Parts Assembly Planning under Uncertainty with Simulation-Aided Physical Reasoning

Sung-Kyun Kim, Maxim Likhachev
The Robotics Institute
Carnegie Mellon University
{kimsk,maxim}@cs.cmu.edu

Abstract— **Parts assembly, in a broad sense, is to make multiple objects to be in specific relative poses in contact with each other. One of the major reasons that make it difficult is uncertainty. Because parts assembly involves physical contact between objects, it requires higher precision than other manipulation tasks like collision avoidance. The key idea of this paper is to use simulation-aided physical reasoning while planning with the goal of finding a robust motion plan for parts assembly. Specifically, in the proposed approach, a) uncertainty between object poses is represented as a distribution of particles, b) the motion planner estimates the transition of particles for unit actions (motion primitives) through physics-based simulation, and c) the performance of the planner is sped up using Multi-Heuristic A* (MHA*) search that utilizes multiple inadmissible heuristics that lead to fast uncertainty reduction. To demonstrate the benefits of our framework, motion planning and physical robot experiments for several parts assembly tasks are provided.**

I. Introduction

Parts assembly is happening not only in factories but also in most of our living spaces. It is an operation of arranging, stacking, or combining objects. Many physical manipulation tasks that are to set relative poses between multiple objects in contact can be considered as parts assembly in a broad sense.

The challenge, however, is that contact involves complex physical phenomena such as friction and force interaction in addition to kinematics and dynamics of objects. These phenomena are hard to estimate or approximate well. Moreover, parts assembly usually requires higher precision in motion because it is a practice of placing objects to fit in, not separating them apart.

Therefore, it is important to reason about the underlying physics and to execute manipulation plans that are robust to uncertainty during assembly. In this paper, we propose a framework to utilize a physics-based simulator for physical reasoning and use graph search algorithms to find a robust motion plan in belief space.

The rationale behind selecting a simulator for physical reasoning is that modeling complex manipulation actions to sufficient degree of fidelity is infeasible. Instead, we should exploit state-of-the-art in physics-based simulation while planning.

This research was sponsored by ARL, under the Robotics CTA program grant W911NF-10-2-0016.

By sampling uncertainty distribution and running physics-based simulation of actions, the planner constructs a belief space. The planner then runs a heuristic search to find a plan while reducing the uncertainty for a higher chance of success.

This paper is organized as follows: Section II reviews the related work in parts assembly and planning under uncertainty. Section III explains some background about graph search and describes how a simulator is integrated with a motion planner for physical reasoning. Section IV provides the detailed formulation and algorithm of the motion planning, and section V shows experimental results for several parts assembly tasks. Section VI concludes the paper.

II. Related Work

Since parts assembly is an essential manipulation task, research about parts assembly has a long history. Some earlier works were about part feeding that utilized contact between an object and the environment to adjust its pose [1], [2]. Basically, these approaches tried to reduce uncertainty in object orientation on a plane by a sequence of actions. They required rigorous analysis of configuration space that particularly depended on the shape of the object and was only possible in 2D space.

More recent works inheriting this scheme can be found in [3] and [4]. In these works, the robot exploits contact with the environment or other objects for robust grasping or in-hand manipulation. (It is called *extrinsic dexterity* in [3].) As in this paper, they rely on the idea of using contact to reduce uncertainty and successfully demonstrate its effectiveness. However, the motions being used are hand-scripted and there is no general motion planner.

In [5], [6], works on parts assembly by multiple robots are presented. They propose an architecture for parts assembly and failure recovery. It is a high-level symbolic planner, so it treats the actual manipulation as a simple operation and does not account for physical phenomena during parts assembly.

On the other hand, there is an abundance of work on planning under uncertainty, An example is collision avoidance under uncertainty [7]. The other example is uncertainty reduction in localization by getting closer to landmarks or beacons [8], [9]. Both can also be pursued simultaneously [10], [11], [12]. In either case, they don't presume any

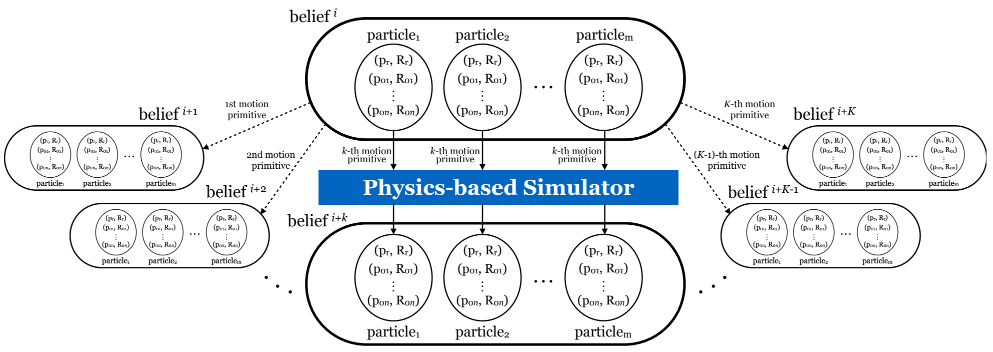


Fig. 1: Belief space representation in a graph and its expansion scheme. belief^i denotes a belief state $s^i = \mathbb{P}^i$, and particle$_j$ denotes a particle sub-state P_j. Each particle in a predecessor belief state is simulated for each motion primitive to get a successor belief state.

contact with the environment, and thus, precise physical simulation is not considered.

A similar approach to this paper can be found in [13], [14] that are about 2D pushing motion planning under uncertainty. They utilize a simulator for physical reasoning and RRT or A* variant search algorithms for motion planning. In their works, however, no interaction between multiple objects is considered and the search space is limited to 2D, which makes the problem simpler. In [14], tactile sensors of the robot gripper are used to observe the true state assuming direct contact with the object, but this approach can hardly be used in our problem because the gripper is indirectly contacting the environment.

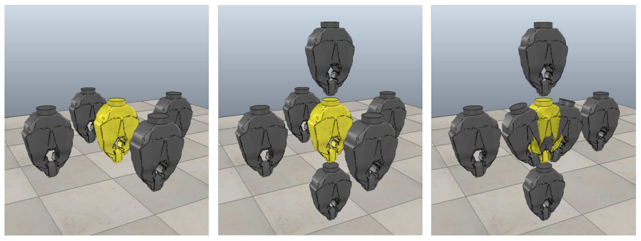
Other relevant works are in [15], [16]. They are using reinforcement learning techniques with deep networks for different parts assembly tasks. No physics-based simulator is used, but instead physical motion data of the robot is used for learning. It would be interesting for us to compare results with this approach in the near future.

III. Graph Construction with Physical Reasoning

A. Notation and Terminology

For a given graph search problem, let $\mathbb{S}$ denote the finite set of states of the search domain. $c(s, s')$ denotes the edge cost between state s and state s', and if they are not connected, $c(s, s') = \infty$. $\textsc{Succ}(s) := \{s' \in \mathbb{S} \mid c(s, s') \neq \infty\}$ denotes the set of all successors of s. Let $g(s)$ denote the current best path cost from s_{start} to s and $h(s)$ denote the heuristic for s which is an estimate of the best path cost from s to s_{goal}. A heuristic is *admissible* if it never overestimates the best path cost to s_{goal} and *consistent* if it satisfies $h(s_{goal}) = 0$ and $h(s) \leq h(s') + c(s, s')$ for any s, s' such that $s \neq s_{goal}$ and $s' \in \textsc{Succ}(s)$.

Let $\mathrm{H} = [h_0, h_1, ..., h_N]$ denote an ordered list of heuristics where h_0 is a consistent heuristic, while the other h_v for $v \in \{1, ..., N\}$ can possibly be an inadmissible heuristic. OPEN$_v$ denotes a priority queue ordered by a key value which is $f_v(s) = g(s) + h_v(s)$ or $f_v(s) = g(s) + w_1 * h_v(s)$ where $w_1 \geq 1$.



(a) 2D (x, y) (b) 3D (x, y, z) (c) 3D (y, z, R)

Fig. 2: An example of belief state representation: particles of cross-polytopes (hyperoctahedron) in different search spaces.

B. Graph Construct: State Lattice with Controllers (SLC)

In order to formulate motion planning as a graph search problem, we need to represent the search space as a graph. One can discretize the world into n-dimensional grid and take each cell as a node on a graph, but in this paper, we adopt a graph construct called *State Lattice with Controllers (SLC)* [17].

SLC is composed of a set of states $\mathbb{S}$ and their connecting edges $\mathbb{E}$. It adds more states and edges to the graph by computing the successor function $\textsc{Succ}(s)$ for $s \in \mathbb{S}$ that is already in the graph. In $\textsc{Succ}(s)$, the original State Lattice construct uses pre-computed metric motion primitives only, but SLC allows us to use controller-based motion primitives as well. This means that motion primitives can be simulated in a realistic controller and trigger setting to generate more plausible successor states. (A trigger is an event, such as detecting a marker or reaching the target position, that is used to terminate the simulation and return the result.) Based on this scheme, a more physically reasonable graph can be constructed for the given robot and the environment.

C. Belief State Representation

Since we assume there is uncertainty in the state of robot and objects, we need to encode uncertainty into the state of the graph. In this work, we represent a state of a graph s as a belief state which is a set of particle sub-states $\mathbb{P} = \{P_0, P_1, ..., P_m\}$ (see Fig. 1). Each particle P_j consists of the pose information of the robot and objects $\{(p_r, R_r), (p_{o1}, R_{o1}), ..., (p_{on}, R_{on})\}$.

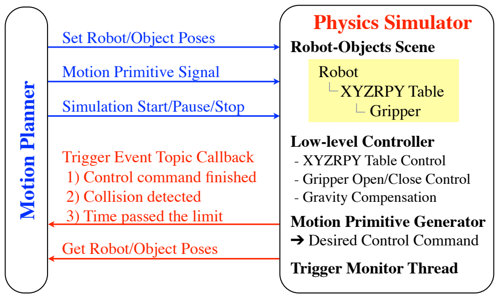


Fig. 3: Integration of a motion planner and a physics-based simulator. The simulator has all the information about the robot and objects, including geometric shapes, inertial properties, and controller characteristics. Simulation of a motion primitive is terminated if one of the trigger events is detected.

For parts assembly tasks, we accept Gaussian distribution as the initial uncertainty of a given task, but note that any distributions can be used as long as the number of particles are sufficient to represent the distribution.

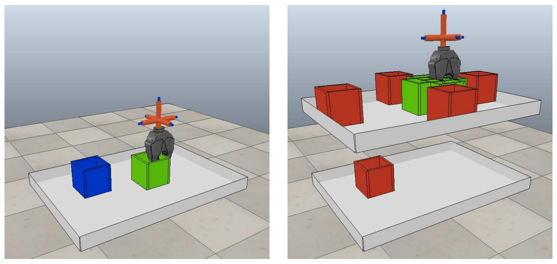
The typical way to draw particles from a given Gaussian distribution is random sampling. However, we cannot employ plenty of random samples to properly represent the original distribution because the simulation for each particle is computationally expensive. Alternatively, we use vertices of a uniform polytope as the particles since it can be seen as an approximation of c_p-sigma ellipsoid where c_p is a constant which is set to 2 in this work. Amongst uniform polytopes, we used a cross-polytope which is a higher dimensional octahedron for particle generation as shown in Fig. 2.

D. Belief State Transition via Simulation

In order to get physically reasonable $\textsc{Succ}(s)$ for a predecessor belief state, we exploit a physics-based simulator. More specifically, we use V-REP (Virtual Robot Experimentation Platform) simulator that supports Vortex Dynamics Engine [18], and integrated it with the motion planner as shown in Fig. 3.

As shown in Fig. 1, every particle sub-state in the predecessor belief state is simulated for each motion primitive, and the results constitute the successor belief states. Note that all the motion primitives being used are controller-based ones in the context of SLC, which means that a motion primitive is terminated by a trigger event in the simulation. Another note is that each motion primitive is to assign incremental displacement to the gripper, not the absolute pose in the global frame.

A motion primitive is triggered when 1) the control command is finished, 2) any change of the collision state is detected, or 3) the simulation time exceeded the limit. The first trigger is usually detected when the robot gripper remains in a free space without any collision until reaching the target pose. The second trigger is detected when the collision (or contact) states between any objects is switched from `NonContact` to `Contact`, or `Contact` to `NonContact`. The third trigger is for exception handling for the case that the simulator gets stuck.



(a) A given planning problem. (b) (x, y, u)-configuration space.

Fig. 4: Illustration of 2D box-on-table task. (a) The green box is the start pose, and the blue box is the goal pose. (b) The red boxes are the contact and goal attractors for inadmissible heuristics. Uncertainty u in the configuration space is visualized in z-axis. The green box on the upper level has high uncertainty and needs to get contact with the walls to reduce pose uncertainty and get down to the lower level of uncertainty where the goal state exists.

IV. Multi-Heuristic A* Search in a Foliated Belief Space

A. Uncertainty in Configuration Space

Before getting into the details for belief space search, we describe the special structure of the configuration space under uncertainty.

First, let us take an example. As shown in Fig. 4(a), the robot gripper is holding a box and wants to place it at the position of the blue box accurately. (This task will be called *box-on-table* hereafter.) However, there is uncertainty in the initial pose of the gripper and the box (depicted in a green color). How can we represent the amount of uncertainty of each belief state?

Let us define a particle-aggregated belief state vector X which describes the distribution of particles.

$$X = \begin{bmatrix} x \\ \omega \\ u \end{bmatrix} = \begin{bmatrix} \mathbb{E}_j[p_j^{tool}] \\ \mathbb{E}_j[\omega_j^{tool}] \\ u \end{bmatrix} \tag{1}$$

where j is the index for particles. p is a position vector, and $\omega = \theta\hat{\omega}$ is an orientation vector that can be obtained from angle-axis representation where θ and $\hat{\omega}$ are the rotation angle and rotation axis, respectively. Basically it is composed of mean vectors of position and orientation of the tool frame in addition to a scalar-valued uncertainty measure. (The tool frame is a local frame attached the object held by the gripper and, in this example, is at the center of the bottom of the box.) Note that position and orientation are with respect to the local frame of the table which is the target object.

In this work, uncertainty is defined as a weighted sum of traces of sample covariance matrices for position and orientation as follows:

$$u = w_p \mathrm{Tr}\left(\mathbb{E}_j\left[\left(p_j^{tool} - \mathbb{E}_j\left[p_j^{tool}\right]\right)\left(p_j^{tool} - \mathbb{E}_j\left[p_j^{tool}\right]\right)^T\right]\right) + w_o \mathrm{Tr}\left(\mathbb{E}_j\left[\left(\omega_j^{tool} - \mathbb{E}_j\left[\omega_j^{tool}\right]\right)\left(\omega_j^{tool} - \mathbb{E}_j\left[\omega_j^{tool}\right]\right)^T\right]\right) \tag{2}$$

where w_p and w_o are weights for position and orientation, respectively. Note that the trace of covariance matrix is used instead of the determinant due to its numerical unstability.

Now notice that we have a uncertainty measure coordinate in addition to pose coordinates in the belief state vector. It means that we have (d+1)-dimensional configuration space for d-dimensional planning problem under uncertainty. Figure 4(b) illustrates the (x, y, u)-configuration space for a planning problem in (x, y)-space.

One important physical fact to note is that the uncertainty of the relative pose between objects cannot be reduced without getting in contact with each other, provided that all the executable actions are to move around in the space. In the box-on-table example, a belief state on a high uncertainty level (the green box in Fig. 4(b)) cannot get down to the lower level of uncertainty without contact with the walls. In a word, this phenomenon leads to *foliation* of the configuration space.

B. Foliated Inadmissible Heuristic Function

As discussed in the previous section, the search configuration space is foliated, which means that transition between states with different uncertainty measures is mostly blocked by (virtual) obstacles except a few narrow passages, such as the walls in the box-on-table example.

In order to tackle this narrow-passage problem, we introduce a foliated inadmissible heuristic function as follows:

$$\begin{aligned} h(s) &= h\left(d_c(s), d_g(s), u(s)\right) \\ &= \begin{cases} w_d d_c(s) + w_u u(s) & (u(s) > u_{tol}) \\ w_d d_g(s) + w_u u(s) & (u(s) \leq u_{tol}) \end{cases} \end{aligned} \tag{3}$$

where w_d and w_u are weights for distance and uncertainty, respectively. d_c and d_g are Euclidean distances to the contact attractor state and the goal attractor state that are given parameters to the heuristic function. u is the uncertainty measure defined in (2), and u_{tol} is the goal tolerance for u. d_c and d_g are defined as follows:

$$\begin{aligned} &d_c(s) = d(s, s_{cont}) \\ &= \frac{1}{m}\sum_{j=1}^{m}\left(w_p\sqrt{(x_s - x_c)^T(x_s - x_c)} + w_o\Delta\theta(s, s_{cont})\right) \end{aligned} \tag{4}$$

$$\begin{aligned} &d_g(s) = d(s, s_{goal}) \\ &= \frac{1}{m}\sum_{j=1}^{m}\left(w_p\sqrt{(x_s - x_g)^T(x_s - x_g)} + w_o\Delta\theta(s, s_{goal})\right) \end{aligned} \tag{5}$$

where s_{cont} is the contact attractor state, and s_{goal} is the goal state. x is the position vector of a belief state defined in (1), and subscripts s, c and g stand for the current, contact attractor, and the goal attractor states, respectively. $\Delta\theta(s, x_{goal})$ is the rotation angle from the current state s to the goal state s_{goal}, which can be computed in angle-axis representation. w_p and w_o are weights for position and orientation, respectively.

As a high-level explanation, this foliated heuristic function is to lead the graph to expand toward the contact attractor state in high uncertainty region and toward the goal attractor state in low uncertainty region. For example, if all the states in OPEN of this heuristic are on the same high uncertainty level and have the same g-value, then the state nearest to the contact attractor will have the minimum h-value and f-value, and it will be selected as the predecessor for expansion.

Also note that w_d and w_u on the right hand side of (3) should be chosen carefully, so that the second term for the uncertainty is signficantly larger than the first term for the distance. It is because the transition in uncertainty coordinate is more difficult due to the foliation of the search space.

We define a consistent *anchor* heuristic function (to adopt MHA* framework which will be introduced in section IV-C) and an edge cost function as follows:

$$h_0(s^i) = h_0(s^i, s_{goal}) = w_d d(s^i, s_{goal}) + w_u u(s^i) \tag{6}$$

$$g(s^{i-1}, s^i) = w_d d(s^{i-1}, s^i) + w_u u(s^{i-1}) \tag{7}$$

where superscript i is the index to represent predecessor-successor relationship in the graph. $d(s^{i-1}, s^i)$ means the Euclidean distance between s^{i-1} and s. Then, we can define $f_0(s^i)$ for the anchor heuristic as follows:

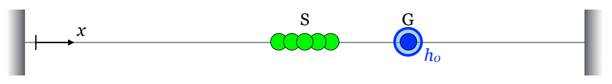
$$\begin{aligned} f_0(s^i) =& \sum_{t=1}^{i} g(s^{t-1}, s^t) + h_0(s^i, s_{goal}) \\ =& \sum_{t=1}^{i}\left(w_d d(s^{t-1}, s^t) + w_u u(s^{t-1})\right) \\ &+ w_d d(s^i, s_{goal}) + w_u u(s^i) \\ =& w_d\left(\sum_{t=1}^{i} d(s^{t-1}, s^t) + d(s^i, s_{goal})\right) \\ &+ w_u\left(\sum_{t=1}^{i} u(s^{t-1}) + u(s^i)\right) \end{aligned} \tag{8}$$

where s^0 is equivalent to s_{start}. Note that the term for uncertainty u in (7) is not the difference between predecessor and the successor states, but the remaining uncertainty of the parent state. This is because, in the former case, the second term in (8) will be the same for all paths that connect s_{start} and s_{goal}. On the other hand, the latter case can effectively penalize a path that remains in high uncertainty region for a long time.

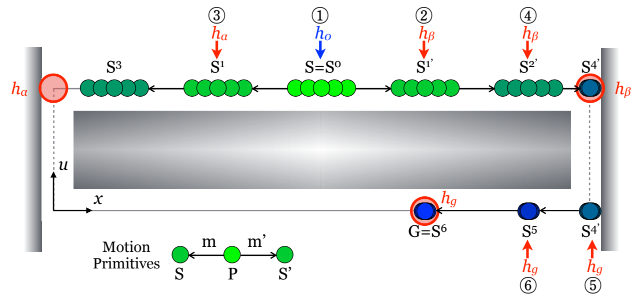
C. Graph Search Algorithm in a Foliated Belief Space

For the motion planning in the above-mentioned foliated belief space, Multi-Heuristic A* (MHA*) search algorithm [19], [20] is used. Refer to the pseudo code of the algorithm in Alg. 1.

MHA* has one consistent heuristic, which is called an anchor heuristic, and N arbitrary inadmissible heuristics of any necessity. This algorithm cycles through each inadmissible heuristics in a round-robin fashion, and determines whether to use the inadmissible heuristic or the anchor heuristic based on the condition in line 37 in Alg. 1. In such a way, it can control the sub-optimality of the solution by a factor



(a) A given problem: move balls at S to G. Consistent anchor heuristic (h_0).



(b) Inadmissible contact heuristics (h_α and h_β) and goal heuristic (h_g).

Fig. 5: Illustration of a 1-dimensional toy example.

of $w_1 * w_2$ where w_1 and w_2 are the weights for weighted A* search (in line 22) and anchor sub-optimality (in line 37).

To adopt MHA* for our problem, two major revisions are made: particle generation/transition and attractor-based heurisitics. As a belief space search problem, we need to generate a set of particles and obtain reasonable transition of them, which are explained in section III. For the foliated belief space, we need to find good (inadmissible) heuristics that can help the search to go through the narrow passages fast, which can be in the form of (3) in section IV-B.

The heuristic function in (3) needs two input parameters, s_{cont} and s_{goal}, and they are being searched in ATTRACTORSEARCH(s_{seed}) as shown in line 8 in Alg. 1. It is a quite simple operation that checks the amount of uncertainty reduction after applying motion primitives. It can apply a single long motion primitive or a sequence of them, and it can start from s_{start} or s_{goal}. As of now it is a naive process, but can possibly be developed as a sophisticated one.

D. Toy Example

Let us take a look at a 1-dimensional toy example to see how MHA* works in a foliated belief space. As shown in Fig. 5(a), the initial belief state at S has high uncertainty in x-position. From the belief state vector representation, we can construct a 2-dimensional configuration space with additional u-coordinate as shown in Fig. 5(b). There are two

TABLE I: MHA* SEARCH PROCESS FOR A TOY EXAMPLE

Index	Turn for Round-Robin	Satisfied Line 37?	Selector Heuristic	Selected Parent	Child States
1	α	No	h_0	$S^0 = S$	S^1, $S^{1'}$
2	β	Yes	h_β	$S^{1'}$	$S^{2'}$
3	α	Yes	h_α	S^1	S^3
4	β	Yes	h_β	$S^{2'}$	$S^{4'}$
5	α	Yes	h_g	$S^{4'}$	S^5
6	β	Yes	h_g	S^5	$S^6 = G$

Algorithm 1 Multi-Heuristic A* Search in a Foliated Belief Space

Input: The start state s_{start} and the goal state s_{goal}, sub-optimality bound factor w_1, w_2 (both ≥ 1), and one consistent heuristic h_0.

Output: A path from s_{start} to s_{goal} whose cost is within $w_1 * w_2 * g^*(s_{goal})$.

```
1:  procedure CROSSPOLYTOPEPARTICLES($\mu$, $\Sigma$)
2:      create a nominal particle $P_0$ from $\mu$
3:      $\mathbb{P} \leftarrow \emptyset$
4:      for $d \in$ SearchSpaceCoordinates do
5:          $\mathbb{P} \leftarrow \mathbb{P} \cup \{P_0 + c_p * \Sigma_{(d,d)}\hat{\mathbf{e}}_d\}$    ▷ $\{\hat{\mathbf{e}}_d\}$: orthonormal basis of SearchSpace
6:          $\mathbb{P} \leftarrow \mathbb{P} \cup \{P_0 - c_p * \Sigma_{(d,d)}\hat{\mathbf{e}}_d\}$
7:      return $\mathbb{P}$
8:  procedure ATTRACTORSEARCH($s_{seed}$)
9:      $\mathbb{S}_{attractor} \leftarrow \emptyset$
10:     for $m_k \in$ LongMotionPrimitives do
11:         $s' \leftarrow$ SUCC($s_{seed}, m_k$)
12:         if $s'$.UNCERT() $< c_u * s_{seed}$.UNCERT() then
13:             $\mathbb{S}_{attractor} \leftarrow \mathbb{S}_{attractor} \cup \{s'\}$
14:     return $\mathbb{S}_{attractor}$
15: procedure NEWINADMISSHEURISTIC($s_{attractor}$, $s_{goal}$)
16:     create a new instance of a heuristic class, $h'$
17:     $h'$.attractor $\leftarrow s_{attractor}$
18:     $h'$.goal $\leftarrow s_{goal}$
19:     return $h'$
20: procedure KEY($s$, $v$)
21:     $h_v \leftarrow$ H.GET($v$)
22:     return $g(s) + w_1 * h_v(s)$
23: procedure MAIN()
24:     $s_{start} \leftarrow$ CROSSPOLYTOPEPARTICLES($\mu_{start}$, $\Sigma_{start}$)
25:     H $\leftarrow \emptyset$, $N \leftarrow 0$
26:     H.ADD($h_0$)
27:     for $s_{attractor}$ in ATTRACTORSEARCH($s_{start}$) do
28:         H.ADD(NEWINADMISSHEURISTIC($s_{attractor}$, $s_{goal}$))
29:         $N \leftarrow N + 1$
30:     $g(s_{start}) \leftarrow 0$, $g(s_{goal}) \leftarrow \infty$
31:     for $v = 0, 1, ..., N$ do
32:         OPEN$_v$ $\leftarrow \emptyset$
33:         insert $s_{start}$ in OPEN$_v$ with KEY($s_{start}$, $v$)
34:     CLOSED$_{anchor}$ $\leftarrow \emptyset$, CLOSED$_{inad}$ $\leftarrow \emptyset$
35:     while OPEN$_0$.MINKEY() $< \infty$ do
36:         for $v = 1, 2, ..., N$ do
37:             if OPEN$_v$.MINKEY() $\leq$ $w_2$ * OPEN$_0$.MINKEY() then
38:                 if $g(s_{goal}) \leq$ OPEN$_v$.MINKEY() then
39:                     if $g(s_{goal}) < \infty$ then
40:                         terminate and return a solution path
41:                 else
42:                     $s \leftarrow$ OPEN$_v$.TOP()
43:                     EXPANDSTATE($s$)
44:                     insert $s$ in CLOSED$_{inad}$
45:             else
46:                 if $g(s_{goal}) \leq$ OPEN$_0$.MINKEY() then
47:                     if $g(s_{goal}) \leq \infty$ then
48:                         terminate and return a solution path
49:                 else
50:                     $s \leftarrow$ OPEN$_0$.TOP()
51:                     EXPANDSTATE($s$)
52:                     insert $s$ in CLOSED$_{anchor}$
```

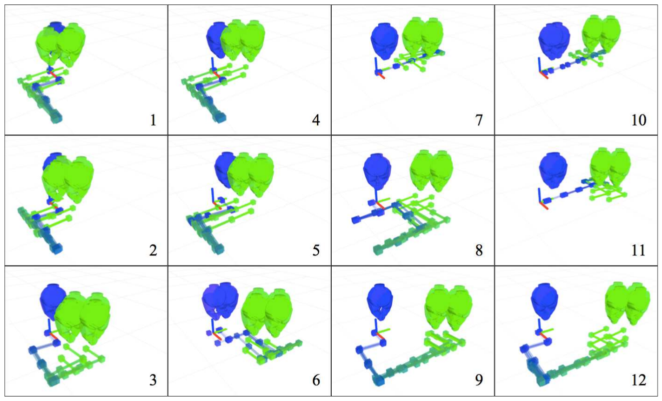


Fig. 6: Planning results for 2D box-on-table tasks with 12 different start poses. The green and blue markers are the start and final poses of robot gripper particles, respectively, and an RGB-colored frame marker is the goal pose for the object held by the robot.

TABLE II: Planning Results for 2D Box-on-Table Tasks
(The step size for translational motion primitives is 0.10 m, and the initial offset from the nominal start pose of each particle is 0.06 m.)

Task ID	Time [s]	Cost	Node # in Path / Expansion #	Pos. Error Mean [m]	Pos. Error Std Dev [m]
1	367	42032	9/10	0.0165	0.0085
2	482	46544	12/12	0.0143	0.0072
3	364	35812	8/10	0.0163	0.0067
4	484	45172	10/11	0.0158	0.0055
5	502	47644	13/13	0.0439	0.0066
6	396	47847	11/11	0.0273	0.0290
7	488	42900	13/14	0.0146	0.0059
8	685	48839	14/16	0.0296	0.0059
9	638	45951	12/16	0.0171	0.0063
10	463	41891	9/13	0.0382	0.0203
11	711	42198	16/19	0.0370	0.0099
12	637	47356	13/16	0.0162	0.0089
Mean	518.1	44515.5	11.6/13.4	0.0239	0.0101
Std Dev	121.2	3705.5	2.3/2.8	0.0108	0.0072

inadmissible heuristics, h_α and h_β, that have their contact attractors on the left and right walls, respectively, and the goal attractors at the goal state position.

As presented in Table I, MHA* algorithm starts to cycle each inadmissible heuristic for graph expansion. For the first iteration, h_α takes the turn but couldn't satisfy the condition of line 37 in Alg. 1. So, OPEN_0 of h_0 is used to select the state for expansion, and then S^0 is expanded. For the second run, h_β takes the turn and satisfied the condition, and $\text{S}^{1'}$ is expanded. In the fourth run, $\text{S}^{2'}$ is expanded by applying motion primitives, m and m$'$, and the successor $\text{S}^{4'}$ gets contact with the wall. Its particles are gathered at one place, so its uncertainty measure reduces to near zero. For the states with low uncertainty, the goal attractor is being used, and after two more expansions, the algorithm successfully finds a path to the goal with reduced uncertainty.

V. Experimental Results

A. Motion Planning

1) Box-on-Table (2D): This task is to place a box held by the robot gripper to a specified pose on the table (Fig. 4). For analysis purposes, 12 different start poses are used for the motion planning, and the results are shown in Fig. 6 and Table 2. It can be seen that the planned motion utilizes contact with the walls close to the start and goal poses to reduce uncertainty.

The contact attractors were at the walls, not at the corners (as in Fig. 4(b)), but many motion plans got to the corner. It can be interpreted that contact attractors don't introduce significant artifacts to the solution path.

2) Box-on-Table (3D): Planning results for box-on-table task in 3D space are presented in Fig. 7 and Table 3. It is compared to the 2D planning case that only differs in the initial z-position. As can be seen in planning time and number of expansions, the 3D planning case obviously suffers from the curse of dimensionality. This is somewhat expected because the number of simulations per expansion equals to the number of particles times the number of motion primitives, which should be (4×4) and (6×6) for 2D and 3D, respectively.

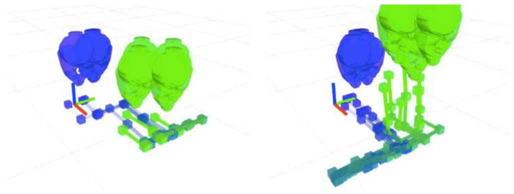
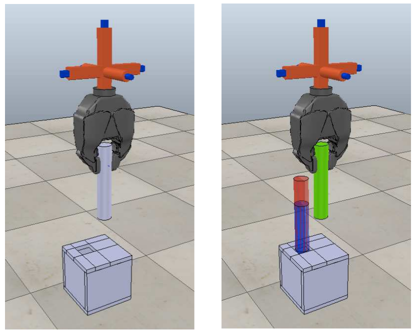


Fig. 7: Comparison of planning results of 2D (left) and 3D (right) box-on-table tasks.

TABLE III: Comparison of Planning Results of 2D and 3D Box-on-Table Tasks
(The motion step sizes and initial particle offsets are the same, but the numbers of particles are 4 for 2D case and 6 for 3D case, respectively, and the nominal pose in 3D case is 0.13 m higher than the goal pose.)

Task ID	Time [s]	Cost	Node # in Path / Expansion #	Pos. Error Mean [m]	Pos. Error Std Dev [m]
6 (2D)	396	47847	11/11	0.0273	0.0290
6' (3D)	1322	42958	11/17	0.0611	0.0009



(a) Models of a peg and a hole. (b) A given problem.

Fig. 8: Illustration of 2D peg-in-hole task. (b) The green peg is the start pose, and the blue peg is the goal pose. The red peg is a goal attractor for an inadmissible heuristic.

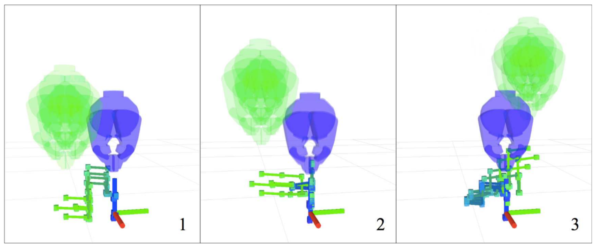


Fig. 9: Planning results for 2D peg-in-hole tasks with 3 different start poses.

TABLE IV: Planning Results for 2D Peg-in-Hole Tasks (The step size for translational motion primitives is 0.10 m, and the initial offset from the nominal start pose of each particle is 0.04 m.)

Task ID	Time [s]	Cost	Node # in Path / Expansion #	Pos. Error Mean [m]	Pos. Error Std Dev [m]
1	706	12164	16/21	0.0032	0.0009
2	604	12474	16/19	0.0029	0.0018
3	830	15311	22/25	0.0035	0.0011
Mean	713.1	13316.3	18.0/21.6	0.0032	0.0013
Std Dev	113.2	1734.4	3.1/3.5	0.0003	0.0005

3) Peg-in-Hole (2D): This task is to insert the peg into the hole. One interesting point of planning task is that, by backward attractor search from the goal, an attractor state was found at the entrance of the hole and set as a goal attractor, not a contact attractor in a high uncertainty region. This is reasonable because the state at the entrance should have low uncertainty as the goal state. The results for three different cases are shown in Fig. 9 and Table 4. The first and third cases reduce uncertainty by contact with the top and the left side of the box, but the second case does that by contact with the top and the inner side of the hole.

B. Motion Execution

1) Box-on-Table (2D): Physical robot experiments were conducted for a 2D box-on-table task (Fig. 10). 10 different runs used the same motion plan searched for the given (nominal) start and goal poses, but the actual start pose for each run was perturbed by Gaussian noise with 0.05 m standard deviation. As shown in Fig. 11, the standard deviation of the final pose reduced to tenth of the initial one. However, there were remaining errors from the goal due to relatively large motion primitive displacements.

2) Peg-in-Hole (2D): Robot experiment results for peg-in-hole tasks are presented in Fig. 12 and Fig. 13. As in the box-on-table case, 10 different runs used the same planned motion computed by the planner for the given start and (nominal) goal poses. The right gripper of the robot held the box with a hole (see Fig. 12), but it was just used to provide a perturbed goal pose for each run. The motion plan from our proposed planner was able to successfully insert the peg into the perturbed hole in all 10 runs.

In order to demonstrate the significance of the proposed approach, two baselines are compared for these tasks. One is a *single-particle planner* which considers only one particle without the notion of uncertainty. The other is a *contact-greedy planner*, an extension of the single-particle planner, which tries to get to the nearest contact point as soon as possible, exploiting the uncertainty reduction effect of contact. In both planners, physics-based simulation was necessary to construct physically feasible graphs.

As shown in Fig. 13, the motion plan from the single-particle planner was very fragile under uncertainty so that it succeeded only in one case which has the almost same height with the nominal goal pose. The contact-greedy planner was more robust than the single-particle planner, but as it also considers only one particle, its motion plan failed in more than the half of the runs.

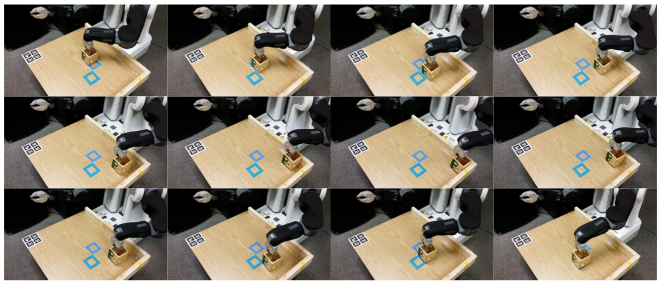


Fig. 10: Snapshots of a robot experiment for a 2D box-on-table task. The motion was planned for a nominal start position (upper square in the picture) and a goal position (lower square in the picture). The actual start position deviated from the nominal start position by 6 cm, but the goal could be accomplished.

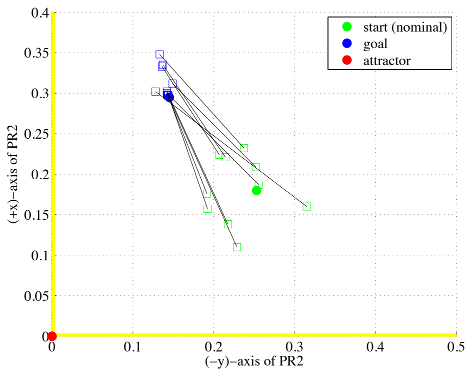


Fig. 11: Plot of robot experiments for 2D box-on-table tasks. (This plot is rotated by 90 degrees, so that it can be depicted as the PR2 views the table.) The green filled circle is the nominal start position, and the blue filled circle is the goal position. Green and blue squares are the initial and the final positions of 10 individual test runs, and the pairs are connected by black solid lines. The red filled circle is the attractor used for planning, and the yellows bars represent the walls of the table. The standard deviation of start position was set to 0.05 m, and the perturbed start position of each test is randomly drawn from the corresponding Gaussian distribution. The step size of translational motion primitives is 0.10 m. The mean and standard deviation of position errors to the goal are (0.0175, 0.0053) and (0.0189, -0.0061) in meter, respectively.

VI. Conclusion

We proposed a motion planning framework for parts assembly under uncertainty. This problem is formalized as

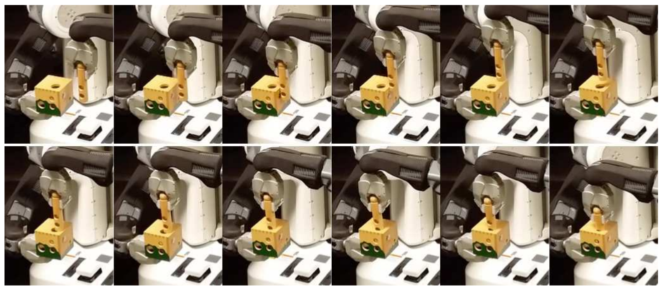
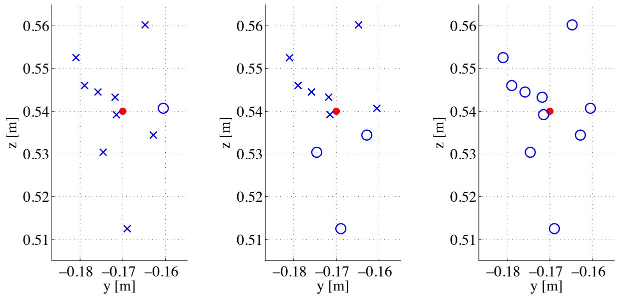


Fig. 12: Snapshots of a robot experiment for a 2D peg-in-hole task.



(a) Baseline (single-particle) (b) Baseline (contact-greedy) (c) Our planner

Fig. 13: Plot of robot experiments for 2D peg-in-hole tasks. The red filled circle is the nominal goal position (the true mean of the Gaussian distribution of the goal position) that is used for planning, and the blue circles or crosses are the actual goal poisitions for 10 test runs that are drawn from Gaussian distribution with a standard deviation of 0.01 m. The step size of translational motion primitives was 0.04 m. Note that in these experiments the peg held by the left gripper is set to be at the same start position for all the test runs, but the box with a hole held by the right gripper which serves as an environment under uncertainty (without any sensor feedback) is set to be at a perturbed goal position. A circle marker represents that a peg-in-hole task for the corresponding goal position was successfully completed, i.e., the peg was inserted into the hole, by the planned motion of a specific planner, while a cross marker represents that the corresponding peg-in-hole task was not successful.

a graph search problem in a foliated belief space where uncertainty reduction is only possible in narrow passages of contact. Physics-based simulation is used to construct a physically reasonable graph, and foliated heuristic functions with contact and goal attractors are adopted in Multi-Heuristic A* search framework to accelerate the search. The planning and experimental results for box-on-table and peg-in-hole tasks demonstrated the effectiveness of our approach.

As a future work, it would be interesting to study how sampling-based methods can help finding good attractor states for inadmissible heuristics and be combined with combinatorial search algorithms. Based on the fact that simulation-based reasoning is highly expensive, we would like to incorporate Experience Graph (E-Graph) into our framework, so that we can reuse the previous planning results for other similar tasks [21].

References

[1] K. Y. Goldberg, "Orienting polygonal parts without sensors," *Algorithmica*, vol. 10, no. 2-4, pp. 201–225, 1993.

[2] S. Akella and M. T. Mason, "Parts orienting with partial sensor information," in *1998 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1. IEEE, 1998, pp. 557–564.

[3] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1578–1585.

[4] C. Eppner, R. Deimel, J. Álvarez-Ruiz, M. Maertens, O. Brock, *et al.*, "Exploitation of environmental constraints in human and robotic grasping," *International Journal of Robotics Research*, vol. 34, no. 7, pp. 1021–1038, 2015.

[5] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "IkeaBot: An autonomous multi-robot coordinated furniture assembly system," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 855–862.

[6] M. Dogar, R. A. Knepper, A. Spielberg, C. Choi, H. I. Christensen, and D. Rus, "Towards coordinated precision assembly with robot teams," in *Experimental Robotics*. Springer, 2016, pp. 655–669.

[7] D. Lenz, M. Rickert, and A. Knoll, "Heuristic search in belief space for motion planning under uncertainties," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2659–2665.

[8] J. P. Gonzalez and A. Stentz, "Planning with uncertainty in position: an optimal and efficient planner," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 2435–2442.

[9] ——, "Planning with uncertainty in position using high-resolution maps," in *2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1015–1022.

[10] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1448–1465, 2009.

[11] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 723–730.

[12] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[13] J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour, "Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3075–3082.

[14] M. C. Koval, N. S. Pollard, and S. S. Srinivasa, "Pre-and post-contact policy decomposition for planar contact manipulation under uncertainty," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 244–264, 2016.

[15] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 156–163.

[16] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[17] J. Butzke, K. Sapkota, K. Prasad, B. MacAllister, and M. Likhachev, "State lattice with controllers: Augmenting lattice-based path planning with controller-based motion primitives," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 258–265.

[18] *(2016) Coppelia Robotics, V-REP (Virtual Robot Experimentation Platform). [Online]. Available: http://www.coppeliarobotics.com.*

[19] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-Heuristic A*," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[20] F. Islam, V. Narayanan, and M. Likhachev, "Dynamic Multi-Heuristic A*," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2376–2382.

[21] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, "E-Graphs: Bootstrapping planning with experience graphs." in *Robotics: Science and Systems*, 2012.