Optimal, Sampling-Based Manipulation Planning

Philipp S. Schmitt[1], Werner Neubauer[1], Wendelin Feiten[1],
Kai M. Wurm[1], Georg v. Wichert[1], and Wolfram Burgard[2]

***Abstract*— When robots perform manipulation tasks, they need to determine their own movement, as well as how to grasp and release an object. Reasoning about the motion of the robot and the object simultaneously leads to a multi-modal planning problem in a high-dimensional configuration space.**

In this paper we propose an asymptotically optimal manipulation planner. Our approach extends optimal sampling-based roadmap planners to efficiently explore the configuration space of the robot and the object. We prove probabilistic completeness and global, asymptotic optimality.

Extensive simulations of a typical pick-and-place scenario show that our approach significantly outperforms a (non-optimal) state-of-the-art approach. We implemented our planner on a real manipulator and were able to compute high quality solutions in less than a second.

I. INTRODUCTION

In manufacturing, logistics, and other industrial domains, many tasks involve handling of objects. To automate such tasks using a robotic manipulator, object positions are usually fixed so that robot movements can be hard-coded into the automation solution. Whenever objects, positions or tasks change frequently this approach cannot be applied. Potentially, the manipulation task at hand has never been encountered before. In such cases, the actions of the robot need to planned autonomously. An illustration of a manipulation task with one object is given in Fig. 1.

The naive approach is to sequentially plan grasps, placements, inverse kinematics and motions. These steps, however, are not independent, e.g., grasps can only be executed if a corresponding configuration of the manipulator can be computed. In addition, these solution will in general not be optimal.

Solving the combination of these planning problems in one batch is known as manipulation planning [1]. In practice, a manipulation planner has to cope with the following challenges:

- Continuous grasps, placements and actions: In typical manipulation scenarios, the objects to be manipulated can be grasped or placed in a continuous set of contact states. In addition, redundant manipulators allow transitioning from grasp to placement or vice versa in a continuous space of configurations. Manipulation actions such as grasping may also be parametrized with continuous variables. Discretizing these spaces quickly leads to a combinatorial explosion, while selecting grasps or placements manually defeats the point of planning.
- Incomplete motion planning: Motion planning for manipulation requires planning in high-dimensional, continuous configuration spaces. Furthermore it is computationally expensive to decide which parts of the configuration space are valid, i.e., at least collision free. The state of the art approach to these issues is sampling-based motion planning. Planners such as Rapidly Exploring Random Trees (RRT) [2] or Probabilistic Roadmaps (PRM) [3] are only probabilistically complete. Therefore they cannot decide within finite time if a motion planning problem has no solution or find an optimal solution. Any practical manipulation planner must handle this incompleteness of motion planning.
- Optimality: In order to be an efficient replacement for human labor, autonomous manipulation must provide high quality or, ideally, optimal robot paths. While the notion of optimality may change from one application to another, neglecting the solution quality of a planner is unacceptable for real life applications.

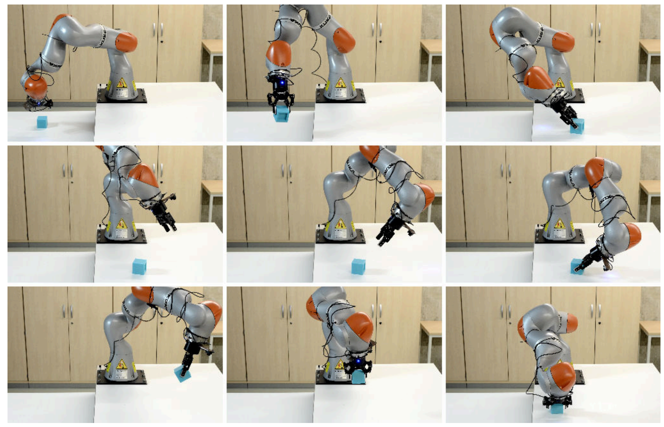

Fig. 1. A manipulation problem - The task of the robot is to bring the small cube onto the table and to place it bottom-up. To do so, the cube must be (re-)grasped and placed correctly at least twice.

Despite the importance of robotic manipulation, to the best of our knowledge, there currently exists no planner that copes with all of these challenges. The contribution of this paper is an asymptotically optimal, sampling-based manipulation planner. We prove probabilistic completeness as well as optimality and validate our approach in thorough experiments, both in simulation and on a real robot.

[1] Siemens Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany
[2] Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany

II. RELATED WORK

Our work integrates and extends two strands within the planning literature: Manipulation planning and optimal, sampling-based motion planning.

A. Manipulation Planning

The first approach to manipulation planning presented by Alami et al. [1] considers the planning problem in which an object is either at a stable placement or grasp. Placements and grasps are chosen from a finite set. Manipulation is then formulated as a graph search with alternating transit and transfer paths, for which individual motion planning queries must be solved. Continuous grasps and placements are addressed in [4]. This is done by modeling the connected components within the intersection of stable placements and grasps as closed-chain systems. The incompleteness of sampling-based motion planners is addressed by Hauser [5], by building a roadmap of roadmaps. This idea is then extended to the Probabilistic Tree of Roadmaps (PTR) planner in [6] to also address continuous contact states. In [7] a RRT-like planner for the same problem class is proposed.

To address a wider class of planning problems, Cambon et al. [8] propose a hybrid planner to integrate task and motion planning. Dornhege et al. [9] propose an extension to the PDDL standard [10] to combine symbolic and motion planning via semantic attachments. Garret et al. [11] reformulate heuristics from the symbolic planning domain [12] for task and motion planning with very long running tasks. In [13], a second heuristic guided planner is proposed, that is capable of handling continuous contact states and the incompleteness of sampling based motion planning.

In order to improve plan quality, Harada et al. [14] propose a post-processing method that attempts to shorten manipulation plans. None of the above mentioned approaches is shown to achieve global optimality for both the sequence of grasps and placements and the intermediate motions.

B. Optimal, Sampling-Based Motion Planning

Planning for robotic manipulators typically requires sampling-based motion planners, such as RRTs [2] or PRMs [3]. Karaman and Frazzoli [15] provide a proof of the sub-optimality of these approaches as well as asymptotically optimal counterparts of the original planners (RRT* and PRM*).

For these planners several improvements and extensions have been proposed. To speed up convergence, heuristic guidance during sampling is introduced in [16]. Hauser [17] uses lazy collision checking to defer expensive computations until a better path has been found. Optimal, sampling-based kinodynamic planning is addressed in [18]. An extension of the original RRT* for closed kinematic chains is presented in [19].

Vega-Brown and Roy proposed Factored Orbital Bellman Trees [20], an optimal planner for problems with piecewise analytic constraints. To be computationally tractable, it requires a factorized sampling strategy. The authors provide such a strategy for a block-pushing scenario, but no extension to prehensile manipulation with articulated robots or redundant manipulators is provided.

III. PROBLEM STATEMENT AND NOTATION

A. Planning Problem

We consider the problem of prehensile manipulation of a single rigid object with a robotic manipulator. Let $\mathcal{C}_r$ be the configuration space of the robot, $\mathcal{C}_o$ that of the object and $\mathcal{C}_r \times \mathcal{C}_o$ their joint configuration space. At all times, the object is rigidly attached to a link of the robot or its static environment. This attachment, along with the necessary transforms, is denoted by a contact state $\sigma \in \Sigma$. The set of contact states Σ may include placements (attachment to the static environment) and grasps (attachment to a gripper). An object configuration $c_o \in \mathcal{C}_o$ can be computed by $c_o = f_k(c_r, \sigma)$, where $f_k(c_r, \sigma)$ denotes the forward kinematic. We abbreviate a joint configuration of robot and object $(c_r, c_o) = (c_r, f_k(c_r, \sigma))$ by (c_r, σ).

Let $\mathcal{C}_{\text{free},\sigma} \subseteq \mathcal{C}_r$ denote the set of robot configurations for which (c_r, σ) is a valid (i.e., collision free) joint configuration. A change between two contact states σ_1 and σ_2 is only allowed at specific transition regions of the robot configuration space: $\mathcal{C}_{\sigma_1,\sigma_2} \subseteq \mathcal{C}_r$. For a placement σ_1 and a grasp σ_2, the transition region might be the set of inverse kinematic solutions that allow grasping the object at placement σ_1 with grasp σ_2. These regions typically form lower-dimensional manifolds of the configuration space of the robot and are typically empty sets for most pairs of contact states.

Let $\pi : [0,1] \rightarrow \mathcal{C}_r$ be a continuous path segment in the robot configuration space. A path $\{(\pi_i, \sigma_i)\}_{i \leq k} \in \Pi$ with $i, k \in \mathbb{N}_{>0}$ is a sequence of path segments and contact states of length k.

Definition 1 (Valid Path) *A path is valid iff* $\pi_i(\tau) \in \mathcal{C}_{free,\sigma_i}$ *for* $\tau \in [0,1], i \in 1...k$ *and* $\pi_i(1) = \pi_{i+1}(0)$ *and* $\pi_i(1) \in \mathcal{C}_{\sigma_i,\sigma_{i+1}}$ *for* $i \in 1...k-1$.

Given an initial joint configuration $(c_{\text{start}}, \sigma_{\text{start}})$ and an end-game region $\mathcal{C}_{\text{goal}} \subset \mathcal{C}_r \times \mathcal{C}_o$, feasible paths can be defined.

Definition 2 (Feasible Path) *A path is feasible iff it is valid and* $(\pi_1(0), \sigma_1) = (c_{start}, \sigma_{start})$ *and* $(\pi_k(1), \sigma_k) \in \mathcal{C}_{goal}$.

The cost function $C : \Pi \rightarrow \mathbb{R}_{>0}$ is defined as follows:

$$C : \{(\pi_i, \sigma_i)\}_{i \leq k} \rightarrow \sum_{i=1}^{k} C_p(\pi_i) + \sum_{i=1}^{k-1} C_t(\sigma_i, \sigma_{i+1}) \quad (1)$$

Where C_p assigns non-negative cost to path segments and $C_t > C_{t,min} > 0$ assigns lower bounded positive cost to transitions.

Definition 3 (Optimal Path) *A path* $\{(\pi_i^*, \sigma_i^*)\}$ *is optimal iff it is feasible and*

$$C(\{(\pi_i^*, \sigma_i^*)\}) = \min_{\{(\pi_i,\sigma_i)\} \in \Pi} C(\{(\pi_i, \sigma_i)\})$$

The tasks of feasible and optimal manipulation planning are now to determine feasible or optimal paths respectively.

B. Primitive Operations

We assume that the following primitive operations are available to our planner. A call to **sampleFree**(σ) returns a random robot configuration within $\mathcal{C}_{\text{free},\sigma}$ and can be implemented via rejection sampling with a collision check. The operation **sampleTransition**(σ_1, σ_2) returns, if possible, a random configuration within $\mathcal{C}_{\sigma_1,\sigma_2}$ that allows a transition between contact states σ_1 and σ_2. This requires sampling an inverse kinematics solution. Finally, **sampleContact** returns a random contact state $\sigma \in \Sigma$ i.e., a grasp or a placement. In order to keep our algorithm and the proofs of its properties as general as possible, we treat these primitive operations as problem specific black-boxes.

IV. OPTIMAL MANIPULATION PLANNER

In this section we introduce our planning approach, Random Manipulation Roadmap-star (RMR*) and discuss measures to speed up roadmap construction.

A. Algorithm

Solving a manipulation planning problem is done in two phases. In an offline phase, a large roadmap is constructed to explore the topology of the manipulation planning problem. It is detailed in the **buildRoadmap** procedure and needs only to be called once for one pair of robot and object geometries. This algorithm builds a weighted undirected graph (V, E). The set of vertices V contains valid configurations (c, σ) and the set of edges E contains cost-weighted pairs of vertices in V. For brevity we omit necessary sanity checks within our algorithm. If it is not possible to sample a configuration or a transition, the succeeding operations are not called.

procedure buildRoadmap (N_c, N_i, N_t)
  for N_c **do**
    $\sigma_S \leftarrow$ sampleContact( )
    Σ_S.append (σ_S)
    buildPRM*(σ_S, N_i)
  for $\sigma_{S,1} \neq \sigma_{S,2} \in \Sigma_S$ **do**
    connectRoadmaps($\sigma_{S,1}, \sigma_{S,2}, N_t$)

This procedure takes three integers, N_c, N_i and N_t, as input to specify the size of the graph. It constructs several within-contact roadmaps, that are subsequently connected via transitions. To this end it samples N_c contact states, which are added to a list Σ_S. For each of these contact states $\sigma_S \in \Sigma_S$ a within-contact roadmap is build within $\mathcal{C}_{\text{free},\sigma_S}$. This is done according to the PRM*-algorithm [15] with N_i samples in procedure **buildPRM***. Procedure **connectConfiguration**(c, σ) tries to connect a joint configuration (c, σ) to previously sampled configurations with the same contact state σ on valid paths within $\mathcal{C}_{\text{free},\sigma}$ according to the PRM* algorithm.

procedure buildPRM* (σ, N_i)
  for N_i **do**
    $c \leftarrow$ sampleFree(σ)

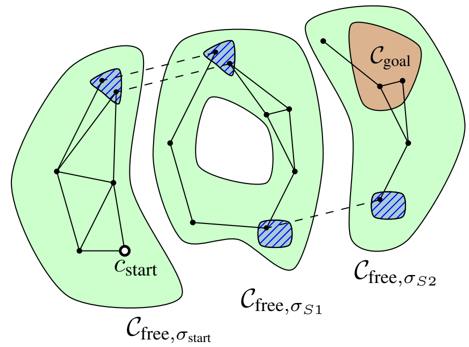

Fig. 2. Roadmap construction of RMR* - The green areas visualize the valid areas $\mathcal{C}_{\text{free},\sigma_x}$ of the robot configuration space for three different contact states σ_x. The striped areas show transition regions $\mathcal{C}_{\sigma_x,\sigma_y}$ Dots mark nodes of the roadmaps built by RMR*. Solid lines mark edges for which the robot moves, but the contact state stays constant. Dashed lines mark a change of contact state.

    V.append((c, σ))
    E.append(connectConfiguration(c, σ))

For each pair of contact states $\sigma_{S,1} \neq \sigma_{S,2} \in \Sigma_S$ we then attempt to sample N_t transitions and connect them to the corresponding roadmaps using **connectRoadmaps**. A schematic illustration of the roadmaps built by our algorithm can be seen in Fig. 2.

procedure connectRoadmaps $(\sigma_1, \sigma_2, N_t)$
  for N_t **do**
    $c \leftarrow$ sampleTransition(σ_1, σ_2)
    V.append($(c, -)$)
    E.append(connectConfiguration(c, σ_1))
    E.append(connectConfiguration(c, σ_2))

As the time complexity of building a PRM* with n nodes is $\mathcal{O}(n \log n)$, the time complexity of building the manipulation roadmap is $\mathcal{O}((N_c N_t + N_i) N_c \log N_i)$.

The second phase of our algorithm is the query phase. In the spirit of the original PRM algorithm [3], we first attempt to connect a start configuration $(c_{\text{start}}, \sigma_{\text{start}})$ to the previously constructed manipulation roadmap. Then we use a standard graph search algorithm to find the minimal-cost path to an endgame region $\mathcal{C}_{\text{goal}}$. Typically, the initial contact state σ_{start} is not an element of the sampled contact states Σ_S. Therefore, it is necessary to construct a docking roadmap with N_i nodes using the PRM*-algorithm. The start configuration is then connected to this docking roadmap which in turn is connected to the manipulation roadmap with at most N_t transitions per contact state in Σ_S.

procedure query($(c_{\text{start}}, \sigma_{\text{start}}), \mathcal{C}_{\text{goal}}, N_i, N_t$)
  V.append($(c_{\text{start}}, \sigma_{\text{start}})$)
  buildPRM*(σ_{start}, N_i)
  E.append(connectConfiguration($c_{start}, \sigma_{start}$))
  for $\sigma_S \in \Sigma_S$ **do**
    connectRoadmaps($\sigma_{\text{start}}, \sigma_S, N_t$)
  return GraphSearch($(c_{\text{start}}, \sigma_{\text{start}})$, (V, E), $\mathcal{C}_{\text{goal}}$)

Building the docking roadmap and connecting it to the manipulation roadmap has a time complexity of $\mathcal{O}((N_c N_t + N_i) \log N_i)$. The following upper bounds hold for the number of vertices and edges within (V, E):

$$|V| < N_c(N_c N_t + N_i) \quad (2)$$

$$|E| < (N_c N_t + N_i) N_c \log N_i \quad (3)$$

These inequalities can be used to estimate the time complexity of the graph search via Equation 4 [21].

$$\mathcal{O}(|E| + |V| \log |V|) \quad (4)$$

B. Roadmap Re-Use and Lazy Collision Checking

Even though the offline phase of our algorithm must only be called once, its runtime quickly becomes a bottleneck as hundreds of contact states and thousands of roadmap nodes per contact are sampled. To reduce runtime, we re-use collision checks and nearest neighbor searches across the construction of the N_c within-contact roadmaps. Furthermore we employ lazy collision checking [22].

Instead of building the within-contact roadmaps at line 6 of **buildRoadmap**, we first build a PRM* with N_i nodes for a planning scene with no object. We can then check which nodes and edges of this roadmap lie within $\mathcal{C}_{\text{free},\sigma}$ for all $\sigma \in \Sigma_S$. This has several advantages. We will never sample a configuration or try to connect an edge for which the robot is in self-collision or collides with its static environment. Furthermore, nearest-neighbor search can be shared across all N_c roadmaps and the necessary data-structures for search can be reused for the connection of transitions.

As a second measure, we do not check if an edge of the within-contact roadmaps is valid during roadmap construction. We employ the graph search algorithm at the end of procedure **query** with the assumption that all edges are valid. If a path is returned, we check only edges on this path. Should one edge be invalid, we remove it from the set of edges E and repeat the graph search.

V. Completeness and Optimality

In order to prove probabilistic completeness and asymptotic optimality we need to define some properties of the planning problems.

Definition 4 (Segment Robustness) *A triple (c_{r1}, c_{r2}, σ) is segment robust iff the PRM* algorithm is asymptotically optimal while planning from c_{r1} to c_{r2} within $\mathcal{C}_{free,\sigma}$.*

Definition 5 (Goal Robustness) *A tuple (c_r, σ) is goal robust iff $\mathcal{C}_{free,\sigma} \times \{\sigma\} \cap \mathcal{C}_{goal} \neq \emptyset$ and the PRM* algorithm is asymptotically optimal while planning from c_r to $\mathcal{C}_{goal}$ within $\mathcal{C}_{free,\sigma}$.*

The required conditions for segment and goal robustness are omitted for brevity and can be found in [15].

Let $\{\sigma_i\}_{i \le k}$ with $i, k \in \mathbb{N}_{>0}$ be a sequence of contact states. Furthermore, let $\{t_j\}_{j<k}$ with $j \in \mathbb{N}_{>0}$ be a sequence of robot configurations that allow a transition between these contact states. We define C^* as the cost of the optimal solution to our planning problem. The value $C^*(\{\sigma_i\})$ denotes the cost of an optimal path using only the contact states within $\{\sigma_i\}$. Finally, $C^*(\{\sigma_i\}, \{t_j\})$ is defined as the optimal path cost using only the contacts in $\{\sigma_i\}$ and transitions in $\{t_j\}$. These definitions imply: $C^* \le C^*(\{\sigma_i\}) \le C^*(\{\sigma_i\}, \{t_j\})$.

Definition 6 (Transition Robustness) *A pair of a configuration and contact sequence $((c_{start}, \sigma_{start}), \{\sigma_i\})$ is transition robust iff: For every $\epsilon \in \mathbb{R}_{>0}$ there exists a probability $P_\epsilon > 0$, such that when sampling a sequence of transitions $\{t_j\}$ using* ***sampleTransition*** *the following holds at least with probability P_ϵ:*

- *$C^*(\{\sigma_i\}, \{t_j\}) \le C^*(\{\sigma_i\}) + \epsilon$*
- *(t_{k-1}, σ_k) is goal robust.*
- *All consecutive pairs of c_{start} and t_j are segment robust within the corresponding contact states.*

Intuitively, transition robustness states that sampling transitions between contacts and connecting them to PRM* roadmaps within these contacts is equivalent to building one large PRM*.

As transition costs are lower bounded by $C_{t,min} > 0$, an optimal path, if one exists, will have a finite number of contact states. Let k^* be the largest number of contacts in any of the optimal paths.

Definition 7 (Contact Robustness) *A planning problem is contact robust iff: For every $\epsilon \in \mathbb{R}_{>0}$ there exists $P_\epsilon > 0$, such that when sampling a sequence of contacts $\{\sigma_i\}_{i \le k^*}$ using* ***sampleContact*** *the following holds at least with probability P_ϵ:*

- *$C^*(\{\sigma_i\}) \le C^* + \epsilon$*
- *$((c_{start}, \sigma_{start}), \{\sigma_i\})$ is transition robust.*

For problems, that have a solution, the random variable $C_{\text{RMR*}}(N_c, N_i, N_t)$ is defined as the solution cost returned by our planner. If our planner fails this variable is set to $C_{\text{fail}} \gg C^*$.

Theorem 1 (Optimality of RMR*) *For a planning problem that is contact robust, RMR* almost surely converges to an optimal solution as N_c, N_i and N_t approach infinity.*

$$P(\lim_{N_C, N_i, N_t \to \infty} C_{RMR*}(N_c, N_i, N_t) = C^*) = 1$$

Proof: Let $\epsilon > 0$. From the contact robustness of our planning problem follows, that RMR* will almost surely sample a sequence of k^* contacts which is transition robust and for which $C^*(\{\sigma_i\}) \le C^* + \epsilon$ holds, as N_c approaches infinity.

From the transition robustness of this sequence follows, that RMR* will almost surely sample a set of $k^* - 1$ transitions which are consecutively segment robust, $(t_{k^*-1}, \sigma_{k^*})$ is goal robust and for which $C^*(\{\sigma_i\}, \{t_j\}) \le C^*(\{\sigma_i\}) + \epsilon$ holds, as N_t approaches infinity.

Between consecutive pairs of start configuration and transitions our algorithm will build increasingly larger PRM* roadmaps, which are asymptotically optimal in N_i due to segment robustness. Within contact state σ_{k^*} PRM* is asymptotically optimal in N_i due to goal robustness. Therefore $C_{\text{RMR*}}(N_c, N_i, N_t)$ will, almost surely, not exceed $C^*(\{\sigma_i\}, \{t_j\})$ by more than ϵk^*, as N_i approaches infinity.

Therefore RMR* will almost surely return a solution that is no larger than $C^* + \epsilon(2 + k^*)$ for any $\epsilon > 0$. This implies asymptotic optimality. ■

Theorem 2 (Probabilistic Completeness of RMR*) *For a planning problem that is contact robust, RMR* is probabilistically complete as N_c, N_i and N_t approach infinity.*

This theorem follows trivially from asymptotic optimality. The discerning reader will have noticed, that our proofs do not build upon the underlying physical mechanics of manipulation, like the distribution of placements or grasps in $\mathcal{C}_o$ or the shapes of $\mathcal{C}_{\sigma_1,\sigma_2}$. Instead we make assumptions on probabilities of sampling contacts or transitions in helpful areas of $\mathcal{C}_o$ and $\mathcal{C}_r$. We argue that the operations **sampleContact** and **sampleTransition** are typically not part of the solution but part of the problem setting. For this reason we do not build our proofs upon assumptions that lie within the mechanics of these operations.

VI. EXPERIMENTS

A. Experimental Setup and Implementation

The industrial manipulator used throughout our experiments consists of a 7-axis, redundant robotic arm, a parallel gripper, and a monocular camera.

We designed seven benchmark tasks that revolve around moving a cube into a target area on a table or into a box. For some of the tasks it is necessary to re-grasp the object several times in order to change its orientation.

The benchmarks are designed to pose problems with different levels of difficulty. It is well known that narrow tunnels in the configuration space pose difficult problem instances for sampling based planners [23]. In our experiments we add such tunnels by design, both in the configuration space of the robot (picking and placing in a box) and in the contact space of the object (changing the orientation of the object).

The experimental setup and one of the two initial positions of the object can be seen in Fig. 3. In all experiments the joints of the robot are initially at their zero position. The blue areas mark the two goal regions for the object. Table I lists the seven benchmark tasks used in our experiments. As cost function for the path segments C_p we chose the euclidean distance traveled in the robot configuration space. The cost for transitions C_t was arbitrarily chosen to be 3.0 for all transitions to discourage unnecessary re-grasps.

Grasps and placements of the cube are randomly distributed around its six faces. Placements are only sampled within the areas marked blue in Fig. 3. Transitions are computed via rejection sampling. We use an analytic inverse kinematics solver, for which the redundancy parameters are randomly chosen.

To evaluate our planner we compare the quality of its solutions to that of the Probabilistic Tree of Roadmaps (PTR)

TABLE I

BENCHMARK PROBLEMS

#	Initial Object Position	Goal Area	Goal Orientation
1	lower table	upper table	any
2	lower table	upper table	bottom up
3	lower table	box	any
4	lower table	box	bottom up
5	box	upper table	any
6	box	upper table	bottom up
7	box	box	bottom up

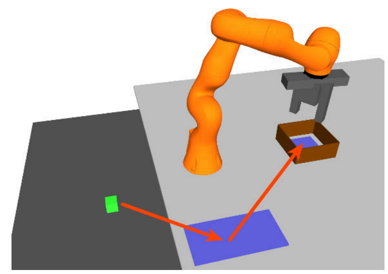

Fig. 3. Experimental Setup: The cube on the left must be brought into one of goal regions marked in blue. The arrows depict the optimal manipulation sequence for benchmark 4. First, the object is placed on the table with a 90 degree turn. Then it is re-grasped and placed bottom up in the box.

planner presented in [6]. This approach from the literature was chosen for two reasons: It is capable of handling manipulation planning problems with continuous contact states without requiring complete motion planners. Furthermore, in its basic form, it does not require domain-specific adaptations, like heuristics, that are hand crafted for the problem class at hand.

Both algorithms are then used to solve all of the seven benchmark tasks. Our approach is run with 25 different settings for N_c, N_t, and N_i. Due to the probabilistic nature of the two algorithms, we repeat this experiment 30 times.

The solutions of both approaches are additionally run through a standard path simplification [24].

The implementation of the algorithms uses the open source library FCL [25] for collision checks which is accessed via the planning scene of MoveIt! [26]. For nearest neighbor search we use randomized k-d trees [27] implemented in the FLANN library [28]. To speed up roadmap construction it is run in parallel using OpenMP [29]. All experiments are run on a ten-core Intel Xeon E5-2650v3.

B. Results

Fig. 4 shows the success rates of our approach on all seven benchmarks with 25 different parameter settings. We have chosen to increase the input parameters N_c, N_i and N_t in a linear fashion. At their highest setting, we build a manipulation roadmap with 250 contact states, 2500 nodes per within-contact roadmap and at most 25 transitions between each pair of contact states. As one can see, the success rates quickly converge towards one for all benchmark tasks.

Fig. 5 depicts the average cost for successful queries and indicates convergence in solution cost for all seven benchmarks. To visualize the distribution of costs, Fig. 6 shows a series of box plots of the solution costs returned for benchmark 7. This benchmark is the most difficult one

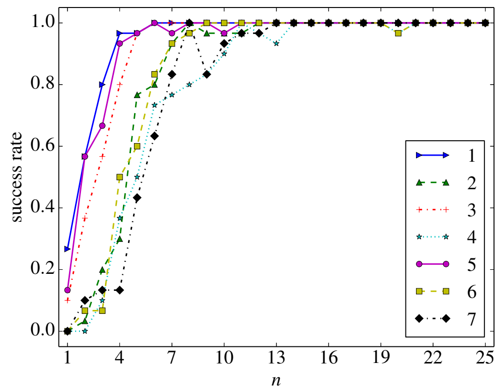

Fig. 4. Success rates for the seven benchmark tasks - Each line visualizes the success rate for one benchmark with different parameter settings for our algorithm: $N_c = 10n$, $N_i = 100n$, $N_t = n$.

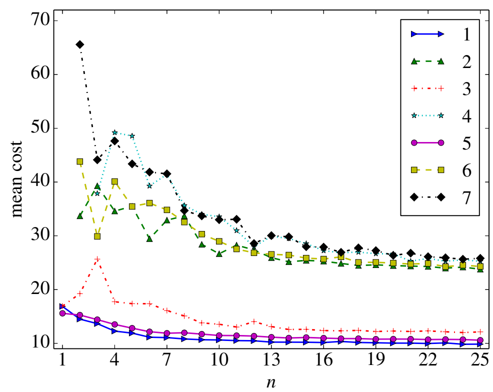

Fig. 5. Path costs without post-processing for the seven benchmark tasks - Each line visualizes the average costs of successful runs for one benchmark with different parameter settings: $N_c = 10n$, $N_i = 100n$, $N_t = n$.

within our experiments, as the object has to be grasped and placed at least two times and the robot must go through two tunnels while picking from and placing into the box. As one can see, not only the average cost, but also its variance is reduced as the parameter settings are increased.

The average times for roadmap construction and query needed to achieve these results are depicted in Fig. 7. It can be seen, that highly reliable and close to optimal planning is possible with query times below one second.

To compare the solution quality of our approach to that of the Probabilistic Tree of Roadmaps (PTR) planner, Table II shows the results of both approaches across our benchmark tasks. The table depicts the average cost and the standard error of the mean (in brackets) for both planners, with and without post-processing. Our planner is run with the maximum settings from the previous experiments: $N_c = 250$, $N_i = 2500n$, $N_t = 25$. Both planners are run 30 times.

We analyzed the resulting data-set under the assumption of independent Gaussian-distributions with unequal variance for our samples. Significance levels were therefore computed via Welch's unequal variances t-test. Two observations can be made. Our planner significantly (at 0.1% level) outperforms the existing one in all seven tasks, both with and without post-processing. Furthermore the post-processing significantly (also at 0.1% level) improves the solution cost

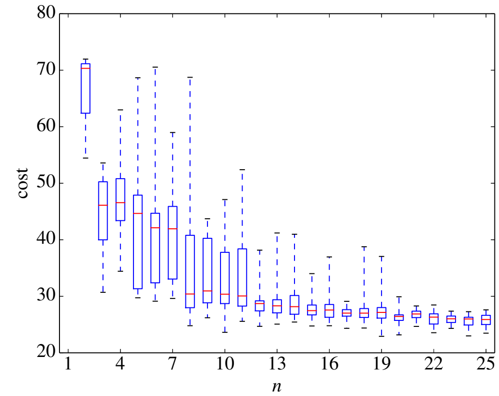

Fig. 6. Distribution of path costs without post-processing for benchmark 7 - Each box plot visualizes the distribution of costs for different parameter settings for our algorithm: $N_c = 10n$, $N_i = 100n$, $N_t = n$. The red line depicts the median. The first and third quartile are represented by the box, minimum and maximum by the whiskers.

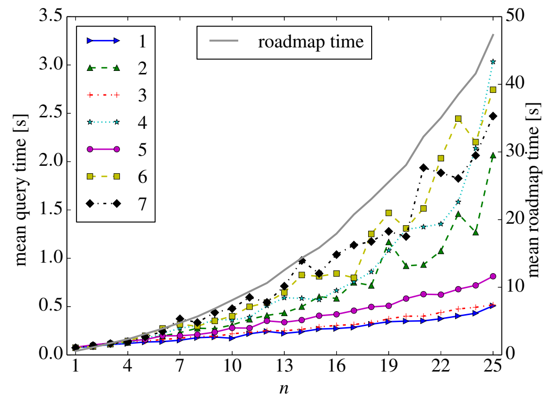

Fig. 7. Average query and roadmap times without post-processing for the benchmark tasks - Seven lines visualize the average query times for the benchmarks and one the average roadmap times with different parameter settings: $N_c = 10n$, $N_i = 100n$, $N_t = n$.

of our planner. This second observation shows, that our planner has not fully converged to an optimal path even at its highest settings. This result is not surprising, as 2500 nodes for the PRM* cannot be expected to sufficiently explore the 7-dimensional configuration space of the robot. To visualize the distribution of solution costs of both planners at all seven benchmarks, Fig. 8 shows the corresponding box plots. As can be seen, RMR* produces solutions of higher quality with much lower variance in solution cost.

Finally, we implemented our approach on a real robotic work-cell. The manipulation sequence shown in Fig. 1 at the beginning of this paper is a solution path of our planner.

TABLE II

AVERAGE COST AND STANDARD ERROR OF THE MEAN

	Benchmark						
	1	2	3	4	5	6	7
PTR	**15.2**	**64.3**	**19.4**	**78.1**	**15.5**	**53.8**	**70.2**
	[1.14]	[3.64]	[1.86]	[4.17]	[1.04]	[3.12]	[3.80]
PTR	**14.6**	**56.6**	**17.4**	**69.7**	**14.9**	**49.3**	**63.6**
post-p.	[1.07]	[3.16]	[1.65]	[3.48]	[1.04]	[3.23]	[3.54]
RMR*	**9.8**	**23.8**	**12.1**	**25.3**	**10.6**	**24.3**	**25.8**
	[0.07]	[0.18]	[0.07]	[0.23]	[0.10]	[0.23]	[0.20]
RMR*	**8.6**	**20.3**	**10.6**	**21.1**	**9.1**	**20.6**	**21.5**
post-p.	[0.04]	[0.11]	[0.06]	[0.22]	[0.07]	[0.21]	[0.14]

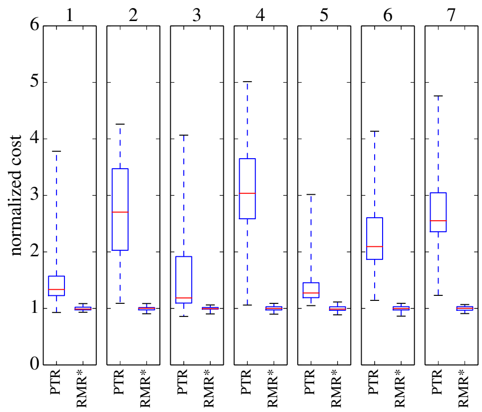

Fig. 8. Comparison of PTR and RMR* - Each box plot depicts the distribution of normalized cost without post-processing for a different benchmark. Costs are normalized to the average cost returned by our planner. RMR* is run with $N_c = 250$, $N_i = 2500$, $N_t = 25$. The red line depicts the median. The first and third quartile are represented by the box, minimum and maximum by the whiskers.

VII. Conclusion

This paper presented an asymptotically optimal manipulation planner. We established convergence under a set of new robustness conditions and validated the practicality of our approach in extensive simulations and on a real industrial manipulator.

The proposed planner is capable of returning high quality solutions to complex manipulation tasks in less than a second. This is achieved without relying on problem specific heuristics or simplifications as our algorithm directly works with the primitive operations common to manipulation.

Currently, the scope of our approach is limited to prehensile manipulation of one single object. Promising areas for future research include extending the approach to new problem domains, as well as methods to increase the speed of convergence for larger problems.

References

[1] R. Alami, T. Simeon, and J.-P. Laumond, “A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps,” in *The fifth international symposium on Robotics research*. MIT Press, 1990, pp. 453–463.

[2] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.

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

[4] T. Simeon, J. Cortes, A. Sahbani, and J.-P. Laumond, “A manipulation planner for pick and place operations under continuous grasps and placements,” in *IEEE International Conference on Robotics and Automation, 2002. Proceedings.*, vol. 2. IEEE, 2002, pp. 2022–2027.

[5] K. Hauser and J.-C. Latombe, “Multi-modal motion planning in non-expansive spaces,” *The International Journal of Robotics Research*, 2009.

[6] K. Hauser, “Task planning with continuous actions and nondeterministic motion planning queries,” in *Proc. of AAAI Workshop on Bridging the Gap between Task and Motion Planning*, 2010.

[7] K. Hauser and V. Ng-Thow-Hing, “Randomized multi-modal motion planning for a humanoid robot manipulation task,” *The International Journal of Robotics Research*, vol. 30, no. 6, pp. 678–698, 2011.

[8] S. Cambon, F. Gravot, and R. Alami, “Overview of asymov: Integrating motion, manipulation and task planning,” in *Intl. Conf. on Automated Planning and Scheduling Doctoral Consortium*, 2003.

[9] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, “Semantic attachments for domain-independent planning systems,” in *Towards service robots for everyday environments*. Springer, 2012, pp. 99–115.

[10] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, “Pddl-the planning domain definition language,” 1998.

[11] 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, pp. 179–195.

[12] J. Hoffmann, “Ff: The fast-forward planning system,” *AI magazine*, vol. 22, no. 3, p. 57, 2001.

[13] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, “Backward-forward search for manipulation planning,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015*. IEEE, 2015, pp. 6366–6373.

[14] K. Harada, T. Tsuji, and J.-P. Laumond, “A manipulation motion planner for dual-arm industrial manipulators,” in *IEEE International Conference on Robotics and Automation, 2014*. IEEE, 2014, pp. 928–934.

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

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

[17] K. Hauser, “Lazy collision checking in asymptotically-optimal motion planning,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2951–2957.

[18] S. Karaman and E. Frazzoli, “Optimal kinodynamic motion planning using incremental sampling-based methods,” in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 7681–7687.

[19] L. Jaillet and J. M. Porta, “Asymptotically-optimal path planning on manifolds,” *Robotics: Science and Systems VIII*, p. 145, 2013.

[20] W. Vega-Brown and N. Roy, “Asymptotically optimal planning under piecewise-analytic constraints,” *The 12th International Workshop on the Algorithmic Foundations of Robotics*, 2016.

[21] K. Mehlhorn and P. Sanders, *Algorithms and data structures: The basic toolbox*. Springer Science & Business Media, 2008.

[22] R. Bohlin and L. E. Kavraki, “Path planning using lazy prm,” in *IEEE International Conference on Robotics and Automation, 2000. Proceedings.*, vol. 1. IEEE, 2000, pp. 521–528.

[23] D. Hsu, J.-C. Latombe, and H. Kurniawati, “On the probabilistic foundations of probabilistic roadmap planning,” *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[24] R. Geraerts and M. H. Overmars, “Creating high-quality paths for motion planning,” *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.

[25] J. Pan, S. Chitta, and D. Manocha, “Fcl: A general purpose library for collision and proximity queries,” in *IEEE International Conference on Robotics and Automation, 2012*. IEEE, 2012, pp. 3859–3866.

[26] I. A. Sucan and S. Chitta. Moveit! [Online]. Available: http://moveit.ros.org

[27] C. Silpa-Anan and R. Hartley, “Optimised kd-trees for fast image descriptor matching,” in *IEEE Conference on Computer Vision and Pattern Recognition, 2008.* IEEE, 2008, pp. 1–8.

[28] M. Muja and D. G. Lowe, “Scalable nearest neighbor algorithms for high dimensional data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[29] L. Dagum and R. Menon, “Openmp: an industry standard api for shared-memory programming,” *IEEE computational science and engineering*, vol. 5, no. 1, pp. 46–55, 1998.