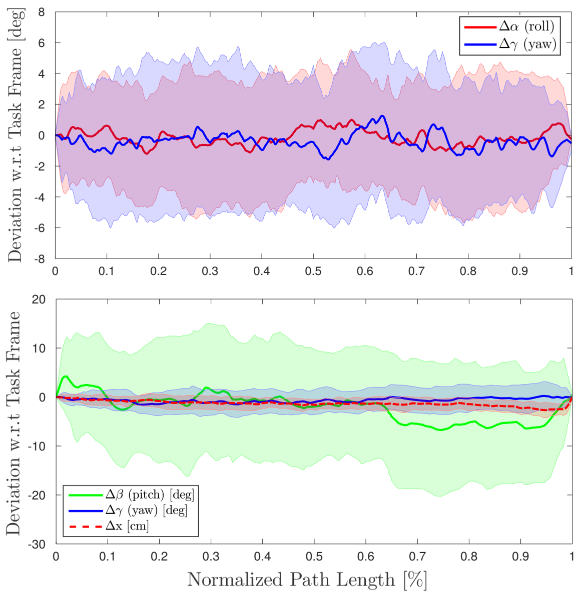

Fig. 4: Average deviation of the constrained task coordinates from the task frame along the end-effector solution path for 100 runs of the transportation of liquids (top) and cart pulling (bottom) planning scenario.

B. Transportation of Liquids

In the next planning scenario, we evaluated the performance of our planner w.r.t. other RRT-based planning algorithms in the presence of serious end-effector orientation constraints. Here, the robot needed to transport a container of liquid from one location to another (Fig. 3, second column). To do so, it needed to maneuver the container out of a narrow space, then underneath a ceiling joist, and finally through a narrow gate, while respecting the task constraints. Again, each planner performed 100 runs, this time each given a maximum planning time of 3 minutes. The coordinate constraint vector for this task is defined w.r.t. the task frame, depicted on the right side in Fig. 2, as follows

$$\mathbf{t}_{cc} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}^T. \tag{19}$$

Accordingly, we set the admissible negative and positive deflection from the task frame for the roll angle $\Delta\alpha_{neg}$, $\Delta\alpha_{pos}$ and yaw angle $\Delta\gamma_{neg}$, $\Delta\gamma_{pos}$ in matrix $\mathbf{B}$ to $-10°$ and $10°$, respectively. Note that we discarded the results of RRT and Informed RRT in the figures for this scenario due to their low performance. Regarding the time required to find a

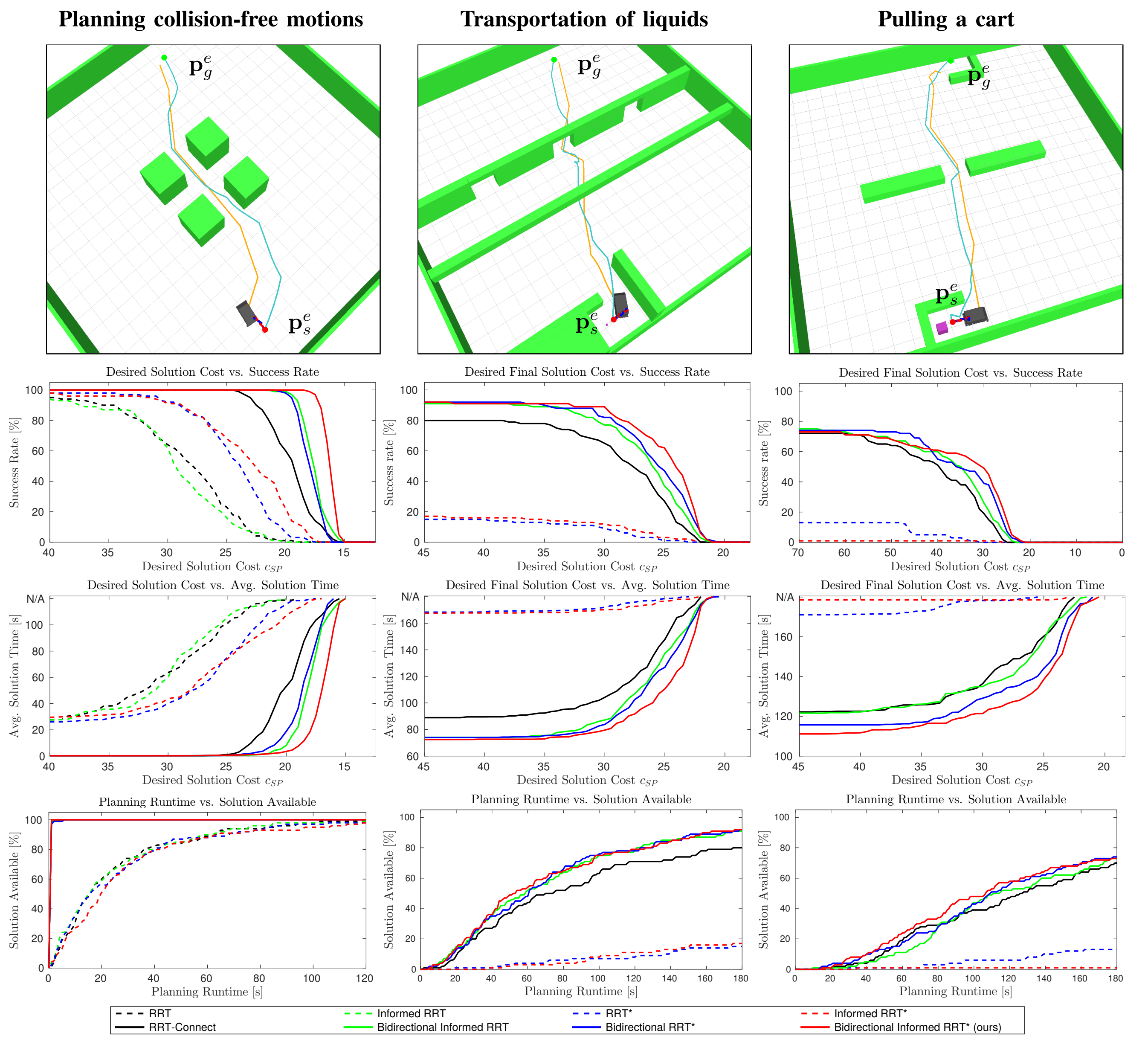

Fig. 3: Comparison of the performances in planning collision-free motions (2min planning time, first column), transportation of liquids (3min planning time, second column) and cart pulling (3min planning time, third column), averaged over 100 runs. Example solution paths are represented by their corresponding base (orange line) and end-effector (blue line) trajectory. First Row: Planning scenarios. Second Row: Percentage of successful runs w.r.t. different desired solution path costs. Third Row: Average time required to generate solutions of specific costs. Fourth Row: Percentage of runs providing a solution as a function of the planning runtime.

first solution (Fig. 3, second column, fourth row), our results show that about 70% of the runs of bidirectional planners provide a solution after 50% of the total planning time of three minutes. Furthermore, the combination of the greedy *connect* heuristic with informed sampling in our planning framework has shown to find better solution more often and in less time. The Informed RRT*, on the other hand, needed to dedicate most of the planning time for the discovery of an initial solution, thus having almost no time left for further path improvements (Fig. 3, second column, second and third row). The mean and standard deviation for the displacement of the constrained orientation angles from the task frame along the solution paths is depicted in Fig. 4. As can be seen, both angles stay within the bounds, defined in $\mathbf{B}$.

C. Pulling a Cart

In a final experiment, we quantitatively evaluated the performance of different planning algorithms considering end-effector position *and* orientation constraints. Here, the robot needed to maneuver a cart equipped with omnidirectional wheels out of a parking lot and into another, while respecting the aforementioned constraints (Fig. 3, third column). To compare the results of our planning framework we resorted to the same set of planning algorithms considered in Sec. VI-B. The coordinate constraint vector for this task is defined w.r.t. the task frame, depicted on the left side in Fig. 2 as follows

$$\mathbf{t}_{cc} = \begin{bmatrix}1 & 0 & 0 & 0 & 1 & 1\end{bmatrix}^T. \quad (20)$$

Here, we set the admissible negative and positive deflection from the task frame for the x-direction Δx_{neg}, Δx_{pos} and yaw angle $\Delta\gamma_{neg}$, $\Delta\gamma_{pos}$ in $\mathbf{B}$ to ± 3 cm and $\pm 5°$, respectively. The pitch angle, on the other hand, is allowed to rotate the end-effector around the cart handle by $\Delta\beta_{neg}$ =-30° and $\Delta\beta_{pos}$ =30°. As our results show, the unidirectional variants RRT* and Informed RRT* fail to find a solution in most of the planning runs. In contrast, bidirectional planning algorithms are able to generate a solution in about 70% of the runs (Fig. 3, third column, fourth row). Moreover, the few solutions generated by the unidirectional planning algorithms are found shortly before the available planning time runs out, whereas their bidirectional counterparts offer solutions of the same cost in a much earlier planning stage (Fig. 3, third column, third row). Regarding the success rate for different desired solution path costs and the corresponding average time required to generate them, the Bidirectional RRT* algorithm shows a similar performance as our planner. Here, Bidirectional RRT* has a slightly higher success rate for solution paths of cost $c_{SP} > 42$, whereas our planner is superior in generating lower cost solution paths due to the integration of informed sampling (Fig. 3, third column, second and third row). Furthermore, we evaluated the average deviations of the constrained coordinates from the task frame. Here, we obtain $\Delta\bar{x}$ =-1.32 cm±1.19cm SD for the x-coordinate, $\Delta\bar{\beta}$ =-2.16°±11.73° SD for the pitch angle and $\Delta\bar{\gamma}$ =-0.61°±2.39° SD for the yaw angle (see Fig. 4). Note that the compliance of the robot's kinematic structure and the object to be manipulated needs to be taken into account when defining the task coordinate bounds in $\mathbf{B}$.

VII. Conclusions

In this paper, we presented a novel framework for mobile manipulation planning under arbitrary geometric end-effector task constraints. Our BI^{2}RRT* planning algorithm uses the greedy *connect* heuristic to quickly find a first solution. To enable informed sampling for the full configuration space of a mobile manipulator, we propose to use two hyper-ellipsoids, representing subsets for the rotational and prismatic components of the configuration space. In this way, the solution can be improved if time allows. For constraint satisfaction, we adopted the first-order retraction method [14], which has shown to be a fast technique for projecting samples onto the constraint manifold. The experiments reveal that our planner generates solutions to complex mobile manipulation problems that satisfy all the desired constraints, e.g., to deliver a glass of water or a tool trolley. Moreover, we demonstrated in the evaluation on different planning scenarios that our approach is capable of providing low-cost solution paths more reliably and faster than existing state-of-the-art RRT-based algorithms.

References

[1] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," *arXiv preprint arXiv:1404.2334*, 2014.

[2] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao, "Perception, planning, and execution for mobile manipulation in unstructured environments," *IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation*, vol. 19, 2012.

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[4] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011, pp. 1478–1483.

[5] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, Tech. Rep. MIT-CSAIL-TR-2013-021, August 2013.

[6] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, p. 0278364915577958, 2015.

[7] R. Alterovitz, S. Patil, and A. Derbakova, "Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011, pp. 3706–3712.

[8] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011, pp. 2640–2645.

[9] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad, "RRT*-SMART: A rapid convergence implementation of RRT*," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[10] D. Kim, J. Lee, and S.-e. Yoon, "Cloud RRT: Sampling cloud based RRT," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014, pp. 2519–2526.

[11] M. Otte and N. Correll, "C-FOREST: Parallel shortest path planning with superlinear speedup," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 798–806, 2013.

[12] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015, pp. 3067–3074.

[13] I. A. Şucan and S. Chitta, "Motion planning with constraints using configuration space approximations," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[14] M. Stilman, "Global manipulation planing in robot joint space with task constraints," *IEEE Trans. on Robotics and Automation*, vol. 26, no. 3, pp. 576–584, 2010.

[15] D. Berenson, J. Chestnutt, S. Srinivasa, J. Kuffner, and S. Kagami, "Pose-constrained whole-body planning using task space region chains," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2009.

[16] S. Chitta, B. Cohen, and M. Likhachev, "Planning for autonomous door opening with a mobile manipulator," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[17] J. Scholz, S. Chitta, B. Marthi, and M. Likhachev, "Cart pushing with a mobile manipulation system: Towards navigation with moveable objects," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011, pp. 6115–6120.

[18] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Transactions on Control Systems Technology*, vol. 2, no. 2, pp. 123–134, 1994.

[19] S. Chitta, I. Sucan, and S. Cousins, "MoveIt! [ROS topics]," *IEEE Robotics Automation Magazine*, vol. 19, no. 1, pp. 18 –19, 2012.

[20] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[21] R. Smits, "KDL: Kinematics and Dynamics Library," http://www.orocos.org/kdl.

[22] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.