Fast, Anytime Motion Planning for Prehensile Manipulation in Clutter

Andrew Kimmel, Rahul Shome, Zakary Littlefield, Kostas Bekris

***Abstract*— 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 current work integrates tools from existing methodologies and proposes a framework that achieves high success ratio in clutter with anytime performance. The idea is to first explore the lower dimensional end effector's task space efficiently by ignoring the arm, and build a discrete approximation of a navigation function, which guides the end effector towards the set of available grasps or object placements. 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. While informed, the method is also comprehensive and allows the exploration of alternative paths over time if the task space guidance does not lead to a solution. 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.**

I. 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(*top*). 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(*bottom*). One possible explanation is that the clearance of the solutions found in cluttered scenes is low, where clutter 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.

The authors are with the Computer Science Dept. of Rutgers University, 110 Frelinghuysen Road, Piscataway, NJ, USA, {andrew.kimmel, rahul.shome, zakary.littlefield, kostas.bekris}@rutgers.edu.

Fig. 1. An example of manipulation in cluttered table and shelf environments, where the grasped object is partially occluded by other objects.

In problems where solutions exist in workspace clutter, 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.

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 contributions of this work are: 1) developing an effective heuristic for guiding the arm by solving the motion planning problem for just the end effector; 2) applying the pseudo-inverse Jacobian as a steering process which allows the arm to be guided by the heuristic; and 3) incorporating task space guided maneuvers effective in clutter into an asymptotically optimal motion planner.

The performance of `JIST` is evaluated experimentally through comparison with 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.

II. 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

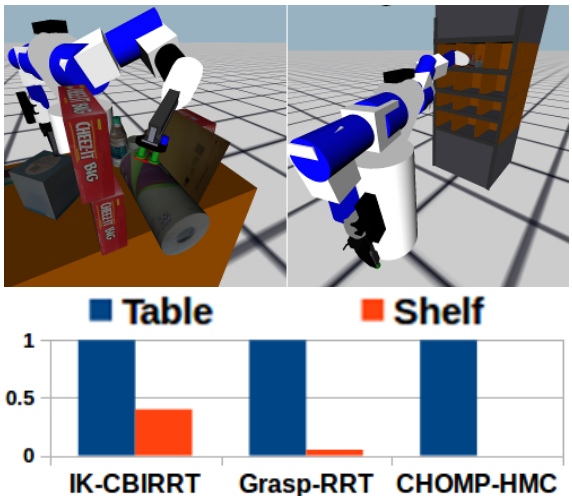

Fig. 2. (top) Two manipulation challenges in a table and shelf environment. (bottom) Success rate (collision-free solution found within 30 seconds) of manipulation motion planning methods averaged over 50 trials.

gradient towards the goal. For arm planning [5], 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 [6]. The typical limitation is the presence of local minima, which can be avoided through integration with a global planner [7] or defining a navigation function [8]. The latter is difficult to find for complex geometric problems [9]. 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][10][11]. 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 [12][13][14]. Heuristics that have been used in the context of manipulation tasks include reachability [15] or geometric task-based reasoning [16]. 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 [17][18] aim to scale better in high dimensions, while providing guarantees, such as asymptotic optimality [19][20]. They incrementally explore the arm's configuration space through sampling. A variety of sampling-based planners has been developed for robotic arms [21][22][23], some of which use the pseudo-inverse of the Jacobian matrix for steering [3] which has been shown to be a probabilistically complete projection onto the grasping manifold [24]. 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 [25][26]. This work focuses on motion planning aspects and uses the output of grasping planners [27][28][29], 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.

III. 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 $\mathbb{O}$.

Assumption 1: The geometric models of object o and obstacles $\mathbb{O}$ are known. Online, before calling the planner, a perception module provides the 6D pose $p_o \in SE(3)$ of o, as well as the 6D poses of obstacles $\mathbb{O}$. □

This setup can be easily extended to point cloud representations, permitting the utilization of recent work on 6D pose estimation [30]. Note that this does not allow precomputation that reasons over the object or obstacle poses, as this information is available only online.

A. Goal Task Space Motion Planning

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

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

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

The pose space $\mathbb{E}$ can be computed given the forward kinematics of the robotic arm $FK : \mathbb{C}_{full} \rightarrow \mathbb{E}$. The inverse kinematics function corresponds to the inverse mapping: $IK : \mathbb{E} \rightarrow \mathbb{C}_{full}$. For redundant manipulators, $dim(\mathbb{E}) < dim(\mathbb{C})$, so IK describes a mapping to self-motion manifolds of the manipulator [24], composed of states, which all have the same end effector pose.

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

For a redundant arm, even a discrete pose set $\mathbb{E}_{goal}$ gives rise to a continuous submanifold of solutions $\mathbb{Q}_{goal}$. Given the above definition, the objective is to solve the following:

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

B. 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 2: A grasping planner produces a set of stable, collision-free grasps for the arm's end effector and the target object o at the object's detected pose p_o. □

A database of grasps can be computed in the object's frame [31][27][28][29] and evaluated in terms of their stability offline [32]. Online, the grasps are transformed given the object pose p_o and collision checked, given the end effector's geometry and the obstacles $\mathbb{O}$. Nevertheless, states that correspond to grasps are on the boundary of $\mathbb{C}_{free}$ with $\mathbb{C}_{obs}$, which result in very hard motion planning problems. For this reason, 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 along a vector away from the object as shown in Fig. 3. 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.

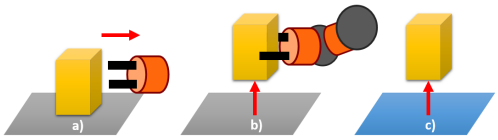

Fig. 3. a) Pre-grasp pose, b) post-grasp state, c) object pre-placement.

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.

Assumption 3: 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 $\mathbb{C}_{free}$. 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.

C. Distance and Cost Function

It has been established that the problem at hand is closely tied to the motions of the end effector, since the goals and heuristic can be described by it. The current work proceeds to define a distance function appropriate for this setting, that minimizes end effector displacement. Another benefit of this is that paths that minimize end effector motion tend to look more natural to human observers who pay more attention to the end effector rather than the rest of the arm [33].

Towards this objective, this work employs the DISP distance function, which is a model-aware distance metric for $SE(3)$, which can be approximated efficiently by the $\mathtt{C-DIST}$ algorithm [34]. 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:

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

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

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

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.

IV. Jacobian Informed Search Tree

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

A. Exploring the End Effector's Pose Space

A simple way to estimate how far the end effector is at pose $e \in \mathbb{E}$ from the set $\mathbb{E}_{goal}$ is to consider the minimum SE(3) distance from e to all goal poses $\mathbb{E}_{goal}$. 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 $\mathbb{E}$ as shown in Fig. 4. The vertices V_{ee} store 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 $\mathbb{E}_{goal}$ and the start pose $e_{start} = FK(q_{start})$ are attached to R_{ee}, which is then searched online for a path from $\mathbb{E}_{goal}$ to e_{start}. Performing the search in this direction produces a multi-start search tree rooted at $\mathbb{E}_{goal}$. During the online search, only collisions between the end effector, the obstacles $\mathbb{O}$, and the object o, but not the arm, are taken into account. This allows for quick estimations of the costs to reach $\mathbb{E}_{goal}$. These costs are used later as heuristics to guide a search procedure in the arm's C-space.

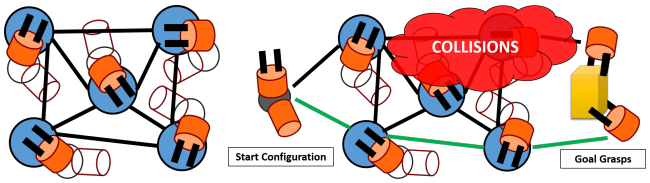

Fig. 4. (left) Construction of R_{ee}. (right) Online search of R_{ee}.

Offline Construction of R_{ee}: 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 $\mathtt{PRM}^*$ [19]. A random arm state $q_r \in \mathbb{C}_{full}$ is sampled and the corresponding end effector pose $FK(q_r) \in \mathbb{E}$ is stored as a vertex. Each vertex is then connected with an edge to the closest $log|V_{ee}|$ vertices in terms of $D(q_r, e)$. Each edge stores an interpolation in SE(3) between the two vertex poses according to a discretization defined by an estimation of the end effector's maximum velocity.

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 $\mathbb{E}_{goal}$ to the start

pose e_{start}. The poses $\mathbb{E}_{goal}$ 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 $\mathbb{E}_{goal}$. Then, each search node corresponds to a path from a pose in $\mathbb{E}_{goal}$ 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 $\mathbb{E}_{goal}$ to u, given the roadmap's edge discretization;
- $f_2(u) = g_2(u) + h_2(u)$, where g_2 is $C(\pi(\mathbb{E}_{goal}, e_u))$ and h_2 is $D(e_u, e_{start})$.

Once a vertex u has been expanded, it is added to the "closed list" $\mathbb{L}_{closed}$. If there is a collision-free end effector path from $\mathbb{E}_{goal}$ to e_{start}, the above search will report the shortest path. Then, the g_2 cost of vertices u in $\mathbb{L}_{closed}$ corresponds to an estimate of the distance along the shortest collision free path from u to $\mathbb{E}_{goal}$, i.e., g_2 is a discrete approximation of a navigation function in $\mathbb{E}$ given the presence of obstacles. The proposed approach uses these g_2 values to heuristically guide the exploration in $\mathbb{C}_{full}$. 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 $\mathbb{L}_{closed}$ will guide the arm exploration along the shortest, least colliding solutions for the end effector.

B. 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" $\mathbb{L}_{closed}$ and the cost estimates g_2 from u to $\mathbb{E}_{goal}$. An illustration of the steps of the algorithm can be seen in Figure 5.

The algorithm is outlined in Alg. 1, which is inspired by a search strategy [20] capable of utilizing heuristic guidance. It receives as input the start arm configuration q_{start}, the goal poses $\mathbb{E}_{goal}$, the end effector reachability roadmap R_{ee}, the number of iterations N, and the parameter of the maximum number κ of greedy edge expansions per iteration. The first step of the method is to execute the multi-start, multi-objective A^* search over R_{ee} outlined in the previous section, in order to acquire the closed list $\mathbb{L}_{closed}$. This contains the vertices of R_{ee} visited during the search and the corresponding cost estimates to the goal poses $\mathbb{E}_{goal}$. The method then builds a search tree whose nodes correspond to arm configurations. The tree is rooted at the start configuration q_{start} (line 3) and is expanded towards configurations that bring the end effector to goal poses.

Each iteration of the algorithm starts by selecting a node of the search tree so as to expand it (lines 6-9). 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 6-7). The heuristic value $h(q, \mathbb{L}_{closed})$ of a configuration q corresponds to an estimate of the cost to reach $\mathbb{E}_{goal}$ approximated using $\mathbb{L}_{closed}$.

If there was no progress made in the previous iteration towards reaching $\mathbb{E}_{goal}$, the SearchSelection subroutine (line 9) 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-go. 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.

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

```
1  L_closed ← MSMO_A*(R_ee, q_start, E_goal);
2  π* ← ∅;
3  T ← {q_start}; q_new ← q_start;
4  A_cand(q_new) ← NULL;
5  for N iterations do
6      if q_new ≠ ∅ and h(q_new, L_closed) <
         h(parent(q_new), L_closed) then
7          q_sel ← q_new;
8      else
9          q_sel ← SearchSelection();
10     if A_cand(q_sel) = NULL then
11         A_cand(q_sel) ←
             GreedyEdges(q_sel, L_closed, κ);
12     else if A_cand(q_sel) = ∅ then
13         A_cand(q_sel) ← FallbackEdges(q_sel);
14     a_best ← argmin_{a∈A_cand} h(a, L_closed);
15     A_cand(q_sel) ← A_cand(q_sel) \ a_best;
16     q_new ← Steer(q_sel, a_best);
17     if BaB(q_new, π*) or CC(q_sel → q_new) then
18         q_new ← ∅;
19     if q_new ≠ ∅ then
20         π* ← AddEdge(T, q_sel → q_new);
21         A_cand(q_new) ← NULL;
22 return π*;
```

Once a configuration q_{sel} is selected for expansion, the set $\mathbb{A}_{cand}$ represents a set of actions that can be used to extend 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 10), the subroutine GreedyEdges is used to generate target poses for extending the tree out of q_{sel} (line 11).

JIST iterates over the generated actions, which are prioritized in terms of the lowest h in terms of the resulting pose and the best one a_{best} is considered for addition at each iteration (line 14). Steer uses an appropriate steering method to generate a trajectory where the state reached at the end corresponds to q_{new} (line 16). The configuration q_{new} must satisfy two conditions so as to be added as a new node (line 17): (a) A branch and bound process (BaB) ensures 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 19), then it is added to the tree T. If a better path to the goal is discovered with the addition of q_{new}, it is recorded as π^* (line 20).

Greedy Edge Generation: These greedy edges are guided

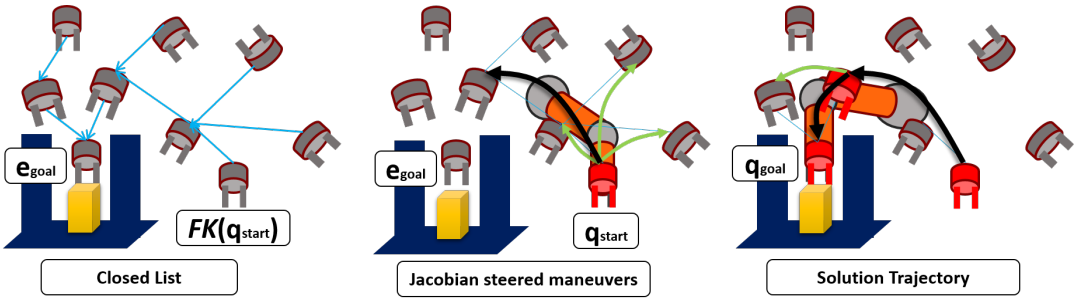

Fig. 5. An illustration of the JIST framework for solving manipulation queries: (left) the closed list $\mathbb{L}_{closed}$ is constructed using a multi-start, multi-objective A* on R_{ee} starting from $\mathbb{E}_{goal}$ to $FK(q_{start})$; (middle) JIST expands arm paths from the start configuration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

by the information stored in the $\mathbb{L}_{closed}$, which contains end effector poses explored during the online search of R_{ee}. First, the nearest end effector pose $e_{near} \in \mathbb{L}_{closed}$ 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}, that belong to $\mathbb{L}_{closed}$ and also have a better heuristic estimate 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 $\mathbb{E}_{goal}$. 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 already been considered, 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 $q_{goal} \in \mathbb{Q}_{goal}$ already discovered by the algorithm as a target to steer towards.

Steering: For the fallback edges, the steering subroutine either uses C-space interpolation to goal arm states, or executes the random controls.

The greedy edges consist of target end effector poses. The approach 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 $\mathbb{J}^+$`steering` is to compute the joint controls which solve $\Delta e = J\Delta q$. The method uses the pseudo-inverse of the Jacobian, $\mathbb{J}^+$, so as to minimize $||J\Delta q - \Delta e||^2$, where $\Delta q = \mathbb{J}^+ \Delta e$. To account for singularities, damping [35] 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 $\mathbb{J}^+$`steering` is:

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

Since the objective is to minimize DISP, the gradient followed by the above rule reduces the distance of the end effector from the the target pose. Any configuration $q_{goal} \in \mathbb{Q}_{goal}$ discovered through $\mathbb{J}^+$`steering` is kept track of. A benefit to using $\mathbb{J}^+$`steering` is that it satisfies the necessary properties of the projection operator [24] needed to ensure coverage of $\mathbb{Q}_{goal}$. $\mathbb{J}^+$`steering` is used in the connection of "pre-grasps", "post-grasps", and "pre-placemements" as described in Section III.

V. Experiments

JIST is compared against other planners for solving manipulation challenges. Recent work [26] has utilized CBiRRT [1] for solving grasping problems. Accordingly, this work compares against an IK-based variant of CBiRRT, which uses IK on the $\mathbb{E}_{goal}$ to construct roots of the goal tree. Grasp-RRT corresponds to an RRT variant, which utilizes $\mathbb{J}^+$`steering` to discover grasp states during the goal biasing phase [3]. CHOMP-HMC corresponds to an OpenRave [36] prob. complete implementation of CHOMP[2].

Setup: A common planning software was used for the sampling-based planners [37]. 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 (7 DOF), Rethink Baxter (14 DOF), and Motoman SDA10F (16 DOF). 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. 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 corresponding time. Solution quality was measured using DISP .

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

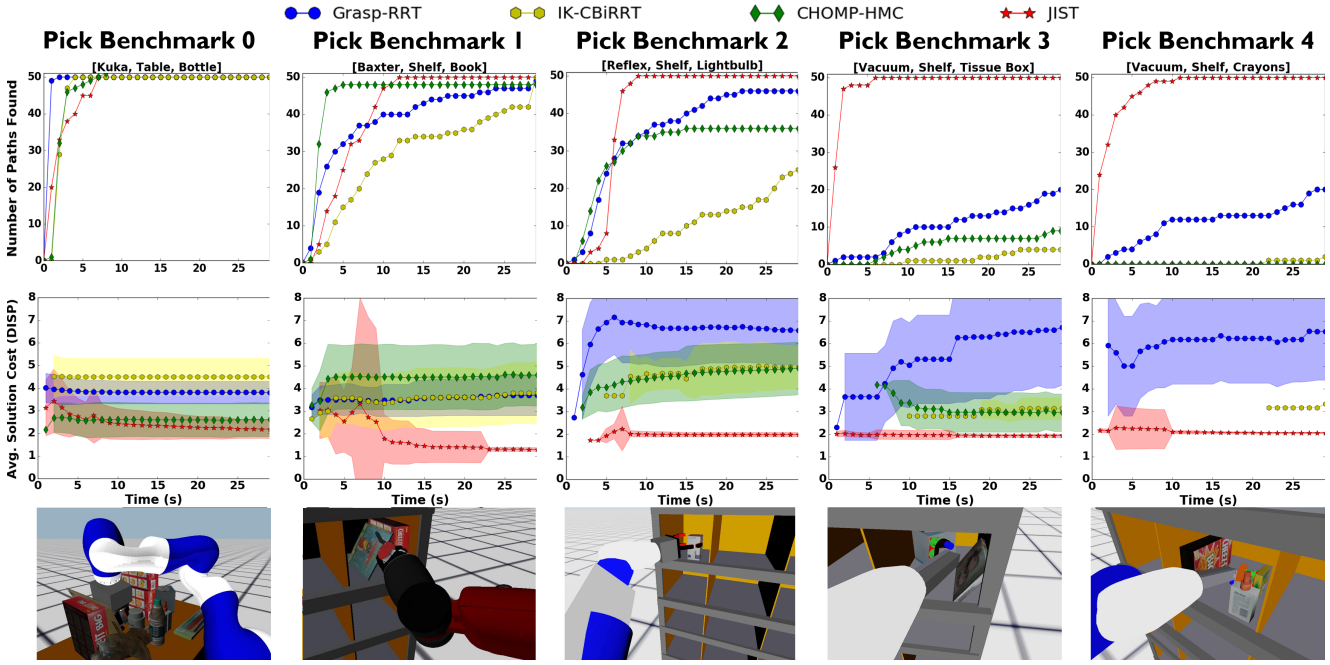

Fig. 6. Success Rates, Solution Costs, and Example Picks for the Kuka+ReFlex (0), Baxter+Parallel (1), Motoman+ReFlex (2), Motoman+Vacuum (3,4)

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, since no single set of parameters sufficed for all. For CHOMP and IK-CBiRRT, reachable, and collision-free arm configurations were computed through IK at goal poses.

Pick Benchmarks: The results for success rate and solution cost over time are shown in Figure 6. 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 7 (left), while also providing the lowest cost solutions in all cases. The end effector heuristic was shown to be effective in guiding the search to produce high quality initial solutions. An interesting side-note here regarding CHOMP is that when initialized with solution states found by JIST and with some additional parameter tuning, the performance of CHOMP improved significantly. This indicates that solutions out of JIST can serve as better initializations for trajectory optimization methods.

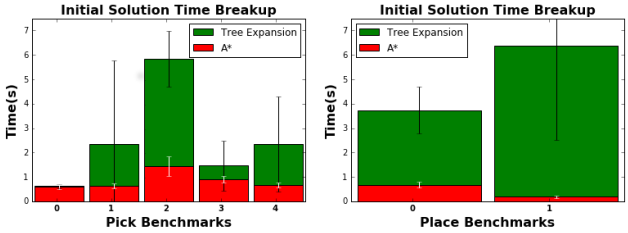

Fig. 7. Breakdown in time for the initial solution using JIST for pick (left) and place (right) benchmarks.

Place Benchmarks: The results for the place benchmarks are shown in Figure 8, with average solution times shown in Figure 7. 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,

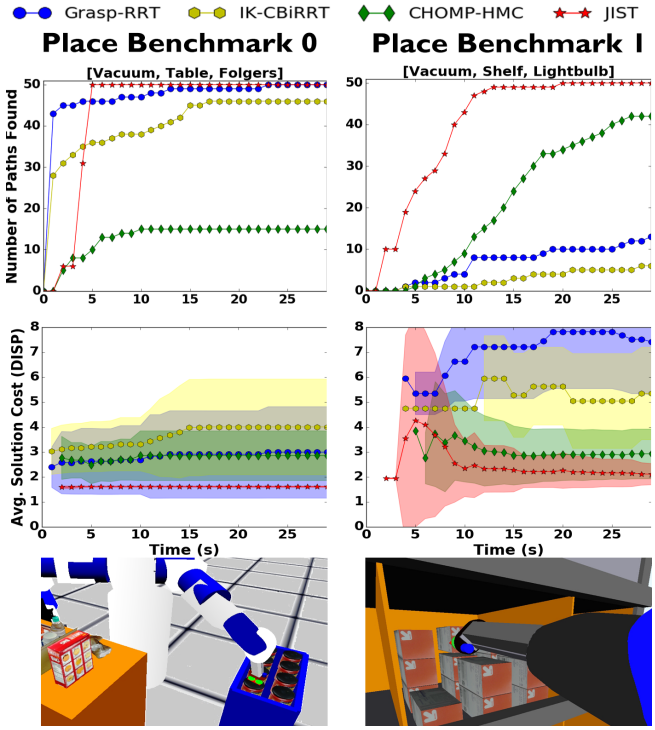

Fig. 8. Success Rates, Solution Costs, and Example Placements for the Vacuum Place Benchmarks in the Table (0) and the Shelf (1) environments.

while also still providing the lowest cost solutions.

Remarks: 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 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.

VI. 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.

There are several directions for further exploration. For instance, goal constraints can be introduced for different arm links, such as a camera attached to the arm. One of the benefits of using Jacobian-based steering is that it allows for the satisfaction of secondary objectives in the null-space of the primary objective to reach the target pose. In this way, JIST could also incorporate additional constraints during task execution, such as maintaining certain orientations of the grasped object or confining the end-effector to a particular workspace region. Recent work in more complex IK-solvers could also be utilized to more efficiently guide the search tree [38]. Another direction is the consideration of alternative cost functions, especially time-based and multi-objective functions. Finally, it is interesting to evaluate the performance of JIST without knowledge of the obstacle geometries by operating directly over point clouds [39].

References

[1] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *ICRA*, 2009.

[2] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, A. Bagnell, and S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *IJRR*, vol. 32, no. 9, 2013.

[3] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Simultaneous grasp and planning: Humanoid robot ARMAR-III," *RAM*, vol. 19, no. 2, 2012.

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

[5] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *IJRR*, vol. 5, no. 1, 1986.

[6] P. Song and V. Kumar, "A potential field based approach to multi-robot manipulation," in *ICRA*, vol. 2, 2002.

[7] C. W. Warren, "Global path planning using artificial potential fields," in *ICRA*, 1989.

[8] E. Rimon and D. E. Koditschek, "Exact robot navigation using cost functions: the case of distinct spherical boundaries in E/sup n," in *ICRA*, 1988.

[9] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *ICRA*, 1991.

[10] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization." in *RSS*, 2013.

[11] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs." in *RSS*, 2016.

[12] B. Cohen, S. Chitta, and M. Likhachev, "Single-and dual-arm motion planning with heuristic search," *IJRR*, vol. 33, no. 2, 2014.

[13] K. Gochev, V. Narayanan, B. Cohen, A. Safonova, and M. Likhachev, "Motion planning for robotic manipulators with independent wrist joints," in *ICRA*, 2014.

[14] B. Cohen, M. Phillips, and M. Likhachev, "Planning Single-arm Manipulations with n-Arm Robots," in *RSS*, 2014.

[15] K. Hang, J. A. Haustein, M. Li, A. Billard, C. Smith, and D. Kragic, "On the evolution of fingertip grasping manifolds," in *ICRA*, 2016.

[16] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015.

[17] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces," in *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, 1996.

[18] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, 2000.

[19] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *IJRR*, vol. 30, no. 7, 2011.

[20] Z. Littlefield and K. E. Bekris, "Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 10/2018 2018.

[21] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *IJRR*, vol. 23, no. 7-8, 2004.

[22] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *TRO*, vol. 26, no. 3, 2010.

[23] T. McMahon, S. Thomas, and N. M. Amato, "Sampling based motion planning with reachable volumes: Application to manipulators and closed chain systems," in *IROS*, 2014.

[24] D. Berenson and S. S. Srinivasaz, "Probabilistically complete planning with end-effector pose constraints," in *ICRA*, 2010.

[25] J. Fontanals, B.-A. Dang-Vu, O. Porges, J. Rosell, and M. A. Roa, "Integrated grasp and motion planning using independent contact regions," in *Humanoids*, 2014.

[26] J. A. Haustein, K. Hang, and D. Kragic, "Integrating motion and hierarchical fingertip grasp planning," in *ICRA*, 2017.

[27] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *ICRA*, 2009.

[28] D. Berenson and S. S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *Humanoids*, 2008.

[29] Z. Xue and R. Dillmann, "Efficient grasp planning with reachability analysis," *IJHR*, vol. 8, no. 04, 2011.

[30] C. Mitash, A. Boularias, and K. E. Bekris, "Improving 6D Pose Estimation of Objects in Clutter via Physics-aware Monte Carlo Tree Search," in *ICRA*, 2018.

[31] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *ICRA*, vol. 2, 2003.

[32] S. Liu and S. Carpin, "A fast algorithm for grasp quality evaluation using the object wrench space," in *CASE*, 2015.

[33] M. Zhao, R. Shome, I. Yochelson, K. Bekris, and E. Kowler, "An experimental study for identifying features of legible manipulator paths," in *Experimental Robotics*, 2016.

[34] L. Zhang, Y. J. Kim, and D. Manocha, "C-DIST: efficient distance computation for rigid and articulated models in configuration space," in *ACM SPM*, 2007.

[35] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *Journal of Graphics tools*, vol. 10, no. 3, 2005.

[36] R. Diankov, ""Automated Construction of Robotic Manipulation Programs"," Ph.D. dissertation, Carnegie Mellon University, 2010.

[37] Z. Littlefield, A. Krontiris, A. Kimmel, A. Dobson, R. Shome, and K. E. Bekris, "An extensible software architecture for composing motion and task planners," in *SIMPAR*. Springer, 2014.

[38] D. Rakita, B. Mutlu, and M. Gleicher, "RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[39] A. Kuntz, C. Bowen, and R. Alterovitz, "Fast Anytime Motion Planning in Point Clouds by Interleaving Sampling and Interior Point Optimization," in *ISRR*, 2017.