To appear in *Advanced Robotics*
Vol. 00, No. 00, Month 201X, 1–22

FULL PAPER

Anytime Motion Planning for Prehensile Manipulation in Dense Clutter

Andrew Kimmel, Rahul Shome, Kostas Bekris
(*Received 00 Month 201X; accepted 00 Month 201X*)

Many methods have been developed for planning the motion of robotic arms for picking and placing, ranging from local optimization to global search techniques, which are effective for sparsely placed objects. Dense clutter, however, still adversely affects the success rate, computation times, and quality of solutions in many real-world setups. The proposed method achieves high success ratio in clutter with anytime performance by returning solutions quickly and improving their quality over time. The method first explores the lower dimensional end effector's task space efficiently by ignoring the arm, and build a discrete approximation of a navigation function. This is performed online, without prior knowledge of the scene. Then, an informed sampling-based planner for the entire arm uses Jacobian-based steering to reach promising end effector poses given the task space guidance. This process is also comprehensive and allows the exploration of alternative paths over time if the task space guidance is misleading. This paper evaluates the proposed method against alternatives in picking or placing tasks among varying amounts of clutter for a variety of robotic manipulators with different end-effectors. The results suggest that the method reliably provides higher quality solution paths quicker, with a higher success rate relative to alternatives.

Keywords: manipulation; asymptotically-optimal motion planning; clutter)

1. Introduction

A variety of robotic tasks, such as warehouse automation and service robotics, motivate computationally efficient and high-quality solutions to prehensile manipulation. Typical applications involve the need to pick and place a variety of objects from tabletop or shelf-like storage units. Such manipulation challenges are demonstrated in Fig. 1.

Consider a tabletop and shelf manipulation challenge, as shown in Fig. 2(*left*). Many existing methods are capable of finding collision-free motions in such tabletop challenges [1] [2] [3]. Prior work [4] has shown that large amounts of clutter can significantly reduce the viable valid grasps on an object. While traditional strategies successfully find solutions in the tabletop setting, more cluttered environments like the shelf results in a massive degradation of performance, as highlighted in Fig. 2(*right*). One possible explanation is that the clearance of the solutions found in cluttered scenes is low, where clearance refers to the minimum distance between the robot geometries and the rest of the environment. This work accordingly refers to scenes with low clearance as *densely cluttered*, which have narrow passages in the workspace that make the motion planning problem harder. Nevertheless, coming up with high-quality motion planning solutions to such challenges in a reasonable amount of time remains an important objective.

In problems where solutions exist in cluttered workspace, there is a need to effectively guide the search process. The key insight is that in manipulation tasks, goals are typically end-effector centric i.e., grasping configurations. Consequently, heuristic guidance in manipulation tasks should reason about the end-effector. Additionally, the planning algorithm has to be designed to effectively use such guidance to find high-quality solutions quickly.

There has been significant recent interest [5] [6] [7] in applying machine learning techniques to high-dimensional motion planning problems. Components of motion planning that can be learned include local maneuvers (i.e. sequences of controls/actions with durations) and effective

Figure 1. Typical real world manipulation scenarios in table and shelf environments, where the dense clutter surrounding the grasped object creates several challenging aspects in the planning problem.

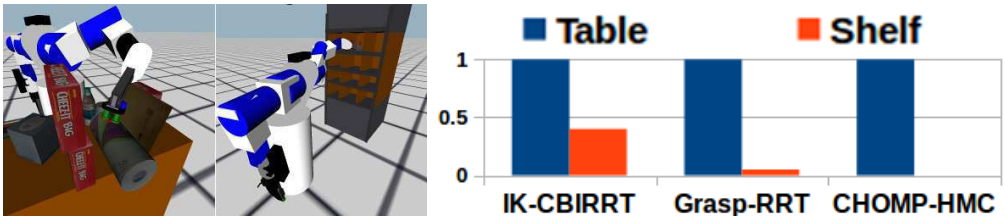

Figure 2. (left) Two manipulation challenges in a simulated table and shelf environment. (right) The success rate of three motion planning methods `IK-CBiRRT`[8], `Grasp-RRT`[9], and `CHOMP`[2] typically used in such scenarios is evaluated in these environments, and the average over 50 trials is reported. A success is counted when the planner returns a solution within 30 seconds.

heuristics. With such an objective in mind, a desirable motion planner must have an abstractions for using maneuvers and heuristics. In order to train such a learning framework, the generation of data in these cluttered scenarios first needs the algorithm to solve a vast majority of these problems effectively. Such an algorithm, can then not only generate the data efficiently but also improve over time using the learned maneuvers and heuristics, thereby yielding a way to frame the motion planning challenge in clutter into a lifelong learning and execution loop.

Towards solving such a challenge, this work proposes the Jacobian Informed Search Tree (`JIST`) method, which is a heuristic-guided sampling-based search algorithm for prehensile manipulation planning in the presence of dense clutter. The key idea is to plan around the constraints induced on the end-effector by the presence of clutter, so as to guide the motions of the arm towards areas which lead to a solution. The features of this algorithm include:

- designing and exploiting an effective heuristic for guiding the arm by solving the motion planning problem for just the end effector;
- applying the pseudo-inverse Jacobian as a steering process which allows the arm to be guided by the heuristic; and
- incorporating task space guided maneuvers (i.e. controls for the arm) effective in clutter into an asymptotically optimal motion planner.

Experiments first indicate that the local primitive, which uses the pseudo-inverse of the manipulator's Jacobian, is computationally efficient and has high success ratio in coming up with high-quality local paths. Then, the performance of `JIST` is compared against other state-of-the-art planners for picking and placing tasks, including both trajectory optimization (`CHOMP-HMC`) and sampling-based methods (`Grasp-RRT`, `IK-CBiRRT`). The results indicate that `JIST`, in contrast to the alternatives, can quickly and reliably compute high-quality solutions in the presence of dense clutter in unknown scenes, for a variety of robotic platforms, without requiring parameter tuning.

This archival version of `JIST` expands on prior work [10], and provides a more complete picture of the approach with several improvements and additions.

(1) The experimental section has been extended to evaluate the effectiveness of different steering methods motivating the choice of the pseudo-inverse Jacobian as the desired steering

process.
(2) Two additional cluttered environments have also been included in the experiments on a *Motoman* robotic manipulator.
(3) An execution on the real *Motoman* robotic manipulator is demonstrated.
(4) A brief discussion of the algorithmic properties (i.e. completeness and optimality) of the proposed algorithm is provided, along with a proof sketch.

2. Background and Relative Contribution

The proposed method draws inspiration from both broad categories of approaches for planning arm motion: a) local optimization and b) global search.

Local optimization follows a locally valid gradient towards finding a solution. *Artificial potential fields* incrementally move the current robot configuration by following such a gradient towards the goal. For arm planning [11], workspace virtual forces are mapped to torques through the manipulator's Jacobian. The framework allows for a hierarchy of objectives that a redundant arm can potentially try to satisfy [12]. The typical limitation is the presence of local minima, which can be avoided through integration with a global planner [13] or defining a navigation function [14]. The latter is difficult to find for complex geometric problems [15]. `JIST` uses gradient information to locally guide the exploration of the arm's path. This is performed in the context of a global search process so as to achieve stronger guarantees.

Trajectory optimization methods define a gradient over the entire trajectory, and are capable of finding smooth solutions for high DOF systems in sparse setups [2] [16] [17]. Such methods use precomputed signed distance fields for defining obstacle avoidance gradients in the workspace. The accuracy and cost of computing these fields depend upon resolution. `JIST` similarly aims for high-quality solutions but avoids dependency on parameters and better handles clutter.

Global heuristic search approaches aim to search the configuration space of a robotic arm given a discretization, frequently by focusing the search in the most promising subset or projection of that space [18] [19] [20]. Heuristics that have been used in the context of manipulation tasks include reachability [21] or geometric task-based reasoning [22]. `JIST` first solves the manipulation problem for a free-flying end effector geometry, and then uses this as a heuristic during the search in the arm's configuration space.

Sampling-based planners [23] [24] aim to scale better in high dimensions, while providing guarantees, such as asymptotic optimality [25] [26]. They incrementally explore the arm's configuration space through sampling. A variety of sampling-based planners has been developed for robotic arms [27] [28] [29], some of which use the pseudo-inverse of the Jacobian matrix for steering [3] or inverse kinematics solvers [30]. Using the pseudo-inverse Jacobian has been shown to be a probabilistically complete projection onto the grasping manifold [31]. The current method follows the sampling-based framework to maintain desirable guarantees, while properly guiding the search to quickly acquire high-quality solutions in terms of end-effector's displacement.

Some approaches deal with integrated grasp and motion planning [32] [33] [34]. This work focuses on motion planning aspects and uses the output of grasping planners [35] [36] [37] [38], i.e., grasps to set the goals of the corresponding picking challenges, which are frequently defined as end effector poses relative to a target object.

3. Problem Setup

Consider a robot arm with d degrees of freedom, which must solve pick and place tasks involving a target object o in a workspace with a set of static obstacles O.

3.1 *Goal Task Space Motion Planning*

Definition 1 (*C*-space)**.** *An arm's configuration q is a point in the robot's C-space:* $\mathrm{C}_{full} \subset \mathrm{R}^d$*. The colliding C-space (*$\mathrm{C}_{obs} \subset \mathrm{C}_{full}$*) corresponds to configurations where the arm collides with* O *or o in their detected poses. The free C-space is then defined as:* $\mathrm{C}_{free} = \mathrm{C}_{full} \setminus \mathrm{C}_{obs}$*.*

The motion planning problem to be solved in manipulation setups is frequently not defined directly in C_{free}. Instead, the task relates to the arm's end effector pose.

Definition 2 (Pose Space)**.** *The end effector's pose space* $\mathrm{E} \subseteq \mathrm{SE}(3)$ *corresponds to the reachable subset of poses for the arm's end effector.*

The pose space E can be computed given the forward kinematics of the robotic arm $FK : \mathrm{C}_{full} \rightarrow \mathrm{E}$. The inverse kinematics function corresponds to the inverse mapping: $IK : \mathrm{E} \rightarrow \mathrm{C}_{full}$. For redundant manipulators, $dim(\mathrm{E}) < dim(\mathrm{C})$, so IK describes a mapping to self-motion manifolds of the manipulator [31], composed of states, which all have the same end effector pose.

Definition 3 (Goal-Constrained Task Space (GCTS))**.** *Given a set of goal poses* $\mathrm{E}_{goal} \subset \mathrm{E}$ *for the end effector, the arm's goal task space* $\mathrm{Q}_{goal} \subset \mathrm{C}_{full}$ *denotes the robot configurations that bring the end effector to a pose in the set* E_{goal}*:* $\mathrm{Q}_{goal} = \{\forall\ q \in \mathrm{C}_{full} \mid FK(q) \in \mathrm{E}_{goal}\}$.

For a redundant arm, even a discrete pose set E_{goal} gives rise to a continuous submanifold of solutions Q_{goal}. Given the above definition, the objective is to solve the following:

Definition 4 (Motion Planning with a GCTS)**.** *Given a start arm configuration* $q_{start} \in \mathrm{C}_{free}$ *and a set of goal poses for the end effector* $\mathrm{E}_{goal} \subset \mathrm{E}$*, a solution path is a continuous curve* $\pi : [0, 1] \rightarrow \mathrm{C}_{free}$*, so that* $\pi(0) = q_{start}$ *and* $\pi(1) \in \mathrm{Q}_{goal}$*. Given the space of all solution paths* Π*, the objective is to find the path* $\pi^* = \mathrm{argmin}_{\pi \in \Pi} C(\pi)$*, which minimizes a cost function C defined over* Π*.*

3.2 *Defining Goal Poses for Pick and Place Instances*

For picking, the goal is to move the end effector to a pose that allows for a stable grasp.

Assumption 1. *A grasping planner produces a set of stable, collision-free grasps for the arm's end effector and the target object o given sensing information.*

A database of grasps can be computed in the object's frame [35] [36] [37] [38] and evaluated in terms of their stability offline [39]. In particular, the grasp generation process follows previous work [40]. The current work considers this module to be a black-box, where the proposed approach only requires information online about the relative configuration of the end-effector and the target object that defines a *prehensile grasp*. This information forms the goal specification for the motion planning problem the proposed algorithm attempts to solve.

Pose Estimation and Modeling: Note that some grasping modules are model-based and depend on the detection of the target object model and its 6D pose [41] [42] in order to place the object model appropriately in the world. In the current work, we assume that the grasping module uses the necessary information from such perception modules to provide our method with the valid grasping poses of the end-effector. In order to check the validity of these grasps and object placements, beyond just stability, collisions must also be accounted for. The current work chooses not to focus on this and assumes the existence of such a collision checking module that is aware of the obstacle geometries, as well as the objects in the scene (either in the form of models at poses or unstructured point clouds). It should be noted that an awareness of the pose of the object once it is picked is necessary during planning for placement since the collision checking must occur with the geometry of the object attached to the end-effector at the specific grasp. These are typical assumptions that lets us focus on the algorithmic challenges.

Online, the grasps are transformed given the object pose p_o and collision checked, given the

end effector's geometry and the obstacles O. Nevertheless, states that correspond to grasps are on the boundary of C_{free} with C_{obs}, which result in very hard motion planning problems.

Note on clearance: The notion of *clearance* (distance of a configuration from the nearest obstacle) is a well studied property [25] of a motion planning problem. Motion planning literature is typically confined to the study of problems that lie in the δ-interior of the space (i.e., the subset of the configuration space that is at least $\delta > 0$ away from obstacles). This measure becomes infinitesimally small over time. The problem of planning for manipulation however require start or goal states in contact with the target object or resting surface, with a clearance of 0. This is not introduced by clutter but by the nature of grasping configurations, or stable placement poses that makes contact between the robot and obstacle geometries.

The goal set for picking instances is defined for "pre-grasp" poses: each grasp returned by the grasping planner is transformed by moving the end effector away from the grasping configuration in a line as shown in Fig. 3. This line is defined by projecting from the end-effector's frame to the target object's surface. A solution returned by the planner that brings the end effector to a "pre-grasp" pose, needs to be appended with a motion so that the end effector achieves the actual grasp.

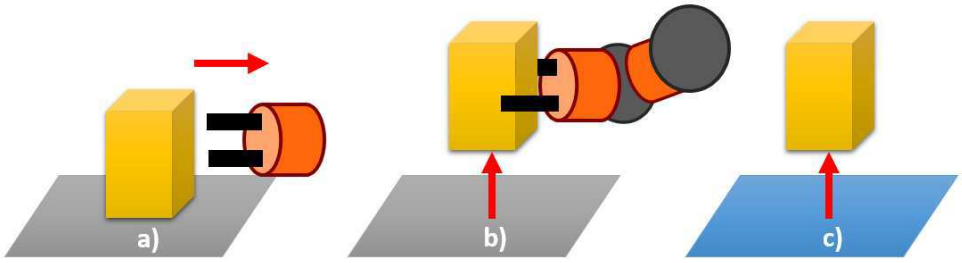

Figure 3. Visualization of offsets for a) Pre-grasp pose, b) post-grasp state, c) object pre-placement. In the cases illustrated in the paper, the pre-grasp pose offsets were along the normal from the palm of a fingered gripper, or along the contact surface for a suction based gripper, and out of the normal of a resting surface for placement offsets.

A similar setup is used for placing, where in the initial state the target object is immobilized by the end effector. In this case, the arm-object system can be considered as the robot and the end effector is the object itself.

In the current work, these *offset* poses of the end-effector are moved *away* from the target object (for grasps), and the resting surface (for placements) by $2cm$. Such an offset is handled by some grasping module aware of the end-effector and the grasp, and is not a focus of this work. The proposed motion planner generates motion plans to and from these configurations that now have non-zero *clearance*. An additional step is necessary at the beginning or end of the solution, amounting to moving the end-effector along the *offset*.

It is important to note the difference between the notion of theoretical zero-clearance and clutter. The aforementioned need to introduce offsets is not unique to our method but affects sampling-based planning methods in the context of object manipulation, inasmuch as being theoretically required in an empty tabletop scene with a single object. Clutter refers to the presence of *other* obstacles around the manipulation targets, which typically arises in real-world scenes. A high DOF manipulator deftly maneuvering into a narrow shelf containing several objects to a *pre-grasp offset* pose for a target object need not touch any of the obstacles, but will necessarily pass very close to them, and consequently have low clearance. It is these cluttered motion planning instances, with small but non-zero clearances inspired from realistic scenes, which prove problematic and therefore are the focus of this work.

Assumption 2. *For placement, a task planner produces a set of stable object placements on a support surface.*

Again, it is helpful to avoid defining problems that bring the robot close to the boundary of C_{free}. Thus, if the initial state has the object on a resting surface, the state is transformed so that the object is raised away from the surface. For a goal object placement, the pose is

transformed so that the object is again raised away from the surface. Planning solutions from "post-grasp" states to "pre-placement" goal poses are again extended so that the robot moves the object from the grasp state to the actual goal placement.

3.3 *Algorithmic Considerations*

This section delves into the algorithmic choices made in the proposed method. While motion planning for a high-dimensional manipulator is a well studied problem, and indeed a natural extension of typical high-dimensional robotic motion planning, it has been demonstrated that practical performance is severely limited in *cluttered* scenarios (Fig 2) where the manipulator has to plan to grasping poses. The current work explores and evaluates certain design choices that can lead to efficient, high-quality solutions to such problems.

3.3.1 End-Effector Guidance

It has been established that the problem at hand is closely tied to the motions of the end effector. The goals to a manipulation problem can be defined in terms of end-effector constraints(i.e., picks and placements). Previous work [43] has shown that the end-effector is an important consideration for human observers who pay more attention to the end effector rather than the rest of the arm during a manipulator's motion. The careful choice of end-effector [40] also heavily affects the success of a manipulation problem, both for picking and planning to picks. This work focuses on solving the problem for the end-effector and using the search process for the end-effector as a guiding heuristic in the higher dimensional configuration space.

3.3.2 Cost Function

The current work proceeds to define a distance function appropriate for this setting, that minimizes end effector displacement, which was shown to be useful in the context of manipulation in clutter [4]. Another benefit of this is that paths that minimize end effector motion tend to look more natural to human observers [43].

A standard approach for computing distances in $SE(3)$ is to apply a weighted Euclidean distance function, depending on the kinematics of the system.Other variations apply some weighted combination of the translational and the rotational component of the distance. Both cases require parameter tuning, and also introduces additional complications arising from symmetries in rotation. Ideally, the distance function used should be parameter-less and should appropriately capture rotational symmetries. Finally, in order to allow sampling-based motion planners to readily apply this distance function and maintain its properties, it should satisfy the properties of a metric space.

Towards these objectives, this work employs the robot displacement metric (DISP) [44], which is a model-aware distance metric for $SE(3)$, that can be approximated efficiently by the C$-$DIST algorithm [45]. Given the convex-hull H of the end-effector's model, the DISP distance D of two poses $e_i, e_j \in SE(3)$ is:

$$D(e_i, e_j) = max_{\mathbf{p}\in H} ||\mathbf{p}(e_i) - \mathbf{p}(e_j)||_2 \tag{1}$$

With slight abuse of notation, the expression $D(q, e)$ between $q \in \mathrm{C}_{full}$ and end effector pose $e \in \mathrm{E}$ will denote the distance $D(FK(q), e)$. It follows that the distance between $q_i, q_j \in \mathrm{C}_{full}$ is $D(FK(q_i), FK(q_j))$ Then, the cost of a path π is:

$$C(\pi) = \sum_{i=0}^{k} D(\pi(i), \pi(i+1)) \tag{2}$$

where k is a discretization along π. For the remainder of the paper, unless otherwise stated, *distances* will refer to Eq. 1, and *costs* will refer to Eq. 2.

3.3.3 Effective Exploitation

The end-effector and the manipulator lie in different spaces $SE(3)$ and C-space respectively. The utilization of the end-effector guidance must

- drive the arm along the end-effector heuristic in $(SE(3))$, i.e., the arm needs to be controlled while moving the end-effector in a pre-defined way
- such motions of the arm need to produce paths that have high-quality DISP cost, so that the discovered solutions demonstrate practical performance

Jacobians are a standard formulation [46] for controlling a robot in a high-dimensional C-space using gradients that exist in a lower dimensional embedding. This work accordingly uses a controller that performs successive *Jacobian* steps that move the end-effector in a desired direction. Further details of the exact way the current work uses this steering method, as well as its benefits, is described further on.

3.3.4 Exploration

Despite narrowing down on the kind of heuristic appropriate for the problem, as well as a way to utilize that heuristic for practical performance, the motion planning for manipulation problem still lies in the C-space of the arm.

There are three bottlenecks specifically related to completeness guarantees which need to be addressed:

- *Jacobians* can have singularities [46] (i.e., there are parts of the space where arm controls are undefined for moving the end-effector in specific ways). The algorithm must have means to explore alternative ways to make progress from such parts of the space, in order to achieve completeness.
- *Redundant* manipulators are high-dimensional arms that have more degrees of freedom than necessary to achieve $SE(3)$ poses for the end-effector. This leads to non-unique solutions for the arm, while moving the end-effector. Asymptotically, all such solutions have to be explored in order to guarantee completeness.
- Both the previous concerns make it unreliable to commit to a single end-effector path and attempt to track it with the arm. Therefore, the heuristic needs to be designed to provide a diverse set of end-effector targets which can make progress towards the goal.

Accordingly, the method uses *Jacobians* to steer the arm as part of a set of greedy (i.e. lowest end-effector displacement cost) maneuvers which make progress towards a variety of promising end-effector waypoints. In order to ensure coverage of the entire space for completeness, these maneuvers are used as a part of a sampling-based framework [47], which conveniently provides *asymptotic optimality* guarantees as well. It should be noted that this work[47] was co-developed with the current work and proposed as the generalized framework that addresses kinodynamic problem domains where such composition of maneuvers vastly improve practical performance.

This work focuses on the study of manipulation in clutter problem, involving designing and using heuristics within the sampling-based framework, and evaluating the efficacy of the approach in addressing the problem.

4. Jacobian Informed Search Tree

This section describes a process for exploring the end effector's space E and how the corresponding information is used to guide the exploration of the arm's space C_{full} .

4.1 *Exploring the End Effector's Pose Space*

A simple way to estimate how far the end effector is at pose $e \in \mathrm{E}$ from the set E_{goal} is to consider the minimum SE(3) distance from e to all goal poses E_{goal}. Nevertheless, in the presence of clutter, these distances are not informative of the end effector's path cost so as to reach goal poses.

To take clutter into account, this work proposes the offline construction and online search of a roadmap $R_{ee}(V_{ee}, E_{ee})$ in E as shown in Fig. 4. The roadmap vertices V_{ee} correspond to reachable end effector poses given the arm's kinematics, while the edges store local connection paths for the free-flying end effector. The goal poses E_{goal} and the start pose $e_{start} = FK(q_{start})$ are attached to R_{ee}, which is then searched online for a path from E_{goal} to e_{start}. Performing the search in this direction produces a multi-start search tree rooted at E_{goal}. During the online search, only collisions between the end effector, the obstacles O, and the object o, but not the arm, are taken into account. This allows for quick estimations of the costs to reach E_{goal}. These costs are used later as heuristics to guide a search procedure in the arm's C-space.

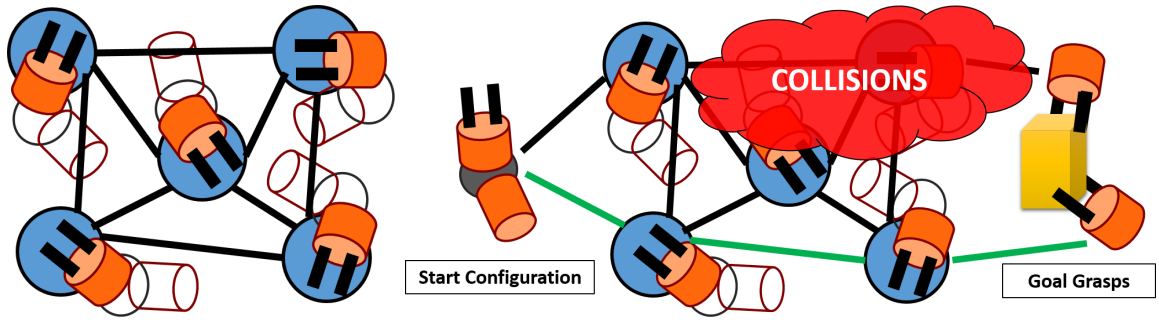

Figure 4. (left) Offline construction of R_{ee} without any prior scene knowledge. Each vertex represents a reachable end-effector pose of the manipulator. Distances between pairs of vertices are computed using DISP, with connectivity following the PRM* [25] algorithm. (right) Online search of R_{ee} which takes into account the obstacles present in the scene, finding collision-free end-effector poses which connect the starting end-effector pose to the set of goal pre-grasps. The search-tree which arises from this process is utilized as a heuristic for guiding the motions of the arm in the proposed method.

4.1.1 Offline Construction of Reachability Roadmap

The end effector reachability roadmap $R_{ee}(V_{ee}, E_{ee})$ is constructed offline without knowledge of the workspace and reflects the robot's reachability. A sampling-based process is followed similar to PRM* [25]. A random arm state $q_r \in \mathrm{C}_{full}$ is sampled and the corresponding end effector pose $FK(q_r) \in \mathrm{E}$ is stored as a vertex. As in PRM*, each vertex is connected with an edge to its k-nearest neighbors, where $k = \log|V_{ee}|$ and nearness is defined by the distance $D(q_r, e)$. Each edge stores an interpolation in SE(3) between the two vertex poses according to a discretization defined by an estimation the end-effector's maximum velocity. This estimate is computed given an upper bound limit of the arm's joint velocities.

4.1.2 Online Computation of Distances to Goal Poses

Given knowledge of the workspace, R_{ee} is searched for collision-free end effector paths from the goal poses E_{goal} to the start pose e_{start}. The poses E_{goal} and e_{start} are added as vertices on R_{ee} and are connected to their closest $\log|V_{ee}|$ neighbors.

Then, a multi-start, multi-objective A* (MSMO_A*) search procedure is performed on R_{ee}. Upon initialization, the priority queue of the search includes all goal poses E_{goal}. Then, each search node corresponds to a path from a pose in E_{goal} to a vertex $u \in V_{ee}$. The priority queue sorts nodes so as to minimize the following two ordered objectives:

- $f_1(u)$, the number of colliding poses along the path from E_{goal} to u, given the roadmap's edge discretization;
- $f_2(u) = g_2(u) + h_2(u)$, where g_2 is $C(\pi(\mathrm{E}_{goal}, e_u))$ and h_2 is $D(e_u, e_{start})$.

Once a vertex u has been expanded, it is added to the "closed list" L_{closed}. If there is a collision-free end effector path from E_{goal} to e_{start}, the above search will report the shortest path. Then,

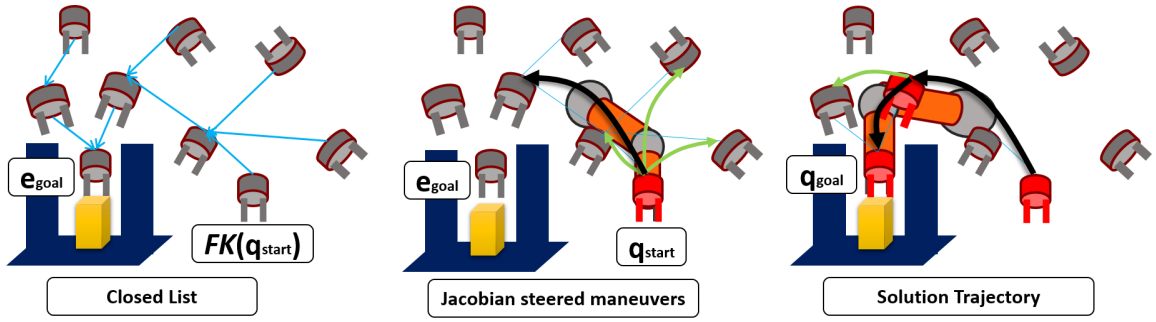

Figure 5. An illustration of the JIST framework for solving manipulation queries: (left) the closed list L_{closed} is constructed using a multi-start, multi-objective A* on R_{ee} starting from E_{goal} to $FK(q_{start})$; (middle) JIST expands arm paths from the start configuration q_{start} guided by the costs stored in L_{closed}; (right) the best path found π^* is kept track of as the solution trajectory, with subsequent solutions from JIST improving upon it.

the g_2 cost of vertices u in L_{closed} corresponds to an estimate of the distance along the shortest collision free path from u to E_{goal}, i.e., g_2 is a discrete approximation of a navigation function in E given the presence of obstacles. The proposed approach uses these g_2 values to heuristically guide the exploration in C_{full}. In dense clutter, if the roadmap's resolution does not permit collision-free paths, the algorithm returns the path with the minimum number of collisions given its discretization. In this case, the g_2 values of vertices in L_{closed} will guide the arm exploration along the shortest, least colliding solutions for the end effector.

4.2 *Generating Arm Paths*

This section describes how to generate paths for the arm given the vertices $u \in V_{ee}$ stored in the "closed list" L_{closed} and the cost estimates g_2 from u to E_{goal}. First, an overview of the algorithm is presented. Then, each module is described in depth, presented along with pseudo-code. Finally, the steering process used in the algorithm is explained. An illustration of the steps of the algorithm can be seen in Figure 5.

Algorithm 1: JIST $(q_{start}, \mathrm{E}_{goal}, R_{ee}, N, \kappa)$

$\mathrm{L}_{closed} \leftarrow \mathrm{MSMO\_A}^*(R_{ee}, q_{start}, \mathrm{E}_{goal})$; // Builds the closed-list heuristic
$\pi^* \leftarrow \emptyset$; // Initialize optimal found solution
$\mathrm{T} \leftarrow \{q_{start}\}$; // Initialize tree
$q_{new} \leftarrow q_{start}$; // Initialize newly added node
$\mathrm{A}_{cand}(q_{new}) \leftarrow NULL$; // Initialize candidate actions
for N *iterations* **do**
  $q_{sel} \leftarrow \mathrm{NodeSelection}(\mathrm{q_{new}}, \mathrm{L_{closed}})$; // Selects the arm node q_{sel} to extend
  $a_{best} \leftarrow \mathrm{EdgeGeneration}(\mathrm{A_{cand}(q_{sel})})$; // Selects the action to execute from q_{sel}
  $\mathrm{TreeExtension}(q_{new}, q_{sel}, a_{best})$; // Steers from q_{sel} by executing a_{best}
return π^*;

4.2.1 Overview

The high-level algorithm is outlined in Alg. 1, which is inspired by a search strategy [26] capable of utilizing heuristic guidance. It receives as input the starting arm configuration q_{start}, the goal poses E_{goal}, the end effector reachability roadmap R_{ee}, the number of iterations N, and the maximum number κ of greedy edge expansions per iteration.

The method first executes multi-start, multi-objective A^* search over R_{ee} as described in Section 4.1.2, which constructs the closed list L_{closed}. This list provides cost estimates to the goal poses E_{goal} for each vertex in R_{ee} visited during the search, which will be used as the

heuristic during the tree search. The algorithm is initialized (lines 3-5) with the initial arm configuration and an empty action set. The method then performs N iterations of the following three high-level procedures: NodeSelection, EdgeGeneration, and TreeExtension.

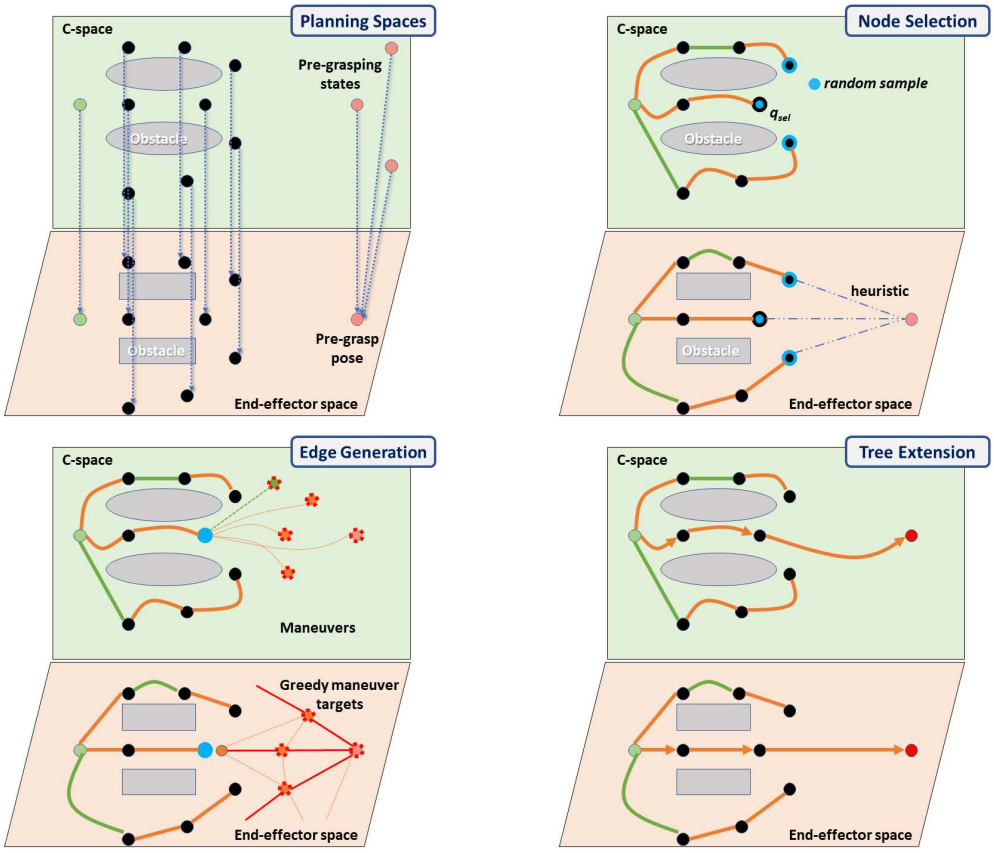

Figure 6. Some of the important aspects of the JIST algorithm are illustrated in this figure. *(Top Left:)* The search uses information from both the configuration space of the arm, and the end-effector space. Every configuration in the C-space maps to a point in the end-effector's space. Multiple states, like the pre-grasping states, can map to the same end-effector point for a *redundant* manipulator. *(Top Right:)* During the online search, JIST builds a search tree that traverses both spaces. Jacobian extensions (orange) will tend to optimize connections in the end-effector space (DISP cost), and fallback exploratory steering (green) optimizes the connections in the configuration space of the arm. In the vicinity of a random sample, the selection process prefers nearby tree-nodes with a good *f-value* i.e., sum of cost to reach the goal and the heuristic [47]. *(Bottom Left:)* A search tree (red lines) obtained over the end-effector roadmap provides guidance towards the targets from q_{sel}. The search tree is traversed towards the goal, and among these end-effector poses and their neighbors on the roadmap a set of promising end-effector targets are selected as candidate greedy maneuvers (orange dotted). A fraction of the maneuvers will also be exploratory fallback maneuvers in the C-space (green dotted). *(Bottom Right:)* The algorithm makes quick progress towards the goal in cluttered scenes and finds high-quality solutions in terms of DISP.

4.2.2 Node Selection

This process is responsible for selecting the portion of the tree to extend. Pseudocode is given for it in Alg. 2, and a visualization of the process is shown in Fig. 6 (Top Right). The heuristic

Algorithm 2: NodeSelection(q_{new}, L_{closed})

1 $h(q_{new}) \leftarrow$ HeuristicLookup(FK(q_{new}), L_{closed}) ; // heuristic computed from L_{closed}
2 **if** $q_{new} \neq \emptyset$ **and** $h(q_{new}) < h(parent(q_{new}))$ **then**
3 $q_{sel} \leftarrow q_{new}$; // the selected node is chosen as q_{new}
4 **else**
5 $q_{sel} \leftarrow$ SearchSelection(); // heuristic-guided sampling in the arm space

of the new node, if it exists, is first computed (line 1). Here the heuristic value $h(q)$ of an end

effector configuration q corresponds to an estimate of the cost to reach E_{goal}. In theory, $h(q)$ could be computed by solving a MSMO_A* on the end effector reachability roadmap R_{ee}. In practice this would increase the computation time of the algorithm significantly, due to the sheer volume of searches that would need to be performed for each new q added to the search tree. Instead, we propose an approximation using L_{closed} which already has costs stored as part of the closed list generated during the initial MSMO_A*. This is a one-time computation cost incurred, and allows computing $h(q)$ to be simplified. By finding the nearest end effector configuration to $FK(q)$ in L_{closed}, its corresponding cost can be used as the heuristic value of q.

If a node was added during the previous iteration, and its heuristic value is better than its parent on the tree, then the newly added node is selected for expansion (line 3). If there was no progress made in the previous iteration towards reaching E_{goal}, the SearchSelection subroutine (line 5) instead uses a probabilistic selection process, where the probability of selecting a node depends on the node's corresponding sum of cost from the root and heuristic cost to reach the goal. All nodes are guaranteed to have a non-zero probability of being selected; however, nodes with better costs and heuristic sum have a greater probability.

4.2.3 Edge Generation

This process is responsible for generating the actions to extend edges from the selected node. Pseudocode is given for it in Alg. 3, and a visualization of the process is shown in Fig. 6 (Bottom Left).

Algorithm 3: EdgeGeneration($\mathrm{A}_{\mathrm{cand}}(\mathrm{q}_{\mathrm{sel}})$)

```
   /* If the action set A_cand(q_sel) has not been generated yet                */
1  if A_cand(q_sel) = NULL then
2      A_cand(q_sel) ← GreedyEdges(q_sel, L_closed, κ);
   /* Else if there are no more greedy actions in A_cand(q_sel)                 */
3  else if A_cand(q_sel) = ∅ then
4      A_cand(q_sel) ← FallbackEdges(q_sel);
5  a_best ← argmin_{a∈A_cand} h(a, L_closed) ;        // select the lowest heuristic action
6  A_cand(q_sel) ← A_cand(q_sel) \ a_best ;    // a_best is removed from the candidate action
     set
```

Once a configuration q_{sel} is selected for expansion, the set A_{cand} is the set of actions that can be used to extend edges of the tree out of q_{sel}, which can correspond either to target poses for the end effector or joint velocities for the arm. The first time that node q_{sel} is selected for expansion (line 1), the subroutine GreedyEdges is used to generate target poses for extending the tree out of q_{sel} (line 2). If all such greedy extensions have been already considered out of q_{sel} (meaning that there are no more actions left), then the algorithm generates fallback actions in A_{cand} (line 4).

Greedy Edge Generation: These greedy edges are guided by the information stored in the L_{closed}, which contains end effector poses explored during the online search of R_{ee}. First, the nearest end effector pose $e_{near} \in \mathrm{L}_{closed}$ is found relative to $FK(q_{sel})$. The method obtains poses to steer towards by examining the adjacent vertices of e_{near} on R_{ee}, which also belong to L_{closed}, and whose heuristic is lower than q_{sel}. The number of target poses added is controlled by parameter κ. Experimental indications show that a value $\kappa = 2{*}\log|V_{ee}|$ works well in practice. The method then updates e_{near} to its predecessor from the closed set, and attempts to add target poses until either κ poses have been discovered, or there are no other predecessors available. In the latter case, the remaining target poses are sampled directly from the set E_{goal}. Overall, this is a greedy procedure, which traces along the search tree produced by the A*, while examining nearby poses, which have also been expanded during the A* search.

Fallback Edge Generation: Every time a node is selected and all greedy edges out of it have been expended, the method reverts to considering two fallback strategies for generating edges. The first strategy corresponds to a random control in terms of joint velocities executed for a random duration. The second strategy randomly selects a goal state $q_{goal} \in \mathrm{Q}_{goal}$ already discovered by the algorithm as a target to steer towards.

`JIST` iterates over the generated actions A_{cand}, which are ordered to promote the lowest h (line 5) in terms of the resulting pose (i.e. the pose obtained by executing the action). The best action a_{best} is then considered for TreeExtension, and is thereby removed from the candidate action set (line 6).

4.2.4 Tree Extension

This process is responsible for performing steering and ensuring that the tree maintains its properties in terms of probabilistic completeness and asymptotic optimality. Pseudocode is given for it in Alg. 4, and a visualization of the process is shown in Fig. 6 (Bottom Right).

Algorithm 4: TreeExtension($\mathrm{q_{new}}$)

1 $q_{new} \leftarrow \mathrm{Steer}(q_{sel}, a_{best})$; `// generates a path to` q_{new}
2 **if** $\mathrm{BaB}(q_{new}, \pi^*)$ **or** $\mathrm{CC}(q_{sel} \rightarrow q_{new})$ **then**
3 $\quad q_{new} \leftarrow \emptyset$; `// the new node violates the criteria and is discarded`
4 **if** $q_{new} \neq \emptyset$ **then**
5 $\quad \pi^* \leftarrow \mathrm{AddEdge}(\mathrm{T}, q_{sel} \rightarrow q_{new})$; `// checks edge for a new optimal solution`
6 $\quad \mathrm{A}_{cand}(q_{new}) \leftarrow NULL$; `// initialize new node's action set`

The tree extension begins by using a specific steering method depending on whether a_{best} is a greedy or fallback action (described in more detail in the following section). This produces a path whose terminal state is q_{new} (line 1). The configuration q_{new} must meet two requirements to be added to the tree as a node (line 2): (a) A branch and bound process (BaB), which rejects edges from the trees whose total cost is greater than the best solution cost found, guarantees that q_{new} has smaller path cost relative to the length of the best solution found π^*; (b) A collision checker CC verifies whether the path from q_{sel} to q_{new} is in collision. If the node q_{new} passes these checks (line 5), then it is added to the tree T. If the addition of q_{new} permits a path to the goal whose cost is lower than the current best path, then the new path is recorded as π^* (line 5) and the newly added node is initialized with an empty action set (line 6).

4.2.5 Steering

For the fallback edges, the steering subroutine executes the random controls for a random duration, thereby simplifying the proof of the algorithmic properties (described in more detail in Section 5). The greedy edges consist of target end effector poses. This work uses a steering method based on the pseudo-inverse of the manipulator Jacobian matrix to achieve the target poses. Given an arm configuration $q = (q_1, ..., q_d)^T$ and the corresponding end effector pose $e = FK(q)$, the Jacobian matrix is $J(q) = \frac{\partial e}{\partial q}$.

For a target end effector pose e_{target}, let $\Delta e = e_{target} - e$ denote the desired change in position of the end effector. The objective of the Jacobian-based steering process J^+steering is to compute the joint controls which solve $\Delta e = J\Delta q$. The method uses the pseudo-inverse of the Jacobian, J^+, so as to minimize $||J\Delta q - \Delta e||^2$, where $\Delta q = \mathrm{J}^+\Delta e$. To account for singularities, damping [48] and clamping are also employed. Thus, at time t, for the configuration q_t and the corresponding end effector pose e_t, the control update rule for J^+steering is:

$$\Delta q_t = \mathrm{J}^+(q_t) \cdot (e_{target} - e_t) \tag{3}$$

Since the objective is to minimize DISP, the gradient followed by the above rule reduces the distance of the end effector to the target pose. Any configuration $q_{goal} \in \mathrm{Q}_{goal}$ discovered through J^+steering is kept track of. A benefit of using J^+steering is that it satisfies the necessary properties of the projection operator [31] needed to ensure coverage of Q_{goal}. J^+steering is used in the connection of the goal poses described in Section 3.

5. Outline of Properties

This section provides a sketch of the asymptotic arguments that can be made about the DISP solution quality of the algorithm. Note that all the arguments rely on the guarantees that a random tree covers C_{free} asymptotically optimally for the Euclidean distance metric, and pseudo-inverse Jacobians can cover the manifold that contains all the target configurations. By showing that the DISP can be conservatively bounded, it follows that our choice of fallback maneuvers that create a random tree with random controls will converge to solutions that asymptotically have a DISP cost of zero variation w.r.t the optimal solution.

Proposition 1. JIST *is guaranteed to eventually reach the optimal grasping state.*

Proof Sketch: Prior work [31] has argued that a pseudo-inverse projection operator can be used to guarantee a probabilistically complete mapping of C_{full} to the end effector pose space E. Using J^+*steering* ensures asymptotically complete coverage of Q_{goal} over the end-effector roadmap, as the size of the roadmap also approaches inf. □

Proposition 2. *The cost of a path that observes the optimal solution path converges to the optimal cost.*

Proof Sketch: A sequence of finite hyperballs of radius δ can be constructed to tile the optimal solution path that has the lowest cost. The existence of a $\delta > 0$ is guaranteed by the assumptions of δ-clearance of the optimal solution. A path that observes the optimal path is contained in the tiling of these hyperballs of some non-zero volume and bears a Euclidean cost. It is expected that for any choice of δ, the path contained inside this tiling π_δ has a finite bounded variation [25] w.r.t. the optimal π^*. In essence this is the sum of the Euclidean distances between pairs of progressing points along the two paths, i.e., $\|\pi_1 - \pi_2\|_{BV} = \int_0^1 \|\pi_1[\tau] - \pi_2[\tau]\|\delta\tau + (\|\pi_1\| - \|\pi_2\|)$ where the first term is the pairwise difference, the second term is called the *total variation*, and $\|\cdot\|$ refers to the standard Euclidean arc length [25].

This measure approaches 0 as δ approaches 0, i.e., $\|\pi_\delta - \pi^*\|_{BV} \to 0,\ as\ \delta \to 0$. It needs to be shown though whether this also implies that the cost is bounded in terms of our distance metric of interest DISP.

Assume the problem does not involve degenerate motions in the optimal path, i.e., ones connecting identical start and goal configurations. DISP forms a metric space [45] when computed over the end effector geometry. For a robotic arm with bounded reachability and a bounded C_{full}, there must exist a finite constant K_c that bounds the ratio between the Euclidean cost and DISP cost for the straight line between any pair of points in C_{full}, i.e., K_c s.t. $\frac{D(q_i,q_j)}{\|q_i,q_j\|} \leq K_c\ \forall\ q_i, q_j \in \mathrm{C}_{full},\ q_i \neq q_j$.

Assuming Lipschitz continuity of curves in C_{full}, the DISP bounded variation of the observing path is bounded w.r.t. the Euclidean bounded variation as $D(\pi_\delta - \pi^*)_{BV} \leq K_c\|\pi_\delta - \pi^*\|_{BV}$. It follows that $D(\pi_\delta - \pi^*)_{BV} \to 0,\ as\ \delta \to 0$, and converges with a reducing δ tiling. Given the bounded continuity assumptions and for finite costs of the observing path and non-zero costs of the optimal path, this convergence holds for DISP. Without additional rigor, it can be argued that this heavily implies $D(\pi_\delta) \to D(\pi^*)$ as $\delta \to 0$. The proof of this stronger notion of asymptotic optimality is left as future work.

Note that the previous work showed the convergence of the bounded variation for a reducing radius δ affected by the number of samples n. We are only interested in part of the result that

implies that as δ reduces, the bounded variation of the paths inside the δ tiling reduces. The proof techniques necessary for connectivity between the non-zero volume hyperballs will arise from the naive random tree being able to trace any such sequence [49]. □

Proposition 3. JIST *is asymptotically optimal.*

Proof Sketch: A Markov chain is constructed and it "observes" an optimal path and has an absorption state corresponding to the goal region. Then, if the probability of transitioning between Markov states is non-zero, the method surely reaches the absorbing state with infinite tries [50] (Theorem 11.3). Given a covering ball sequence for an optimal path π^* with some δ clearance, `JIST` satisfies the following conditions:

(1) The subroutine SearchSelection ensures that the probability of selecting a node is always non-zero.
(2) There is a non-zero probability of moving from one covering ball to the next once a node is selected in a random tree with random controls. Prior work (Theorem 17 in [49]) proves this given assumptions about smooth system dynamics. This is guaranteed to have positive probability when the propagation is not biased during expansion. This is achieved by Alg. 1 through the random controls with random duration as fallback edges. This ensures that the Euclidean cost of the path converges.
(3) The optimal goal state is guaranteed to be discovered and optimally connected to the search tree by Proposition 1.
(4) The DISP cost variance of the discovered path w.r.t. to the optimal path will asymptotically reduce to 0, by Proposition 2.

□

Bringing it all together, the algorithm is expected to converge to the optimal DISP solution asymptotically. Practically, a random tree has a very poor rate of convergence, which makes the specification of asymptotic optimality of the proposed method a theoretical curiosity. The emphasis of the current work is on the use of greedy maneuvers to converge to high quality solutions very quickly. The inspection of the more detailed properties that take into account the greedy maneuvers of J$^+$steering that make progress towards goals on a lower dimensional manifold is left as future work.

6. Experiments

This section first evaluates various steering methods for reaching target poses in the presence of clutter. Then, `JIST` is experimentally compared against other planning algorithms in a variety of cluttered benchmarks.

6.1 *Steering Method Evaluation*

Steering is typically used to connect nearby robot configurations inside a motion planning method. While our proposed approach reverts to the use of C geodesic connections between manipulator configurations over the fallback maneuvers, the greedy maneuvers should employ a steering method that is most likely to yield a collision free solution quickly. The existence of clutter in the target applications indicates that typical sweeping motions in the workspace generated by C interpolation increases the likelihood of colliding with the obstacles. In order to validate the hypothesis that steering methods which reason about the end-effector reduces the likelihood of collisions the following study has been conducted.

Accordingly this study measures the effectiveness of different ways to steer a robotic arm from configuration q to a target end-effector pose e in the local vicinity of $FK(q)$.

Setup: We study two different environments, a cluttered table and a shelf, as shown in Figure

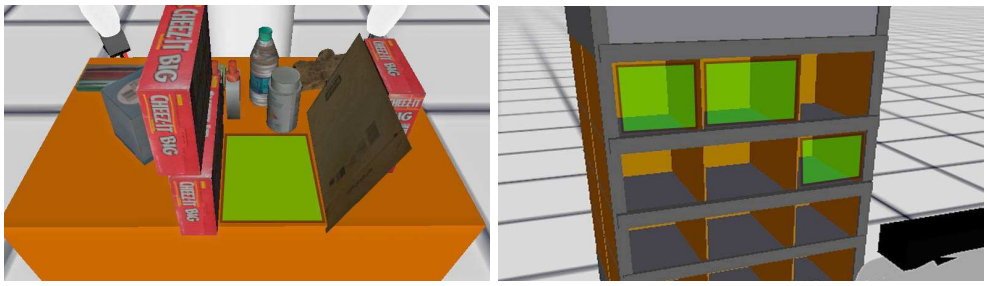

Figure 7. The environments considered in the experiments with the target object removed: (left) a cluttered table and (right) a shelf with tight openings. The highlighted sections represent possible target object placements.

7. Within each environment, we first generate a set of target poses E_{eval} by sampling in SE(3) within a bounded region. This bounded region is selected so as to bias the samples in areas of the workspace that are cluttered (e.g. inside a bin in the shelf or directly above the table). If the sampled pose e produces an arm configuration $q = IK(e)$ which is collision-free, then the pose e is added to the evaluation set, along with any collision-free IK solutions of q. This process is repeated until 50,000 end effector poses are generated. Note that the selection of end effector here can have a direct impact on the collision-free check of the arm. To remove this dependence on end effector geometry, the arm is evaluated without any end effector attached. Consequently all IK calls are made relative to the last joint of the arm.

For each $q \in \mathrm{C}_{eval}$, the process involves steering to each $e \in \mathrm{E}_{eval}$ using the following methods:

- ***local***: computes a straight-line interpolation in C_{full} from q to the first random solution of $IK(e)$;
- ***seeded***: seeds the IK-solver for e with the arm configuration q - otherwise the method is similar to *local*;
- IK***-track***: computes a straight-line interpolation in E from $FK(q)$ to e, and calls IK on each pose seeded with the previous IK solution;
- $\boldsymbol{J^{+}}$***steering***: as described in the end of Section 4.

The TRAC-IK library [51] was used as the IK solver and for the computation of the manipulator's Jacobian matrix.

Metrics: The following metrics are used to evaluate the performance of the steering methods in the trials - each metric reported is an average over all trials. *Reachability Rate* represents whether the method successfully reached the target end effector pose. For each successful path, *Computation Time* (in seconds), cumulative C_{full} *Length*, cumulative DISP *Path Length*, and *Collision Free Rate* (the fraction of the path which is collision free) is reported.

Remarks: As shown in **Table 1**, in terms of computation time and reachability rate, the *local* and *seeded* steering methods are very efficient. These methods, however, tend to produce long paths, especially in terms of DISP. What happens is that the local and seeded methods frequently cause arm reconfigurations during the steering. This results in many configurations along the path to collide with obstacles, rendering these methods less useful. Note that since no planning is involved in any of the methods, nearly all paths collided. Nevertheless, the frequency of collisions along such paths is informative. The IK*-track* method, produces the lowest DISP length trajectories but has a very low reachability rate due to a higher probability of not finding IK solutions, along with being the most computationally expensive. Overall, the J^{+}steering method provided the best trade-off. It has low computational cost, results in low DISP length paths, which are collision-free with very high probability, and has a relatively high reachability rate. When used in the context of greedy maneuvers in the proposed algorithm `JIST` the use of J^{+}steering will increase the likelihood of the algorithm succeeding during the greedy phase of the search.

Table 1. Statistics for computing a trajectory from an initial configuration q to a target pose e averaged over 50,000 trials

Steering Trials in Table Environment

	local	*seeded*	*IK-track*	J^+steering
Reachability Rate	1.0	1.0	0.332	0.785
Computation Time	0.0016	0.0013	0.008	0.0045
C_{full} Path Length	5.46	3.745	2.97	3.35
DISP Length	1.95	1.168	0.401	0.421
Collision Free Rate	0.785	0.860	0.945	0.959

Steering Trials in Shelf Environment

	local	*seeded*	*IK-track*	J^+steering
Reachability Rate	1.0	1.0	0.345	0.785
Computation Time	0.00138	0.0008	0.006	0.0042
C_{full} Path Length	5.65	3.645	2.81	3.15
DISP Length	2.07	1.147	0.335	0.338
Collision Free Rate	0.663	0.792	0.995	0.993

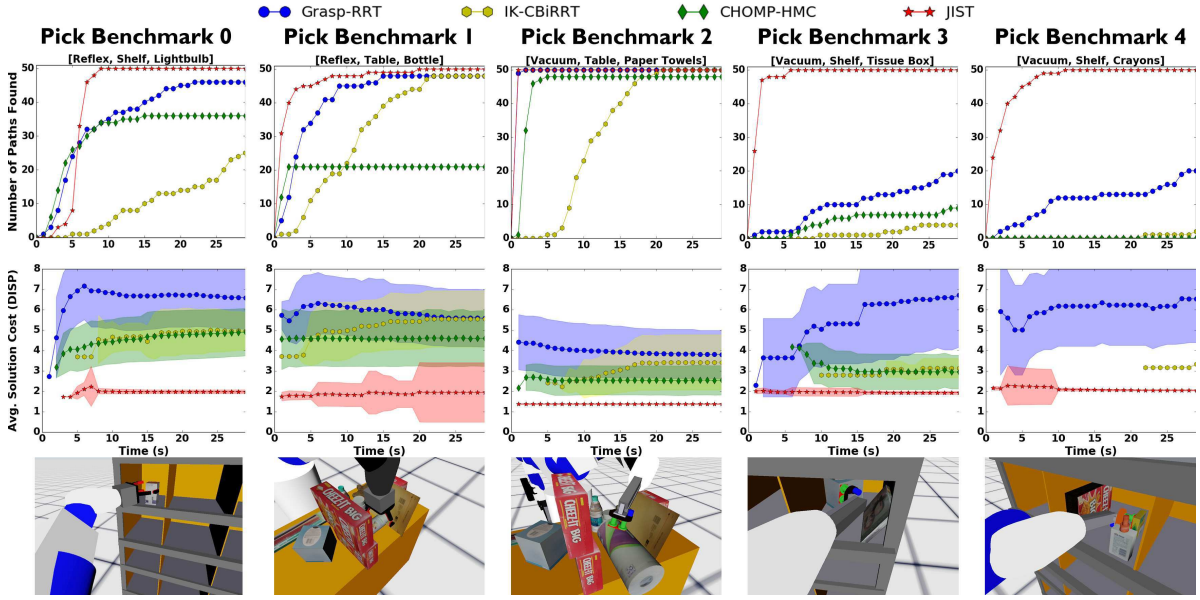

Figure 8. Number of successes over time (*top*), average solution costs over time (*middle*), and representative solution pre-grasps (*bottom*) for 50 trials on the Motoman+ReFlex (0,1), Motoman+Vacuum (2,3,4). For the same underlying manipulator these benchmarks evaluate the performance of the algorithms to plan to pre-grasp poses, with a Reflex hand and a Vacuum gripper. A target object is placed in the shelf and tabletop environment, surrounded by different degrees of clutter. The number of successes over time indicate both the final completeness of the algorithms, as well as how quickly they discover the initial solution. The average solution cost over time represents the rate at which the algorithms converge to the high-quality solutions. The proposed approach `JIST` outperforms the competing methods in terms of both metrics.

6.2 *Manipulation Benchmarks*

Algorithms: `JIST` is compared against other planners for solving manipulation challenges. Recent work [34] has utilized `CBiRRT` [1] for solving grasping problems. Accordingly, this work compares against an IK-based variant of `CBiRRT`, which uses IK on the E_{goal} to construct roots of the goal tree. `Grasp-RRT` corresponds to an `RRT` variant, which utilizes J^+steering to discover grasp states during the goal biasing phase [3]. `CHOMP-HMC` corresponds to an OpenRave [52] probabilistically complete implementation of `CHOMP` [2].

Setup: A common planning software was used for the sampling-based planners [53]. All methods were evaluated on a single Intel Xeon E5-4650 processor with 8 GB of RAM. Experiments were conducted on a variety of robotic manipulators: Kuka LBR iiwaa, Rethink Baxter, and

Motoman SDA10F. Three types of end effectors were evaluated: the ReFlex hand, a parallel-jaw, and a vacuum suction-cup. Each benchmark involved computing a trajectory for one of the arms to a set of goal end effector poses. These goals are computed following prior work [40], where a maximum of 100 attempts to generate viable grasps per benchmark is allowed. Then, as described in Section 3.2, an offset is introduced to the grasps. For pick benchmarks, the poses correspond to pre-grasps 2 cm away from the actual grasps. For place benchmarks, the goal poses corresponded to pre-placements for the object 5 cm above the resting surface.

Metrics: The number of planning successes and solution quality were measured over time, and an average over 50 runs was reported at each time interval. A success was counted if the planner computed a collision-free trajectory from the start state to the goal end effector pose within the time limit of 30 seconds. Solution quality was measured using cumulative DISP over the solution path.

Initialization: Both `JIST` and `CHOMP` required some additional initialization prior to attempting to solve any of the benchmarks. `JIST` required R_{ee} to be first computed offline (without knowledge of the scene). For the experiments, the size of R_{ee} was $25,000$ vertices, and the maximum number of maneuvers κ was set to 20. For `CHOMP` each environment had a signed-distance field constructed for it, and also had its parameters tuned per benchmark. Specifically, the parameters (n_iter, hmc_resample_lambda, lambda, obs_factor) for the Table benchmarks were (20,0.02,10,25) respectively, and for the Shelf benchmarks (100,0.02,1000,200) respectively. Both `CHOMP` and `IK-CBiRRT` were provided with reachable, collision-free arm configurations computed using IK at the goal poses.

6.2.1 Pick Benchmarks

The results for success rate and solution cost over time are shown in Figure 8 for the Motoman robot. Although the alternative methods do not inherently optimize the DISP cost function, these comparisons still illustrate several important conclusions. Across all experiments, `JIST` converged to a 100% success rate within the time limit of 30 seconds, with an average initial solution time breakdown shown in Figure 9 (left).

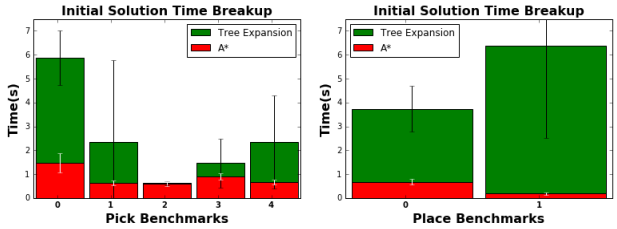

Figure 9. Breakdown in time for the initial collision-free solution using `JIST` for pick (left) and place (right) benchmarks. The A* time represents how much time it took to compute the closed list L_{closed} using MSMO_A*.

Remarks: Although the parameters of `CHOMP` were tuned for each benchmark (as described in Initialization), it was not able to reach 100% success rate within the time limit. It is possible that with further tuning, these rates could be improved - however, the proposed method did not require any parameter tuning to produce the presented results. Furthermore, both the initial solution costs of `JIST` and the variance of solutions costs were the lowest. Together this shows that the proposed heuristic was effective both in computing feasible motion plans quickly and in guiding the search to produce high quality initial solutions. An interesting side-note here regarding `CHOMP` is that when initialized with solution configurations found by `JIST` the performance of `CHOMP` improved significantly. This indicates that solutions out of `JIST` can serve as better initializations for trajectory optimization methods.

The average clearance (i.e. shortest distance between the robot and the obstacles) of solutions found by all methods in the table environment was 5.7 cm, whereas in the shelf environment it was 2.2 cm. Accordingly, the table benchmarks were the easiest for all methods to solve. This is also seen in the initial solution times of `JIST` where on average the MSMO_A* spent more time

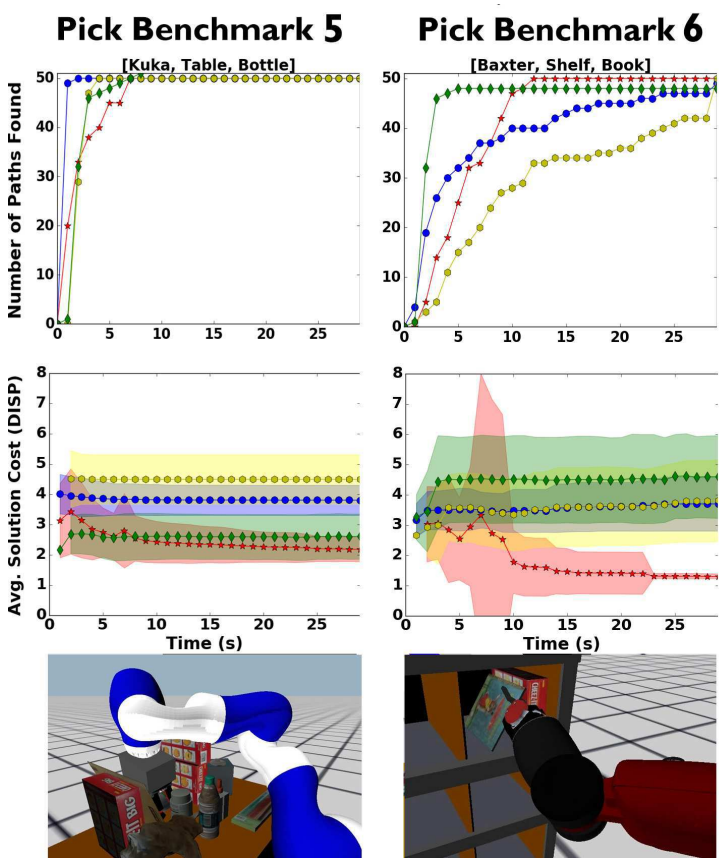

Figure 10. Success Rates, Solution Costs, and Example Picks for the Kuka+ReFlex (5), Baxter+Parallel (6) benchmarks to showcase applicability to different systems. The results indicate that JIST maintains its performance despite differences in kinematics between the Kuka, Baxter, and Motoman robots.

in the table benchmarks (1,2) than in the shelf benchmarks (0,3,4). This is primarily due to the fact that overhand grasps over the table were viable solutions. However, in the shelf benchmarks, JIST maintains its high performance despite the clutter severely reducing the overall clearance and viable grasps.

6.2.2 Application to Different Systems

Two additional pick benchmarks were evaluated on a Kuka LBR iiwaa, and a Rethink Baxter robotic arm. The results, as shown in Fig 10, indicate that the performance trends observed in the Motoman trials are consistent across different robotic platforms. On average JIST reported the first solution at 3.76s and 3.06s for benchmarks 5 and 6 respectively.

6.2.3 Place Benchmarks

The results for the place benchmarks are shown in Figure 11, with average solution times shown in Figure 9. On average, JIST spent longer in the tree expansion portion of the method, relative to the pick benchmarks due to the attached target object causing more collisions. Despite this initial slowdown, JIST still managed to converge to a 100% success rate before all other methods, while also still providing the lowest cost solutions.

6.2.4 Real World Demonstration

Fig 12 showcases an execution of a trajectory computed by the proposed method, demonstrating the sort of motion planning problem in clutter that JIST solves. We replicate the setup of Pick Benchmark 4, placing the objects in the top middle bin of a Kiva Pod shelving unit. The target object is a Crayola box inside the shelf which is partially occluded behind the Elmer Glue

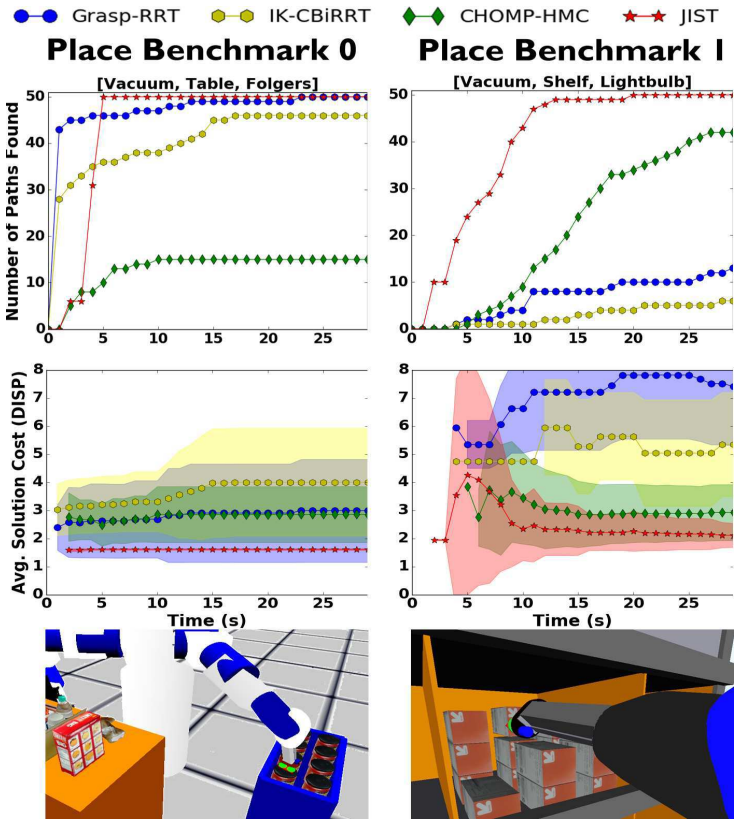

Figure 11. Success Rates, Solution Costs, and Example Placements for the Vacuum Place Benchmarks in the Table (0) and the Shelf (1) environments.

Figure 12. The solution path produced by JIST is executed on the system in an open-loop fashion, and corresponding frames of this execution along with the timestamps are shown here. The Motoman begins executing a motion plan (top left) computed for reaching an object located behind some clutter, ending with a grasp (bottom right).

Bottles in front of it and a Cheezit box on the side.[1]

An initial solution was returned after 2.41 seconds of computation time, with an initial DISP cost of 3.05. The planner was then run for the remainder of the 30 seconds, and the best solution found (with DISP cost 2.02) was executed in an open loop fashion on a Yaskawa Motoman SDA10f robotic manipulator. The narrowness of the bin through which the arm must reach through, along with the proximity of clutter, should be noted as it creates a very narrow passage with low clearance. Despite this challenge, JIST manages to find a feasible, high quality solution - the end-effector goes straight to the target object, while the arm minimizes reconfigurations.

[1]Exact specifications of each benchmark including object poses, arm stats, and meshes, are available upon request.

7. Discussion

This paper presented the Jacobian Informed Search Tree (JIST) algorithm, which is an asymptotically optimal, informed sampling-based planner for controlling an arm in densely cluttered scenes so as to achieve desired end effector configurations. The method optimizes a cost function representing the displacement of the end-effector, and uses a heuristic computed by effectively searching the end effector's task space. JIST employs Jacobian-based steering to bias the expansion of the tree towards end effector poses that appear promising given the heuristic guidance. The method was shown to have high success rate , even in densely cluttered scenes, with fast initial solution times and high quality solutions across a variety of robot manipulators. An increased focus on integrated task and motion planning challenges, means that JIST can prove to be an efficient underlying motion planning primitive for complex manipulation tasks that can involve many objects, and arms [54, 55].

JIST currently optimizes over the a single cost function minimizing end-effector motions. Future work can look into multi-objective optimization metrics that can introduce additional constraints that can affect the quality, and validity of the path like smoothness, task-specific constraints etc. JIST also provides a valuable tool for use in bootstrapping machine learning methods [56], which typically require not only large amounts of data, but also good coverage of the task space [57]. Since JIST is capable of finding high-quality solutions in even the most constraining of environments, learning approaches can better increase their success rates by utilizing solutions computed by our approach. Furthermore, the proposed framework can take advantage of learning approaches, by augmenting the maneuver generation and heuristics, which could greatly reduce overall computation time. This improved version of the method could again then be used to train the learning process, and this cycle of alternating could lead to significant performance increases, indicating a life-long learning hierarchy.

References

[1] Berenson D, Srinivasa SS, Ferguson D, Kuffner JJ. Manipulation planning on constraint manifolds. In: Icra. 2009.

[2] Zucker M, Ratliff N, Dragan A, Pivtoraiko M, Klingensmith M, Dellin C, Bagnell A, Srinivasa S. CHOMP: Covariant Hamiltonian optimization for motion planning. IJRR. 2013;32(9).

[3] Vahrenkamp N, Asfour T, Dillmann R. Simultaneous grasp and planning: Humanoid robot ARMAR-III. RAM. 2012;19(2).

[4] Azizi V, Kimmel A, Bekris K, Kapadia M. Geometric reachability analysis for grasp planning in cluttered scenes for varying end-effectors. CASE. 2017;.

[5] Ichter B, Harrison J, Pavone M. Learning sampling distributions for robot motion planning. In: 2018 ieee international conference on robotics and automation (icra). IEEE. 2018. p. 7087–7094.

[6] Chiang L, Faust A, Sugaya S, Tapia L. Fast swept volume estimation with deep learning. In: Wafr 2018, merida, mexico. 2018.

[7] Everett M, Chen YF, How JP. Motion planning among dynamic, decision-making agents with deep reinforcement learning. arXiv preprint arXiv:180501956. 2018;.

[8] Berenson D, Srinivasa S, Kuffner J. Task space regions: A framework for pose-constrained manipulation planning. The International Journal of Robotics Research. 2011;30(12).

[9] Vahrenkamp N, Do M, Asfour T, Dillmann R. Integrated grasp and motion planning. In: Robotics and automation (icra), 2010 ieee international conference on. 2010.

[10] Kimmel A, Shome R, Littlefield Z, Bekris K. Fast, anytime motion planning for prehensile manipulation in clutter. In: 18th IEEE-RAS international conference on humanoid robotics, humanoids 2018, beijing, china 2018. IEEE. 2018.

[11] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. IJRR. 1986;5(1).

[12] Song P, Kumar V. A potential field based approach to multi-robot manipulation. In: Icra. Vol. 2. 2002.

[13] Warren CW. Global path planning using artificial potential fields. In: Icra. 1989.

[14] Rimon E, Koditschek DE. Exact robot navigation using cost functions: the case of distinct spherical boundaries in E/sup n. In: Icra. 1988.

[15] Koren Y, Borenstein J. Potential field methods and their inherent limitations for mobile robot navigation. In: Icra. 1991.

[16] Schulman J, Ho J, Lee AX, Awwal I, Bradlow H, Abbeel P. Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization. In: Rss. 2013.

[17] Dong J, Mukadam M, Dellaert F, Boots B. Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs. In: Rss. 2016.

[18] Cohen B, Chitta S, Likhachev M. Single-and dual-arm motion planning with heuristic search. IJRR. 2014;33(2).

[19] Gochev K, Narayanan V, Cohen B, Safonova A, Likhachev M. Motion planning for robotic manipulators with independent wrist joints. In: Icra. 2014.

[20] Cohen B, Phillips M, Likhachev M. Planning Single-arm Manipulations with n-Arm Robots. In: Rss. 2014.

[21] Hang K, Haustein JA, Li M, Billard A, Smith C, Kragic D. On the evolution of fingertip grasping manifolds. In: Icra. 2016.

[22] Garrett CR, Lozano-Pérez T, Kaelbling LP. Ffrob: An efficient heuristic for task and motion planning. In: Algorithmic foundations of robotics xi. Springer. 2015.

[23] Kavraki LE, Svestka P, Latombe JC, Overmars MH. Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces. In: Ieee transactions on robotics and automation. Vol. 12. 1996.

[24] Kuffner JJ, LaValle SM. RRT-connect: An efficient approach to single-query path planning. In: Icra. Vol. 2. 2000.

[25] Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning. IJRR. 2011;30(7).

[26] Littlefield Z, Bekris KE. Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions. In: Ieee/rsj international conference on intelligent robots and systems (iros). Madrid, Spain. 2018 10/2018.

[27] Siméon T, Laumond JP, Cortés J, Sahbani A. Manipulation planning with probabilistic roadmaps. IJRR. 2004;23(7-8).

[28] Stilman M. Global manipulation planning in robot joint space with task constraints. TRO. 2010; 26(3).

[29] McMahon T, Thomas S, Amato NM. Sampling based motion planning with reachable volumes: Application to manipulators and closed chain systems. In: Iros. 2014.

[30] Vahrenkamp N, Berenson D, Asfour T, Kuffner J, Dillmann R. Humanoid motion planning for dual-arm manipulation and re-grasping tasks. In: Iros. 2009.

[31] Berenson D, Srinivasaz SS. Probabilistically complete planning with end-effector pose constraints. In: Icra. 2010.

[32] Fontanals J, Dang-Vu BA, Porges O, Rosell J, Roa MA. Integrated grasp and motion planning using independent contact regions. In: Humanoids. 2014.

[33] Hang K, Stork JA, Pollard NS, Kragic D. A Framework For Optimal Grasp Contact Planning. RAL. 2017;2(2).

[34] Haustein JA, Hang K, Kragic D. Integrating motion and hierarchical fingertip grasp planning. In: Icra. 2017.

[35] Miller AT, Knoop S, Christensen HI, Allen PK. Automatic grasp planning using shape primitives. In: Icra. Vol. 2. 2003.

[36] Goldfeder C, Ciocarlie M, Dang H, Allen PK. The columbia grasp database. In: Icra. 2009.

[37] Berenson D, Srinivasa SS. Grasp synthesis in cluttered environments for dexterous hands. In: Humanoids. 2008.

[38] Xue Z, Dillmann R. Efficient grasp planning with reachability analysis. IJHR. 2011;8(04).

[39] Liu S, Carpin S. A fast algorithm for grasp quality evaluation using the object wrench space. In: Case. 2015.

[40] Littlefield Z, Zhu S, Kourtev H, Psarakis Z, Shome R, Kimmel A, Dobson A, De Souza AF, Bekris KE. Evaluating end-effector modalities for warehouse picking: A vacuum gripper vs a 3-finger underactuated hand. In: 2016 ieee international conference on automation science and engineering (case). IEEE. 2016. p. 1190–1195.

[41] Michel F, Kirillov A, Brachmann E, Krull A, Gumhold S, Savchynskyy B, Rother C. Global hypothesis generation for 6D object pose estimation. In: Cvpr. 2017.

[42] Mitash C, Boularias A, Bekris KE. Improving 6D Pose Estimation of Objects in Clutter via Physics-aware Monte Carlo Tree Search. In: Icra. 2018.

[43] Zhao M, Shome R, Yochelson I, Bekris K, Kowler E. An experimental study for identifying features of legible manipulator paths. In: Experimental robotics. 2016.

[44] LaValle SM. Planning algorithms. Cambridge university press. 2006.

[45] Zhang L, Kim YJ, Manocha D. C-DIST: efficient distance computation for rigid and articulated models in configuration space. In: Acm spm. 2007.

[46] Spong MW, Hutchinson S, Vidyasagar M, et al.. Robot modeling and control. Vol. 3. wiley New York. 2006.

[47] Littlefield Z, Bekris KE. Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions. In: 2018 ieee/rsj international conference on intelligent robots and systems (iros). IEEE. 2018. p. 1–9.

[48] Buss SR, Kim JS. Selectively damped least squares for inverse kinematics. Journal of Graphics tools. 2005;10(3).

[49] Li Y, Littlefield Z, Bekris KE. Asymptotically optimal sampling-based kinodynamic planning. IJRR. 2016;35(5).

[50] Grinstead C, Snell J. Introduction to Probability. Providence, RI: American Mathmatical Society. 2012.

[51] Beeson P, Ames B. TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In: Humanoids. 2015.

[52] Diankov R. "Automated Construction of Robotic Manipulation Programs". [Ph.D. thesis]. Carnegie Mellon University. 2010.

[53] Littlefield Z, Krontiris A, Kimmel A, Dobson A, Shome R, Bekris KE. An extensible software architecture for composing motion and task planners. In: Simpar. Springer. 2014.

[54] Shome R, Solovey K, Yu J, Bekris KE, Halperin D. Fast and high-quality dual-arm rearrangement in synchronous, monotone tabletop setups. In: Workshop on the algorithmic foundations of robotics (wafr). Mérida, México. 2018 12/2018.

[55] Shome R, Bekris KE. Anytime multi-arm task and motion planning for pick-and-place of individual objects via handoffs. In: 2nd IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS). New Brunswick, NJ, USA. 2019.

[56] Ichter B, Pavone M. Robot motion planning in learned latent spaces. IEEE Robotics and Automation Letters. 2019;.

[57] Calli B, Kimmel A, Hang K, Bekris K, Dollar A. Path planning for within-hand manipulation over learned representations of safe states. In: International symposium on experimental robotics (iser). 2018.