Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA'07), April, 2007.

Manipulation Planning Among Movable Obstacles

Mike Stilman[†] Jan-Ullrich Schamburek[†‡] James Kuffner[†] Tamim Asfour[‡]

[†]*The Robotics Institute*
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA, 15213, USA
robot@cmu.edu, kuffner@cs.cmu.edu

[‡]*Institute of Computer Science and Engineering*
University of Karlsruhe (TH)
Haid-und-Neu-Straße 7, 76131 Karlsruhe, Germany
jan@schamburek.de, asfour@ira.uka.de

***Abstract*— This paper presents the ResolveSpatialConstraints (RSC) algorithm for manipulation planning in a domain with movable obstacles. Empirically we show that our algorithm quickly generates plans for simulated articulated robots in a highly nonlinear search space of exponential dimension. RSC is a reverse-time search that samples future robot actions and constrains the space of prior object displacements. To optimize the efficiency of RSC, we identify methods for sampling object surfaces and generating connecting paths between grasps and placements. In addition to experimental analysis of RSC, this paper looks into object placements and task-space motion constraints among other unique features of the three dimensional manipulation planning domain.**

I. Introduction

While extracting disaster victims from rubble or looking for tools in a mechanic's workshop, autonomous robots will be required to remove obstacles in order to perform manipulation tasks. In this paper we present an efficient algorithm that reasons about the motion space of articulated robots and constructs plans for obstacle manipulation.

II. Related Work

The proposed domain is a generalization of Navigation Among Movable Obstacles (NAMO). In addition to traversing the space the robot must also manipulate a desired object to its goal configuration. NAMO is known to be NP-hard and currently intractable for brute force planners [1], [2]. The first practical planner in this domain, [3], follows a hierarchical greedy strategy that leads to local minima. [2] and [4] discuss graph-based methods for detecting and removing objects that directly block robot motion. Most recently, [5] and [6] introduced techniques for removing *indirectly* blocking objects. [6] employs a probabilistic approach that incrementally advances search by increasing the probability of displacing colliding objects. [5] uses a reverse planning method that directly computes the volume of space that is necessary for future object motions.

Existing solutions to NAMO have focused on planar examples with mobile robots. These tasks do not require the planner to choose object placements or handle articulated kinematics. We will extend [5] to 3D manipulation by an articulated robot. Our work is closely related to *assembly planning*, [7], [8]. However, assembly planners focus on separating a collection of parts and typically ignore the robot/manipulator. Domain operators also allow unassembled

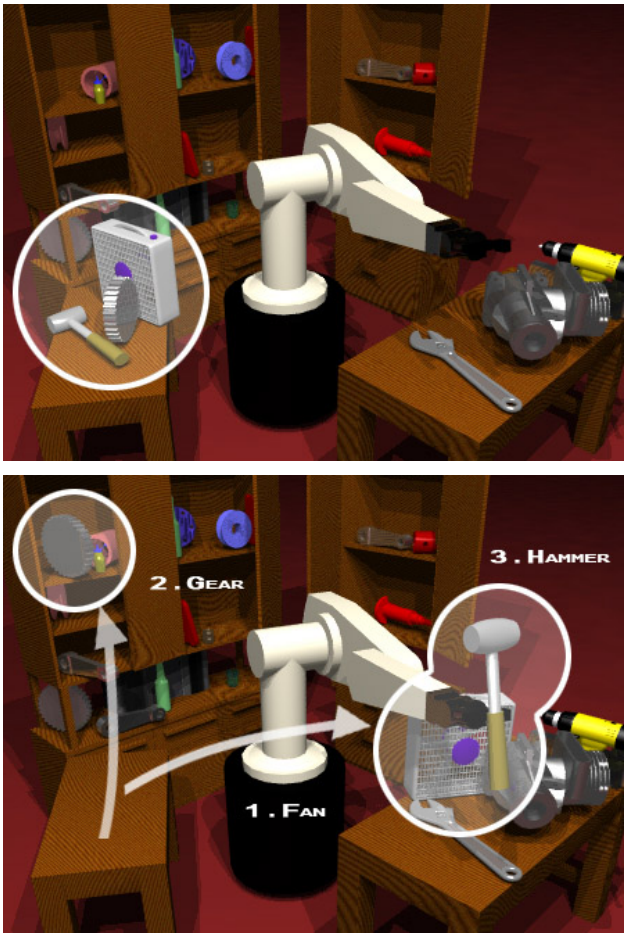


Fig. 1. The robot displaces objects to make space for a desired manipulation of the target. In this simulated experiment the robot autonomously plans to move the fan and the gear prior retrieving the hammer.

parts to be removed to "infinity." *Rearrangement planning* includes the robot, but specifies the final configurations of all objects [9], [10]. Similarly, high dimensional *manipulation planners* do not address the interactions between multiple movable objects described in Section IV [11], [12], [13]. We assign a single manipulation task and ask the robot to detect and resolve the displacements of interfering objects.

Our approach of placing constraints on past motions by analysis of future tasks is similar to backward chaining, [14]. Rather than explicit pre-image computation, we sample future paths. Using these paths as constraints for motions is related to [15]. In contrast to assuming a priority on object

motions, we search the space of object choices and orders. To generate the sample paths we apply multi-goal RRT-Connect, a rapid motion planner described in [16], [17].

III. Problem Statement

Our domain contains a robot manipulator with n degrees of freedom, a set of rigid body static obstacles $\mathbf{O}_F = \{F_1, \ldots, F_f\}$ and a set of rigid movable objects $\mathbf{O}_M = \{O_1, \ldots, O_m\}$. Each movable object is associated with the following:

Geometry	One closed triangular mesh.
Center of Gravity	A given point.
Motion Constraints	One vector specifying allowable motion in generalized coordinates.
Grasps	A set of workspace transforms for the end effector in the local object frame.

Each object has a workspace configuration q consisting of a translation and orientation. The robot configuration r is defined in joint space. We are given the initial configuration of the robot and all movable obstacles: $W^0 = (0, r^0, q_1^0, q_2^0, \ldots, q_m^0)$ and a final configuration q_G^{goal} for movable obstacle O_G. The robot must construct a sequence of joint paths that result in the desired object placement.

A. Operators

The desired sequence of paths can be interpreted as an iteration of operators: $Navigate : N(\tau)$ and $Manipulate : M(\tau, O_i)$, also referred to as $Transit$ and $Transfer$ in [11], [5]. The former moves only the robot, while the latter also displaces a single rigidly grasped object. Each operator is parameterized by a path in the configuration space of the robot: $\tau : [0, 1] \rightarrow r$, where $\tau(r_i, r_j)$ is some path from r_i to r_j. Let $\tau(s)$ be a configuration along the path.

The operators map $W^t = (t, r^t, q_1^t, q_2^t, \ldots, q_m^t)$ to W^{t+1}, where $q^{t+1} = q^t$ for all unaffected objects. The operators are subject to constraints: $N(\tau(r^t, r^{t+1}))$ is valid when the robot in any configuration $\tau(s)$ does not collide with an element of $\mathbf{O}_F$ or $\mathbf{O}_M$ in W^t or itself.

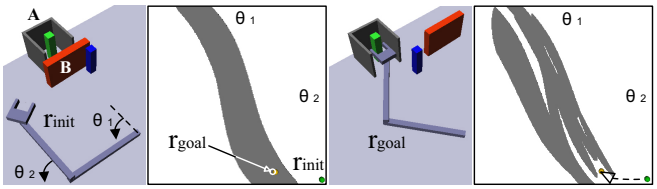
To define a valid $Manipulation$, let $K(r^t)$ be the workspace end effector pose as found by direct kinematics. $T_{q_i^t}^{K(r^t)}$ is a relative transform from the end effector pose to the object configuration/pose at the start of the action. Due to rigid grasping the object path is defined by the robot path:

$$\tau_{O_i}(s) = T_{q_i^t}^{K(r^t)} K(\tau(s)) \tag{1}$$

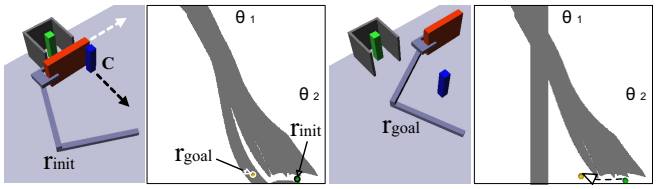
$M(\tau(r^t, r^{t+1}), O_i)$ requires three conditions: First, the relative transform $T_{q_i^t}^{K(r^t)}$ must be a valid grasp. Second, any robot configuration $\tau(s)$ and corresponding object configuration $\tau_{O_i}(s)$ must be collision free with respect to unaffected objects in W^t and self-collision. Lastly, the final object configuration $\tau_{O_i}(1)$ must be a statically stable placement of the object.

B. Simplifying Assumptions

For the purposes of computational efficiency and algorithmic clarity we present methods that are applicable to a subset of domain problems. We first assume that the problem



(a) The $\mathcal{C}$-space of the first two robot links before and after box B is moved.



(b) The $\mathcal{C}$-space of the robot grasping B before and after C is moved. For visualization, only translations of B are permitted.

Fig. 2. Visualization of the workspace and configuration space ($\mathcal{C}$-space) of a planar robot are computed in our simulator.

is *monotone* or that if a solution exists, it can be found by moving each obstacle once. Consequently we need not consider plans longer than the number of movable obstacles.

We further constrain object placements to variations of initial configurations that alter position and rotation about z. The objects must be placed on a flat surface to establish planar contact under the COG. While movable objects may be stacked initially, they can only be placed on fixed objects by the robot. These are sufficient but not necessary conditions for static stability. They allow us to pre-sample the space for possible placements without regard for object shape or the displacement of potential surfaces.

IV. Challenges

Despite the assumptions in Section III-B the computational complexity of our domain remains exponential. Suppose that there are p sampled placements for objects. At each branch point t, a forward planner has chosen t displacements and must select from $m - t$ objects and p placements for each object. If we let e be the time to verify the existence of $Navigate$ and $Manipulate$ paths, the overall complexity is $O(m!(pe)^m)$, as estimated in [5]. Searching the configuration space of an articulated manipulator, e, is expensive, especially in the case of redundant joints [18]. Exponential repetition of the search is currently intractable.

Clearly, not all displacements are equally useful. Perhaps we can reason about *spatial connectivity* to identify which manipulation actions help the robot reach the goal [2]. In Fig. 2(a), displacing object B deforms the configuration space, ($\mathcal{C}$-space), of the robot, creating free space for a valid $Navigation$ to grasp object A. Analogously, in Fig. 2(b), displacing object C makes the $Manipulation$ of object B possible. Constructive subgoals take the form of creating free space for some path in the robot $\mathcal{C}$-space or the joint $\mathcal{C}$-space of the robot and a grasped object.

However, even translations of cubes create complex nonlinear deformations in the configuration space. We have no analytical means for representing the mapping between object displacements and their effect on $\mathcal{C}$-space. Furthermore, while an object displacement may permit an immediate

$Manipulation$, it could block a future path for the robot. Explicitly representing all possible displacements of each object and its effect on the $\mathcal{C}$-space of all other objects would be exponentially expensive in memory and time [11].

V. Algorithm

Rather than explicitly planning to establish *spatial connectivity* as described in Section IV, we propose to sample the space of future paths. We recursively use these samples to construct paths for blocking objects.

The last step of the plan is always to $Manipulate\ O_G$ to its goal configuration along some path τ. If no objects are moved from their initial configurations, $Manipulate(\tau, O_G)$ would collide with a set of objects: $\mathbf{O}_{PAST}$. Our planner identifies these objects and plans to displace them. Since these displacements may also be blocked, the set $\mathbf{O}_{PAST}$ is expanded to include indirectly blocking objects.

Observe that while there may be infinite possible paths for object displacement, there is a finite number of object orderings. For a particular choice of paths, the number of objects that must be moved is relatively small. Hence we can search the space of orderings while incrementally selecting object placements and paths that are valid with regard to future $Navigation$ and $Manipulation$.

To formalize the search we let $\mathbb{C}_V$ be the volume of space that must be unoccupied to validate future operators. After sampling $Manipulate(\tau, O_G)$, $\mathbb{C}_V$ contains the workspace volume that is occupied by the robot and O_G during the continuous execution of τ. All objects that collide with $\mathbb{C}_V$ are placed in $\mathbf{O}_{PAST}$ and must be displaced to configurations that do not collide with $\mathbb{C}_V$. Recursively, paths that $Manipulate$ these objects and $Navigate$ to the subsequent subgoals define volumes of workspace that are added to $\mathbb{C}_V$ to constrain the placement of prior obstacles.

Our algorithm ResolveSpatialConstraints(RSC) is detailed in Fig. 4. RSC follows Fig. 3 in a depth first search over orderings of the blocking obstacles. Each node in the diagram represents a sampled $Manipulation\ of$ some object and $Navigation\ from$ grasping the object in its final configuration to grasping the subsequent object in its initial configuration. The children of the node are choices for the directly preceding object displacement.

The algorithm is initialized by specifying the goal object, O_G, and a goal configuration for the robot, r^{t+2}. The first call to RSC specifies these two parameters and empty sets for O_{PAST}, O_{FUT} and $\mathbb{C}_V$. The planner terminates a branch when PlanGrasp, PlanManipulation or PlanNavigation are unsuccessful. It also backtracks when all the orderings represented by a node's children terminate unsuccessfully.

VI. Motion Sampling

In order to apply the RSC algorithm we are required to choose a method for sampling the placement space of objects and the space of robot paths for $Navigation$ and $Manipulation$. Operator paths are generated using the rapid RRT-Connect algorithm. Placements are drawn from a uniform distribution to avoid bias.

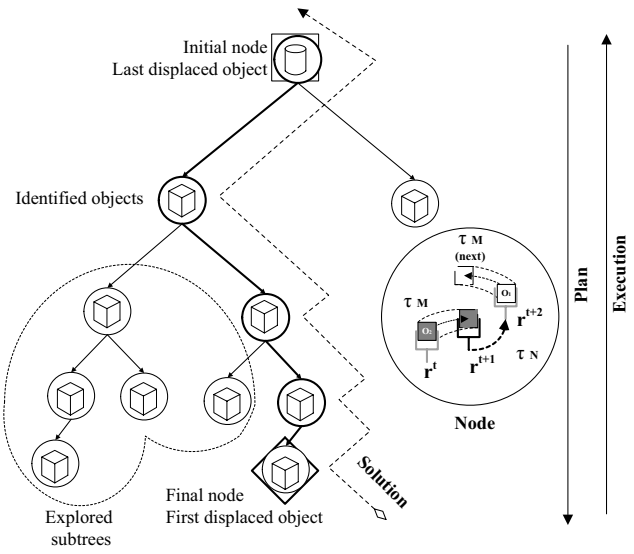


Fig. 3. Construction of a plan by searching object orderings and selective sampling of paths. Object displacements are planned in reverse order from execution.

ResolveSpatialConstraints$(O_c, \mathbf{O}_{PAST}, \mathbf{O}_{FUT}, r^{t+2}, \mathbb{C}_V)$
1 $(r^t, \text{grasp}) \leftarrow$ PlanGrasp(O_c)
2 $\mathbf{P} \leftarrow$ FindPlacements$(O_c, \text{grasp}, \mathbf{O}_{FUT}, \mathbb{C}_V)$
3 $M(\tau_M, O_c) \leftarrow$ PlanManipulation$(O_c, r^t, \mathbf{P}, \mathbf{O}_{FUT})$
4 $\mathbf{O}_{FUT} \leftarrow \mathbf{O}_{FUT.}$ **append** O_c
5 $N(\tau_N) \leftarrow$ PlanNavigation$(\tau_M(1), r^{t+2}, \mathbf{O}_{FUT})$
6 $\mathbb{C}_V \leftarrow \mathbb{C}_V$**append** SweptVolume$(\tau_M, \tau_N)$
7 $\mathbf{O}_{PAST} \leftarrow \mathbf{O}_{PAST}$ **append** IdentifyBlocking(τ_M, τ_N)
8 **if** $\mathbf{O}_{PAST} = \emptyset$
9 **then return** $\diamond$
10 **for each** O_P **in** $\mathbf{O}_{PAST}$
11 **do if** ResolveSpatialConstraints
12 $(O_P, \mathbf{O}_{PAST} - O_P, \mathbf{O}_{FUT}, r^t, \mathbb{C}_V)$
13 **then return** $\diamond$
14 **return** NIL

Fig. 4. Pseudo-code for the RSC algorithm. The algorithm retunrs NIL and backtracks if any of the three Plan operations fail (Lines 1,3 and 5).

A. Sampling Paths

Any node in the RSC tree shown in Fig. 3 corresponds to the displacement of some object O_c. The displacement can be defined by three reference configurations for the robot: r^t, r^{t+1} and r^{t+2}. r^{t+2} is the grasping configuration for the parent object O_p. r^t and r^{t+1} are grasping configurations for O_c in its initial and final configurations respectively. Since O_p is scheduled to be displaced immediately after O_c, r^{t+2} is known. PlanGrasp, PlanManipulation and PlanNavigation are three calls to a motion planning algorithm that are used to select r^t,r^{t+1} and sample paths $\tau_M(r^t, r^{t+1})$ and $\tau_N(r^{t+1}, r^{t+2})$ that connect them.

Since we are interested in changing and changeable environments, we apply a single query planner to construct sample paths. Namely, the RRT-Connect algorithm has been experimentally validated for rapid planning in high-dimensional spaces [16]. Furthermore, since objects can have

(a) (b)

Fig. 5. Randomly sampled placements autonomously selected for the drill. (a) shows collision free placements. (b) is the subset that has valid grasps.

numerous grasps, placements and inverse kinematics solutions, we apply multi-goal RRT-Connect whenever possible [17]. This planner grows random configuration space trees from each of the goals and allows the start tree to be connected to any of them.

We now describe the three motion plans in a step of RSC. For all plans the planner is not allowed to collide with static obstacles or obstacles in $\mathbf{O}_{FUT}$, whose initial configuration remains static until after O_c is displaced.

- PLANGRASP decides r^t, the grasp of O_c: This is a multi-goal plan from r^0, the initial configuration, to any robot configuration that grasps O_c. This step ensures that r^t can be reached by the robot from the initial state given the static objects.

- PLANMANIPULATION decides r^{t+1}: This multi-goal plan is given the starting configuration r^t and the associated GRASP of O_c. Its goal is to connect r^t to any valid configuration in the set pre-selected by FINDPLACEMENTS (Section VI-B). The returned plan is the path τ_M.

- PLANNAVIGATION - The final path planner is a single goal RRT-Connect that identifies τ_N. The planner searches from the final grasp of O_c: $\tau_M(1) = r^{t+1}$ to the grasp of the next object r^{t+2}.

Since RRT trees grow indefinitely when a solution is not found, we limit the depth of expansion. RSC backtracks when any of the three plans are unsuccessful.

Observe that in each call to PLAN, we accept the first successful connection as the path sample. Therefore, the RRT tends to connect objects to nearby goal configurations. This choice is reasonable but not exhaustive. When all obstacle orderings have been attempted, one should consider choosing different path samples. We propose two methods that avoid using the same paths and placements in subsequent trials: removing goals from the multi-goal RRT and biasing RRT search away from previous solutions.

B. Sampling Placements

The set of placements consists of random points on triangles, $\mathcal{T}$, with upward facing normals, $\mathbf{n}(\mathcal{T}_j)$. It is important to select placements uniformly from the entire surface area since the areas of $\mathcal{T}_j$ may differ. The area of the sample space, S, is a sum of bounding box areas for the triangles $\mathbf{b}(\mathcal{T})$. For a conditional impulse δ, we have:

$$|S| = \sum_{O_i \in \mathbf{O}_F} \sum_{\mathcal{T}_j \in O_i} \mathbf{area}(\mathbf{b}(\mathcal{T}_j))\delta_{(\mathbf{n}(\mathcal{T}_j)=[0\ 0\ 1]^T)} \quad (2)$$

We rejection sample a placement by randomly choosing a triangle according to its boxed contribution to S and selecting a point from a uniform distribution $U(S)$ over $\mathbf{b}(\mathcal{T})$. The probability distribution of a point p is:

$$P(p) = U(S) = \frac{\mathbf{area}(\mathbf{b}(\mathcal{T}(p)))U(\mathcal{T}(p))}{|S|} \quad (3)$$

Each sampled point decides the translation of the placement and a separate random draw is used for orientation around the z-axis. Since objects may only be placed on static surfaces, we initialize the algorithm by sampling a set of potential object placements.

The call to FINDPLACEMENTS$(O_c, \text{GRASP}, \mathbf{O}_{FUT}, \mathbb{C}_V)$ positions O_c at each of the sampled placements subject to three constraints. First, the object may not be in collision with a static object or a swept volume of a future $Navigation$ or $Manipulation$ path. Second, there must be a valid inverse kinematics solution, r^{t+1}, for the GRASP previously selected in PLANGRASP. Third, the robot in configuration r^{t+1} may not collide with fixed obstacles or elements of $\mathbf{O}_{FUT}$ which are fixed in their initial configurations due to the monotone assumption. The function returns a set of grasping configurations that satisfy these criteria.

VII. CONSTRAINTS

In addition to the standard collision detection that is required for motion planning, RSC search must handle other forms of constraints. One simple example of higher-order constraints is an ordering constraint on the motion of supporting objects. Obstacles that are initially supported by other movable objects must be moved prior to manipulating their support. Two of the more complex constraints are the placement constraints that result from reverse search and the motion constraints on real world objects.

A. Placement Constraints

The RSC algorithm is based on sampling future motions and using the resulting paths as constraints for previous motion. Namely, $\mathbb{C}_V$ is a swept volume of the space occupied by the future motions of the robot and displaced obstacles. This volume must be cleared of all objects for the sampled path to be valid. $\mathbb{C}_V$ serves two purposes: to detect objects that collide with sampled paths and to constrain the displacement of these objects. When searching for object placements, we ensure that the objects do not collide with the swept volumes.

In our current implementation, $\mathbb{C}_V$ is represented simply as a discrete sequence of robot and object models. This is reflected in the running time of placement search (Section VIII). When generating these constraints, future work should consider local convex hull approximations or occupancy grid methods to reduce the collision checking involved in placement selection. An important topic is to ensure the safety of the volumes. While overestimating the

Fig. 6. Still frames from our simulation environment. In this experiment an 8-DOF mobile manipulator operates the workshop. The robot is asked to deliver the drill. RSC autonomously identifies the door and the pulley as blocking objects and generates motion plans to remove them.

size of a volume may lead to the removal of some valid placements, underestimating it would mean that the planner constructs invalid paths.

B. Motion Constraints

The second type of constraint handled by our planner restricts motion of scene objects. Each object is associated with a transformation that serves as a reference frame for the task constraint. A boolean vector determines which degrees of freedom are can be changed for a valid displacement. For instance, cabinet doors only permit rotation about the z-axis.

To maintain the random sampling of paths, we considered modifications of the RRT algorithm for constrained objects. [19] shows that First-Order Retraction (FR-RRT) is both efficient and largely invariant to parameter choices. FR-RRT modifies the NEW_CONFIG method of the RRT planner. First, it computes the task error, $\Delta\mathbf{x}_{err}$, of each sampled configuration, $\mathbf{q}_s$, and the pseudo-inverse of the task Jacobian for the manipulator, $\mathbf{J}_t^\dagger$. Then it retracts $\mathbf{q}_s$ onto the constraint manifold by recursively applying Eq. 4.

$$\mathbf{q}_s' = \mathbf{q}_s - \mathbf{J}_t^\dagger \Delta\mathbf{x}_{err} \tag{4}$$

In our application, this planner found solutions to problems with constrained objects and had no effect on planning for unconstrained motion.

VIII. RESULTS

In order to test our approach, we performed simulated experiments with a three-link planar robot arm and an 8-DOF mobile manipulator based on the 6-DOF PUMA arm (Fig. 1,6,7). The planar environment shown in Fig. 7 consists of five rectangular movable objects and one fixed cubicle. The PUMA was placed in a simulated workshop environment with various shapes of tools and mechanical objects. This scene has 28 objects, 16 of which are movable. In the workshop some objects such as doors were constrained to only move along or about fixed axes.

For each scenario we performed 100 experiments. Table I summarizes the average total planning time in seconds, t_{total}, the average number of objects moved, $\overline{m}$, and individual times for each operation of RSC. $(t_{sv}, t_{place}, t_M, t_N, t_{grasp})$ correspond to the time spent creating collision models for swept volumes, identifying valid placements, sampling valid $Manipulate$, $Navigate$ and $Grasp$ paths.

Table II shows the average proportion of time spent on each subtask over all three experiments. RRT and $Place$ represent the cumulative cost for sampling paths and placements throughout planning. In both tables subtasks are grouped when their time measurements are independent.

Overall, we found that RSC consistently finds solutions to monotone problems in seconds of planning time. Due to the sampling strategy, the algorithm does not require significant overhead for high dimensional problems. One clear distinction is that for Fig. 7 sampling τ_N takes longer than τ_M. This contrasts Fig. 1, 6. Faster manipulation search is reasonable since τ_M has multiple feasible goals. In the detailed scenarios PLANMANIPULATION is more expensive due to collision and motion constraints on complex manipulated objects.

IX. CONCLUSION

In this paper we have introduced the RSC sampling-based reverse search algorithm for handling exponentially complex non-linear spaces. These spaces are common in manipulation due to the interactions between the robot, the movable objects and the environment. We have shown that even in such spaces there exist strategies for finding efficient solutions.

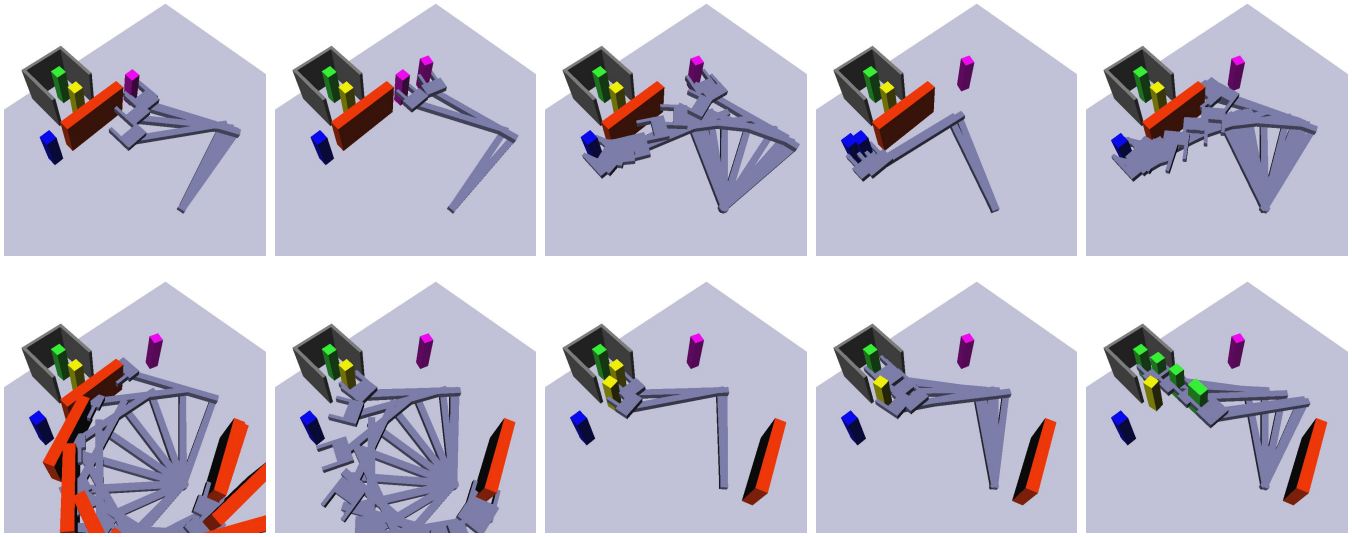


Fig. 7. A simulated example involving a planar robot arm. The goal is to retrieve the green object from the cubicle.

Exp.	$\overline{m}$	t_{total}	t_{rrt}	t_{sv}	t_{place}	t_M	t_N	t_{grasp}
Fig. 1	3.0	13.12	6.55	1.12	1.75	7.52	1.06	1.15
Fig. 6	3.8	13.26	4.56	3.16	0.55	5.92	1.30	1.69
Fig. 7	4.6	7.15	2.63	1.68	0.97	1.10	2.38	0.71

TABLE I

RSC: RUN TIMES FOR ALGORITHM COMPONENTS

Total	*RRT*	*SV*	*Place*	*M*	*N*	Grasp
100%	46.7%	22.9%	11.8%	44.9%	20.6%	12.5%

TABLE II

AVERAGE COST FOR EACH PROCEDURE OVER ALL EXPERIMENTS.

This domain leaves many interesting problems for future work. For instance, we have concentrated on monotone planning domains and were therefore able to terminate the ordering of objects. In some situations there may be interdependence between the motions of objects that will require moving them one after another.

While we have presented simple motion sampling strategies by using existing rapid configuration space planners there may be alternatives for representing the space of paths. The same question applies to the space of placements. Our placement choices are dictated by their accessibility to the motion planner. In the future it will be valuable to investigate other metrics for placement, both in terms of stability and utility for the complete motion plan.

X. ACKNOWLEDGEMENT

We thank the InterACT program for sponsoring this joint research. The work described in this paper was partially conducted within the German Humanoid Research project (SFB 588) funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft).

REFERENCES

[1] G. Wilfong. Motion panning in the presence of movable obstacles. In *Proc. ACM Symp. Computat. Geometry*, pages 279–288, 1988.

[2] M. Stilman and J.J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *Proc. IEEE Int. Conf. on Humanoid Robotics (Humanoids'04)*, 2004.

[3] P.C.Chen and Y.K.Hwang. Pracitcal path planning among movable obstacles. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 444–449, 1991.

[4] K. Okada, A. Haneda, H. Nakai, M. Inaba, and H. Inoue. Environment manipulation planner for humaonid robots using task graph that generates action sequence. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, pages 1174–1179, 2004.

[5] M. Stilman and J. Kuffner. Planning among movable obstacles with artificial constraints. In *Workshop on the Algorithmic Foundations of Robotics*, 2006.

[6] D. Nieuwenhuisen, A.F. van der Stappen, and M.H. Overmars. An effective framework for path planning amidst movable obstacles. In *Workshop on the Algorithmic Foundations of Robotics*, 2006.

[7] M. H. Goldwasser and R. Motwani. Complexity measures for assembly sequences. *Int. J. of Computational Geometry and Applications*, 9:371–418, 1999.

[8] R. Wilson. *On Geometric Assembly Planning*. PhD thesis, Department of Computer Science, Stanford University, 1992.

[9] O. Ben-Shahar and E. Rivlin. Practical pushing planning for rearrangement tasks. *IEEE Trans. on Robotics and Automation*, 14(4):549–565, 1998.

[10] J. Ota. Rearrangement of multiple movable objects. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1962–1967, 2004.

[11] R. Alami, J.P. Laumond, and T. Sim'eon. Two manipulation planning algorithms. In *Workshop on the Algorithmic Foundations of Robotics*, 1994.

[12] K. Gupta J. Ahuactzin and E. Mazer. Maniopulation planning for redundant robots: A practical approach. *International Journal of Robotics Research*, 17(7), 1998.

[13] T. Simeon, J. Cortes, A. Sahbani, and J.P. Laumond. A manipulation planner for pick and place operations under continuous grasps and placements. In *Proc. IEEE Int. Conf. on Robotics and Automation,*, 2002.

[14] Tomas Lozano-Perez, Matthew Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1), 1984.

[15] Michael Erdmann and Tomas Lozano-Perez. On multiple moving objects. In *IEEE Int. Conf. on Robotics and Automation*, pages 1419–1424, Apr. 7-10 1986.

[16] J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Workshop on the Algorithmic Foundations of Robotics*, pages 995–1001.

[17] Y. Hirano, K. Kitahama, and S. Yoshizawa. Image-based object recognition and dexterous hand/arm motion planning using rrts for grasping in cluttered scene. pages 3981–3986, 2005.

[18] S.M. LaValle and J.J. Kuffner. Rapidly exploring random trees: Progress and prospects. In *Workshop on the Algorithmic Foundations of Robotics*, 2000.

[19] M. Stilman. Task constrained motion planning in robot joint space. Technical Report CMU-RI-TR-06-43, Carnegie Mellon University: The Robotics Institute, Sept 2006.