Reliably arranging objects in uncertain domains

Ariel S. Anders, Leslie P. Kaelbling, and Tomas Lozano-Perez

***Abstract*— A crucial challenge in robotics is achieving reliable results in spite of sensing and control uncertainty. In this work, we explore the conformant planning approach to robot manipulation. In particular, we tackle the problem of pushing multiple planar objects simultaneously to achieve a specified arrangement without external sensing. Conformant planning is a belief-state planning problem. A belief state is the set of all possible states of the world, and the goal is to find a sequence of actions that will bring an initial belief state to a goal belief state. To do forward belief-state planning, we created a deterministic belief-state transition model from supervised learning based on off-line physics simulations. We compare our method with an on-line physics-based manipulation approach and show significantly reduced planning times and increased robustness in simulated experiments. Finally, we demonstrate the success of this approach in simulations and physical robot experiments.**

I. Introduction

Constructing multi-object arrangements is an important component of a variety of robot tasks: arranging boxes in a warehouse, placing groceries on a shelf, packing objects in boxes, etc. The fundamental goal in these tasks is to achieve a set of relative position constraints among a set of parts; in general, these constraints require contacts among multiple objects. These contacts are difficult or impossible to perceive visually and difficult or impossible to achieve via open-loop positioning actions.

A key strategy for robustly constructing object arrangements is to use actions, such as pushing and compliant motions, that achieve desired contact relationships between objects by *exploiting the task mechanics*. For example, the robot can achieve contact between two object faces by pushing one up against the other, or achieve an insertion by using a remote-center compliance strategy. These actions have a tendency to act as "funnels," [1, 2] which can map a large set of possible initial configurations into a more compact set.

In general, the result of these actions is non-deterministic, depending on the actual (partially observed) initial state of the world and (unobserved) properties of the robot and objects. Given this setting, we are faced with the question of how to plan sequences of actions that can achieve a desired goal state without the ability to sense the precise outcome of each action. We frame this problem as one of *conformant planning*: given a set of possible initial object configurations, a set of actions with non-deterministic outcomes, and a set of goal configurations, the objective is to find a sequence of actions that is guaranteed to drive the objects into a configuration satisfying the goal. The problem of conformant planning can be seen as a search through a space of *belief states*, which are sets of possible configurations of the objects in the arrangement. Each primitive action is modeled using a transition function that maps an initial belief state into a resulting belief state; the resulting belief state is the union of the possible configurations resulting from applying that action primitive to every configuration in the initial belief state.

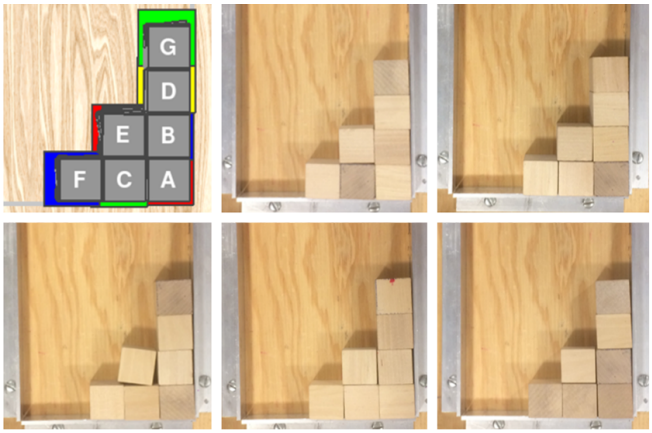


Fig. 1. Robust execution of 7-block arrangement on a real robot; results shown for 1000 noisy simulated executions and 5 executions on a real robot.

It is difficult to analytically derive a belief-state transition function because it depends on interactions among multiple objects which may be affected by detailed physical properties of the objects and robot. One approach is to rely on physics-based simulations. In the presence of uncertainty, we can use multiple simulations to model transitions, as in a particle filter. We show later that obtaining reliable plans with this approach requires a relatively large number of "particles" and prohibitive amounts of computation.

We present an alternative approach that uses machine learning methods to acquire belief-state transition models from physics simulations prior to planning. We learn transitions for the belief state of individual objects given local context, called *compositional belief-state transition models* (CBSTs), and then compose these to construct a belief-state transition model for the overall system. Although we demonstrate the learned model in the context of conformant planning, with no external sensing, it could be used to

Massachusetts Institute of Technology, Cambridge, MA USA {aanders, lpk, tlp@mit.edu}

We gratefully acknowledge support from support from NSF grants 1420316, 1523767, and 1723381, from AFOSR FA9550-17-1-0165 and from Draper Laboratory. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

Fig. 2. Real robot reliably assembling an arrangement. First the robot places a block, then pushes it using a paddle.

implement a belief-space replanning strategy [3] that would incorporate intermediate sensing. The conformant setting offers a simpler, cleaner framework for evaluating the effectiveness of these learned models.

We begin by describing a very general class of conformant arrangement planning problems and outlining a generic search-based solution strategy for them. We then describe a particular instance of this problem class, in which the robot pushes objects resting on a table into a specified configuration, as shown in Figure 1. Although this task is quite constrained, it does require simultaneous interaction between the robot, the pushed objects, and static objects in the environment. We show that, in spite of its apparent simplicity, deterministic approximations are ineffective and on-line simulation approaches are intractable. We present methods for learning the compositional transition model from a physics-based simulation. We finish by describing experiments on both simulated and physical robots, as seen in Figure 2. Our results demonstrate effective planning and execution for assemblies of up to seven objects that considerably outperforms deterministic and particle-based models.

Related work: A key challenge in the "exploit task mechanics" approach to manipulation is to understand the task mechanics well enough to be able to choose actions to achieve the desired outcomes. A number of operations, most notably pushing, have been the subject of extensive study, and analytic models have been derived for the task mechanics; Mason [4] provides a helpful overview. In many cases, however, the outcomes of actions depend on physical properties that we do not have access to, such as pressure distributions or frictional coefficients. Nevertheless, in some cases the understanding of task mechanics can be exploited to plan robust manipulation. For example, Lynch and Mason [5] showed how to exploit multiple contacts to produce stable pushing trajectories and Dogar and Srinivasa [6] showed how to reliably funnel a range of initial locations of an object with known shape into a target location in the hand.

Recently, there have been a number of examples of *physics-based manipulation*, which exploit the availability of physics simulation engines, originally developed for computer games, to predict the effect of actions in situations where analytic methods would be cumbersome at best. Most relevant to our work is the work on rearrangement planning [7], which addresses pushing an object to a target location in the presence of "clutter", objects that are allowed to be pushed out of the way. Planning these operations relies on being able to predict, using physics simulations, the motion of multiple interacting objects being pushed by the robot. Related work on grasping through clutter also relies on such simulations [8, 9]. Note that this work is generally focused on placing or grasping a single object, even though several "clutter" objects will end up being moved in the process; it does not address the goal of achieving a final arrangement of multiple objects.

An alternative to using analytical models or on-line simulation for physics-based planning is to learn compact models from experience (real or simulated). In the observable case, where the state of the world after an action can be determined, this is a simple regression problem: given many observations of *(state, action, next state)* learn a function to predict the *next state* given *(state, action)*. Given such a learned function, planning proceeds as before. A number of instances of this approach exist [10, 11, 12].

It is also possible to try to learn a policy directly and bypass planning; for example, Laskey et al. [13] learn a policy for grasping in clutter. However, such learned policies tend not to generalize as well as learning a model and then planning.

The work described above typically assumes that the initial state is known and the actions are reliable. In the presence of uncertainty in the initial placements of objects or in the outcome of the actions, we are interested in finding actions, such as pushing and compliant motions, that reliably achieve the goal state in spite of this uncertainty and without assuming the availability of additional observations. This class of problems is known as *conformant planning*; it has been explored in robotics for planning compliant motions [2] and sequences of tray tilting operations [14] and, more recently, for rearrangement planning [15, 16].

Outside the context of planning, there has been work in learning to predict the effect of object interactions without full state information [17, 18, 19]. For example, Kroemer et al. [19] worked on classifying object contact distributions to predict whether a set of inputs would yield a stable grasp or place. This contact classifier in conjuction with sampling object positions enabled a 1-step planner that could be used sequentially to stack objects resting on a table. In contrast, we are learning composable models to be used in a multi-step planner for construcing multi-object arrangements.

More broadly, the notion of conformant planning has been explored in the AI community starting with Kushmerick et al. [20], who addressed it within the framework of probabilistic planning, and Goldman and Boddy [21] who used expressions in a logic of knowledge to characterize belief states. Early work from theoretical computer science [22] shows that finding a finite-horizon optimal policy for a completely unobservable MDP is NP-complete, making this class of problems more efficient to solve than POMDPs in general. Yu et al. [23] found that conformant planning is an effective strategy for a multi-robot "tag" domain.

In this paper we address a problem that is related to the physics-based rearrangement planning problem of prior work [7, 15] but is substantially different; in particular, it

requires assembling a number of objects into a specified pattern. A sequence of distinct trajectories (some placing, some pushing) must be planned to achieve the goal.

II. Conformant arrangement planning

A conformant arrangement planning problem is specified by $(\Omega, \mathcal{C}_F, A_\Omega, A_m, \mathcal{T}, G)$, where

- Ω is a set of n known rigid objects;
- $\mathcal{C}_F$ is a set of collision-free complete configurations of all n objects, a subset of the whole configuration space $\mathcal{C}$; we additionally define $\mathcal{C}_O$, where $O \subseteq \Omega$, to be the space of collision-free configurations of the subset of objects O;
- A_o is a set of actions that introduce object o into the arrangement; we let $A_\Omega = \cup_{o \in \Omega} A_o$;
- A_m is a set of actions that manipulate (change the configurations) of some or all objects currently in the arrangement that are guaranteed to terminate;
- $\tau_{a,o,O} : \mathcal{C}_O \to \mathcal{P}(\mathcal{C}_{O \cup \{o\}})$ is a transition function for the introduction action a that characterizes the effect of introducing object o to the arrangement currently consisting of objects in O by mapping an initial configuration in $\mathcal{C}_O$ to a set of configurations that is a subset of $\mathcal{C}_{O \cup \{o\}}$ ($\mathcal{P}$ denotes the *powerset* operator);
  $\tau_{a,O} : \mathcal{C}_O \to \mathcal{P}(\mathcal{C}_O)$ is a transition function characterizing the effects of manipulation action a on objects in the current arrangement; let $\mathcal{T}$ be the union of these $\tau_{a,o,O}$ and $\tau_{a,O}$;
- $G \subset \mathcal{C}_F$ is the set of configurations that are successful assemblies.

Our approach is to convert this problem into a forward search in *belief space*; a belief space (or information space [24]) is a space of elements that characterizes the robot's information or uncertainty about its domain. In this work, we will let the belief space $\mathcal{B} = \mathcal{P}(\mathcal{C}_F)$, that is, the set of all subsets of $\mathcal{C}_F$, and $\mathcal{B}_O = \mathcal{P}(C_O)$ be the restriction to subset of objects O.

We have defined the transition models τ as mappings from a single configuration to a set of configurations; in some cases it may be more convenient to specify $\tau' : \mathcal{B} \to \mathcal{B}$, mapping a set of configurations into the set of possible resulting configurations; such a model can be directly constructed from τ as $\tau'(b) = \bigcup_{c \in b} \tau(c)$.

In a continuous configuration space, it will be impossible to finitely represent all possible subsets of $\mathcal{C}_F$. Therefore, we propose to use a conservative approximation of τ' that generates belief states that are correct (contain all possible true configurations) but may not be as small as possible. Let $\widehat{\mathcal{B}}$ be a set of compactly representable elements of $\mathcal{B}$; then we define a conservative transition model $\widehat{\tau} : \widehat{\mathcal{B}} \to \widehat{\mathcal{B}}$, such that $\widehat{\tau}(b)$ is a smallest element $\hat{b} \in \widehat{\mathcal{B}}$ such that $b \subseteq \hat{b}$.

The size of the configuration space grows exponentially with the number of objects in the arrangement, and the size of the belief space grows exponentially with the size of the configuration space (in the discrete case). In order to fight this curse of dimensionality we use a factored representation of belief states. If we think of a configuration $c = \langle c_1, \ldots, c_n \rangle$, where $c_i \in \mathcal{C}_i$ is the configuration of object i, then $\mathcal{C} = \mathcal{C}_1 \times \ldots \times \mathcal{C}_n$ is the Cartesian product of the configuration spaces of the individual objects. We cannot generally represent $\mathcal{C}_F$ as a Cartesian product, however, due to collisions between the objects.

This decomposition leads us to the idea of representing the belief space as the product of independent beliefs about the configuration of each individual object, so that $\widehat{\mathcal{B}} = \widehat{\mathcal{B}}_1 \times \ldots \times \widehat{\mathcal{B}}_n$, where $\widehat{\mathcal{B}}_i \subset \mathcal{P}(\mathcal{C}_i)$. Representing sets of possible configurations of each object rather than sets of possible complete configurations is much more compact, although it is not as expressive. For example, it is necessary to augment such a factored representation with a constraint that the entire configuration must be collision free, so we will in general define $\widehat{\mathcal{B}} = (\widehat{\mathcal{B}}_1 \times \ldots \times \widehat{\mathcal{B}}_n) \cap \mathcal{C}_F$.

Given a conformant arrangement planning problem specification $(\Omega, \mathcal{C}_F, A_\Omega, A_m, \mathcal{T}, G)$ and a belief-state representation $\widehat{\mathcal{B}}$, we can apply the A^* (or other forward search) algorithm to solve it, if we restrict the action sets A_Ω and A_m to be finite. There may be sample-based search strategies that are effective in infinite action spaces but we do not consider them here. The search problem given to A^* is:

- Initial state: the belief state containing a single element, which is the empty configuration $\{\langle\,\rangle\}$;
- Successor function:

$$\begin{aligned}\mathrm{succ}(\hat{b}) = &\{\widehat{\tau}_{a,O}(\hat{b}) \mid a \in A_m\} \cup \\ &\{\widehat{\tau}_{a,o,O}(\hat{b}) \mid o \in \Omega \setminus O, a \in A_o\}\end{aligned}$$

  where $\hat{b}$ contains the set of objects O;
- Goal test: $g(\hat{b}) = \hat{b} \subseteq G$.

We illustrate the general class of conformant arrangement planning problems with an example domain, implemented in simulation and on a real robot, which uses a manipulator arm to place and push objects into planar arrangements.

III. Arrangement by pushing

In this section, we formalize a concrete robotics problem as an instance of conformant arrangement planning. The goal is to create planar arrangements of objects in contact with one another and with fixed obstacles using a combination of "gross" and "fine" motions [2]. The gross motions use a robot hand to place objects near their target poses, with considerable error in object placement due to control and calibration error in the arm and to objects sticking slightly to the fingers when they are released. In addition, the robot's fingers are large, so objects cannot be placed directly next to other objects. The fine motions use the robot hand, holding a paddle, to push objects up against one another and the fixed obstacles, effectively aligning them and moving them into place.

We can describe these problems as instances of the general conformant arrangement planning problem as follows:

- The objects Ω are wooden cubes, 1 inch on each side, that can be placed on a planar surface, which has fixed obstacles in a "u" shape as shown in Figure 1.
- The configuration space of each individual object $\mathcal{C}_i$ consists of its position and orientation in $\mathcal{SO}_2$, bounded by

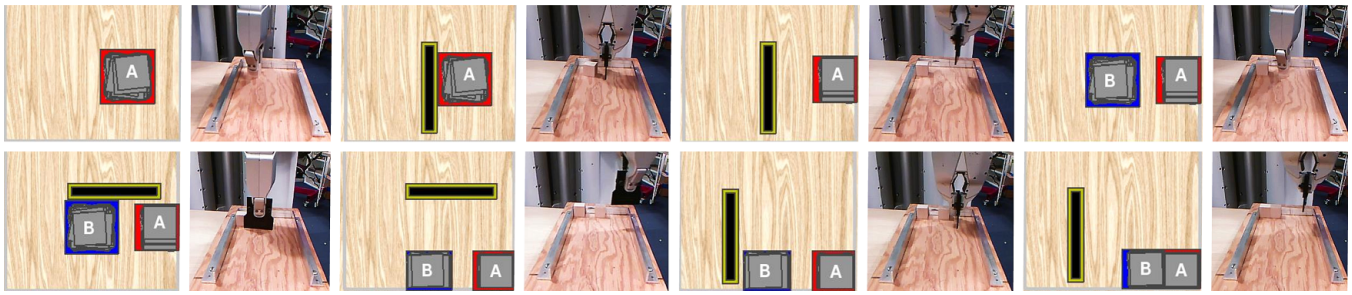


Fig. 3. Execution of a sample 8-step plan; in each pair, the left figure shows the results of 1000 simulations of the plan (gray blocks) as well as the predicted workspace bounding boxes from the transition model (in red and blue) and the right figure shows execution on the real robot.

the workspace area; the free configuration space $\mathcal{C}_F$ of the whole system is the product of the $\mathcal{C}_i$ together with the constraint that the objects are not in collision with each other or the walls.

- The actions A_Ω add an object to the arrangement by using a robot gripper to place an object into free space at a selected (x, y) position; the robot attempts to place the object so that it is rotationally aligned with the workspace, but there is significant error in the calibration, resulting in errors that are well bounded by $\pm 0.2\,\text{in}$ in x and y and $\pm 15°$ in θ; in our current implementation we consider 9 possible values for (x, y) corresponding to constant offsets from the object's target position (x_g, y_g) in the goal.
- The actions A_m manipulate one or more objects already in the arrangement through *push* actions, in which the robot holds a 2.5-inch paddle, places it at pose (x_p, y_p, θ_p) just above the surface of the table and attempts to move in the direction orthogonal to the face of the paddle; the controller has a low gain so that if it encounters obstacles during the motion they remain on the table and are pushed until they can move no further.
- The transition models τ are presented in section III-.1.
- The goal test G specifies a workspace bounding-box $\langle x_g, y_g, \Delta x_g, \Delta y_g \rangle$ for each object and is satisfied if each object is guaranteed to be inside its bounding box.

The representable belief space for each object i, $\widehat{\mathcal{B}}_i$, is a "box" in $\mathcal{SO}_2$, represented with parameters $\langle x, y, \theta, \Delta x, \Delta y, \Delta\theta \rangle$ specifying the center and dimensions of the box; typically $\theta = 0$. We will often denote the center of a belief box by $q = \langle x, y, \theta \rangle$ and its dimensions (the ''uncertainty'') as $\Delta q = \langle \Delta x, \Delta y, \Delta\theta \rangle$. The complete belief space is the product of $\widehat{\mathcal{B}}_i$ for each object, together with the non-collision constraint.

The bounding box can be computed from the configuration-space belief, where, for a square block of dimension $2r$,

$$\text{BB}(\langle x, y, \theta, \Delta x, \Delta y, \Delta\theta \rangle) = \\ \langle x, y, r + \Delta x + r\cos(\theta + \Delta\theta), r + \Delta y + r\sin(\theta + \Delta\theta) \rangle.$$

The parameters of push actions are selected from a set that depends on the current belief state. The pushing angle θ is selected to be either 0 or $\pi/2$ (down or to the right) because the set of goals we consider is limited to arrangements of objects that are supported by the bottom and right walls. We generate the push parameters of a belief state b for a downward push as follows. For each object i currently in the arrangement, with workspace bounding box $\text{BB}(b_i) = (x, y, \Delta y, \Delta x)$, we let $\theta_p = 0$, $x_p = x + \Delta x + \epsilon$ and we consider several possible y_P values, corresponding to $y + \delta$ for $\delta \in \{-1.0, -0.5, 0.0, 0.5, 1.0\}$. For a rightward push, $\theta_p = \pi/2$, $x_p = x + \delta$, and $y_p = y + \delta y + \epsilon$.

A belief state b satisfies the goal G if $\text{BB}(b)$ is entirely contained in $\langle x_g, y_g, \Delta x_g, \Delta y_g \rangle$.

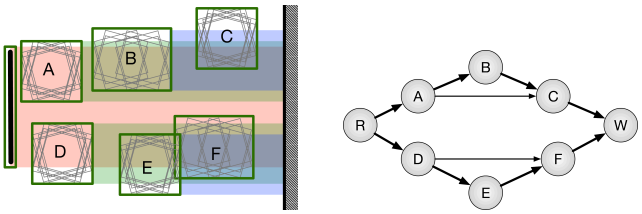
The transition model for object placement is straightforward, simply appending the belief for the object being placed to the existing belief representation:

$$\tau_{(x,y,\theta),o,O}(\langle b_1, \ldots, b_m \rangle) = \langle b_1, \ldots, b_m, \langle x, y, \theta, d_x, d_y, d_\theta \rangle \rangle$$

where $m = |O|$ and (d_x, d_y, d_θ) are bounds on the error in the placement operation. An object placement action is invalid if the workspace bounding box of the introduced object $\text{BB}(b_o)$ overlaps any existing object's workspace bounding box.

The transition model for push actions is significantly more complex, both because the results of pushing a single object are difficult to predict given variability in quantities such as surface texture and force applied by the robot, and because the robot may affect the state of several objects with a single push action. We discuss transition models for pushing in detail in the following subsection.

1) Structured transition model for pushing: The transition model for the push action has a structured decomposition and a local quantitative model that is learned from simulated data. Intuitively, the strategy is to find one or more sequences of blocks that will be pushed up against one another, constrained on one side by the robot paddle and on the other side by a wall (see Figure 3). Given such a sequence, we apply a learned quantitative local uncertainty model to compute the the delta values for the resulting object beliefs, starting from the object closest to the wall and working back toward the paddle. Because we are ultimately going to use this model to search for a plan with reliable effects, it is only necessary to make predictions for actions for which we can confidently predict the posterior belief state. Thus, the transition model will be partial, in some situations declining to make a prediction. This partiality will allow us to maintain the correctness of the planner, but may cause it



(a) Swept volumes and bounding boxes

(b) Contact graph

Fig. 4. a) An example contact graph constructed from the swept volumes in (a). For clarity some arcs are omitted: nodes R and W are connected to all other nodes.

to be incomplete (in the sense that there may be problem instances for which a legal plan exists, but our system is unable to find it.)

The first step is to determine which objects are affected by the pushing operation. To do this, we construct a *contact graph* and extract *contact paths* of objects that are mutually constraining.

The contact graph has a root node R for the robot, a leaf node W for the wall, and nodes for objects that are potentially moved by this pushing operation. Figure 4(a) illustrates a belief state and the process of determining possible contacts.

For each object i, we construct a swept volume of the belief bounding box moving in the direction of the push action. Then we test the bounding box for every object j currently in the arrangement. Every intersection of object j's bounding box with the swept volume creates a potential contact from object i to object j. These potential contacts form directed edges in the contact graph. Figure 4(b) shows the resulting contact graph. Further details on construction of the contact graph and pseudocode can be found in [25].

The next step is to find maximal contact paths between nodes R and W. A contact path p is maximal if there is no other path p' between R and W such that $p \subset p'$. These paths represent "trains" of objects that will end up nearly in contact with one another, pushed between the robot and the wall. In this example, there are two maximal paths: $RABCW$ and $RDEFW$. If the paths are not of the same length, then the prediction process also fails. Consider the case in which object D is not present in this example; the robot paddle will compress objects A, B, and C against the wall but leave E and F unaffected. However, if D were present, but E and/or F were missing, then D would be pushed, but without a constraint on its right, so we are unable to reliably predict its resulting position and uncertainty; in this case, the prediction model returns **None**, indicating that this operation cannot be used to construct a plan.

For each valid contact path, we use a local predictive model compositionally to compute the posterior belief for each object. The predictive model takes as input the belief states of three objects that occur sequentially in the contact path. In the *single-object* pushing model, we disallow any operations in which there is more than one path of influence between the robot and the wall.

2) Learning a quantitative model for pushing: We will work with a factored version of τ', which maps the belief state of an object and its neighbors on either side to its resulting belief state. In the transition model for a push to the right, we begin by predicting the uncertainties in the resulting object position, using a function PREDICT, which is learned from data. It has inputs and outputs in the form:

$$\Delta q' = \langle \Delta x', \Delta y', \Delta\theta' \rangle = \text{PREDICT}(b[prev], b[cur], b[next]) \ ,$$

where the inputs to the procedure are the current beliefs about three sequential objects in a pushing sequence. We assume that the center of the resulting uncertainty box has the same y coordinate as before and that the median rotation is 0. The median x coordinate is computed by finding the median x coordinate of the object to its right, subtracting the dimension of an object, d, and then subtracting the resulting x uncertainty, $\Delta x'/2$.

The problem of learning the PREDICT function can be treated as a supervised regression problem with three output dimensions. Again, for simplicity of exposition, we limit our attention to the push-right action. The inputs to the PREDICT procedure are $(q_p, \Delta q_p, q_c, \Delta q_c, q_n, \Delta q_n)$. We compute from these inputs a feature vector ϕ:

$$\langle \Delta q_p, \Delta q_c, \Delta q_n, q_p.y - q_c.y, q_n.y - q_c.y, ct \rangle.$$

The fourth element of the feature vector $(q_p.y - q_c.y)$ represents the vertical offset between the previous object and current object uncertainty centers. Correspondingly, the fifth element represents the vertical offset between the current object and the next object. The last feature in this vector, ct, encodes the types of the objects involved in this contact; it can take on the values in $\{row, roo, ooo, oow\}$ where r stands for $robot$, o for $object$, and w for wall. For a fixed object, such as a wall, which can only appear as the last object, we assume $\Delta q_n = (0, 0, 0)$, and that $q_n.y - q_c.y = 0$.

Because generating training data on the real robot would be prohibitively costly in time, we generate training data using a Box2D simulation [26]. We construct a data set of 1800 examples, using the following process to construct each example:

- An initial belief state b is randomly constructed, with Δx and Δy values drawn uniformly in the interval $[0.0, 0.4]$ inches and $\Delta\theta$ values drawn uniformly in the interval $[0, 15]°$; b may contain 1, 2, or 3 blocks.
- For 1000 iterations, an initial state is drawn uniformly from b and constructed in Box2D, with the robot paddle offset to the left and a fixed wall, parallel to the paddle, to the right of all the objects. The paddle's y coordinate is randomly varied to generate a variety of offset values. The friction coefficients for the simulation of robot, table, and objects are drawn uniformly in the range $[0.25, 0.55]$. The robot's paddle is moved in the desired direction using a position controller to a distant set-point with gains of 1.0 for position and angular error; the controller is run for 50 simulation steps with $\Delta_t = 0.01\ s$. The final pose of the each object (x_i, y_i, θ_i), together with its contact type, is recorded.

- The resulting belief state dimensions are computed as:

$$\Delta x = xmax - \min_i x_i$$
$$\Delta y = \max_i \|y_i - y\|$$
$$\Delta \theta = \max_i \|\theta_i\|$$

where $xmax$ is the maximum possible x coordinate for this object, if all the objects to its right were perfectly aligned and as far to the right as possible. Notice that Δy and $\Delta\theta$ are computed based on the assumed structure of the transition function: the resulting uncertainty box has the same y coordinate as before and the median rotation is 0. Even though the median values may vary from this assumption, the computed uncertainty widths increase to account for this error making the structured belief-state transition a conservative over approximation.

This data is then used to train a multi-output random-forest regressor using the Scikit-Learn toolkit [27] using hyper-parameters that were found using a grid search with 8-fold cross-validation. We used 90% of the data for training and hyper-parameter optimization and held out 10% for final evaluation, in which we found a root-mean-squared test error of 0.509 in Δx, 0.079 in Δy and 2.799 in $\Delta\theta$.

Search: In section II we described a generic forward-search problem that could be solved using A^* and in the previous section we specified the necessary state space and successor function for our example arrangement-by-pushing domain. We assign a cost of 1 to every action.

To improve the performance of the search we implemented two heuristics. The first, which is admissible, effectively acts as a binary filter on states, assigning infinite cost to any state in which the objects in a horizontal or vertical contact path are not in the order specified by the goal. It is not possible, given the operations in this space, to reach the goal from such a state. This heuristic provides some useful search guidance but is not very strong.

We define an additional heuristic that is inadmissible in general, but highly effective at improving the speed of the search without much reduction in the solution quality. We define:

$$H(b, g) = \sum_{o \in \Omega} H(\mathrm{BB}(b[o]), g[o])$$

where

$$H(bo, go) = \begin{cases} 2 & \text{if } bo == \textbf{None} \\ 2 & \text{if } bo.x \not\subseteq go.x \text{ and } bo.y \not\subseteq go.y \\ 1 & \text{if } bo.x \subseteq go.x \text{ and } bo.y \not\subseteq go.y \\ 1 & \text{if } bo.x \not\subseteq go.x \text{ and } bo.y \subseteq go.y \\ 0 & \text{otherwise} \end{cases}$$

This heuristic estimates that it will take one place action and one push action to add a new object to the arrangement and one push per object dimension that is not currently contained within its goal interval. It is inadmissible because it is possible to push multiple blocks at once.

The simple belief representation of CBST leads to a natural dominance-based pruning method to further improve the search performance. We begin by observing that some belief states are contained by others: given two belief states, b and b', we say that b *dominates* b' if and only if:

- b and b' are both defined on the same set of objects O;
- for all $o \in O$: the configuration space cube $b[o]$ is a subset of the configuration space cube $b'[o]$, that is $b[o] \subseteq b'[o]$.

Intuitively, if b dominates b', then b may be a more useful state in the search: the objects have overlapping possible states and less uncertainty in b than in b'. Therefore, we apply a pruning strategy in which, during the process of A^* search, whenever it reaches a state b' that is dominated by some state b that has already been visited, then b' is not added to the agenda. This strategy can result in computational improvements but it does risk pruning out correct solution paths.

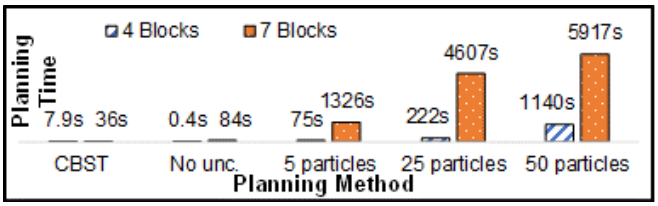


Fig. 5. Comparison of planning time and different belief-state transition models for (a) 4-block and (b) 7-block arrangement.

IV. Results

In this section we present quantitative results of our approach in the arrangement-by-pushing domain.

For comparison, we implemented a planner that approximates the belief state using P sampled states, or "particles" and computes the transition by simulating the action on each one. For place actions, P collision-free placements are sampled. For push actions, each of the particles, which specify the configurations of all the placed objects, is simulated as we do during off-line learning, including adding "noise" in the form of parameter variations in friction, mass, and push parameters. A state satisfies the goal if all of its particles are in the goal region. We also implement a *deterministic planner* by using a single particle to represent the nominal configuration of the objects and not adding noise during simulation. In these planners, we defined the heuristic value of a particle-based belief state to be the maximum over the particles of the H value for each particle, assuming that its BB consists only of a single point.

A. Belief-state transition comparison

We compared the performance, both in terms of accuaracy and speed, of our CBST planner against four different planners based on on-line simulation: a deterministic planner as well as particle-based planners with 5, 25, and 50 particles. We ran each planner on a 4-block arrangement problem and a 7-block arrangement problem, each a total of 10 times.

Figure 5 compares the average planning time for different belief-state transition models. As expected, the deterministic model and CBST are fastest. The planning time with the particle transition models might be expected to scale linearly

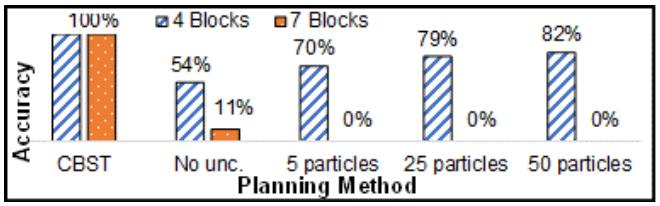


Fig. 6. Execution accuracy for different planning strategies.

Search Type	Domination	Solutions Found	1 Block Nodes	1 Block Cost	2 Blocks Nodes	2 Blocks Cost	3 Blocks Nodes	3 Blocks Cost	4 Blocks Nodes	4 Blocks Cost	7 Blocks Nodes	7 Blocks Cost
BFS	No	50.00%	25	3	160	4	4483	6	--	--	--	--
BFS	Yes	50.00%	27	3	191	4	2982	6	--	--	--	--
Admiss.	No	50.00%	21	3	136	4	1149	6	--	--	--	--
Admiss.	Yes	50.00%	27	3	179	4	1123	6	--	--	--	--
Inadmiss.	No	100.00%	9	3	29	6	60	9	98	12	535	24
Inadmiss.	Yes	83.30%	9	3	28	6	54	9	95	12	284	21

Fig. 7. Search Results Table.

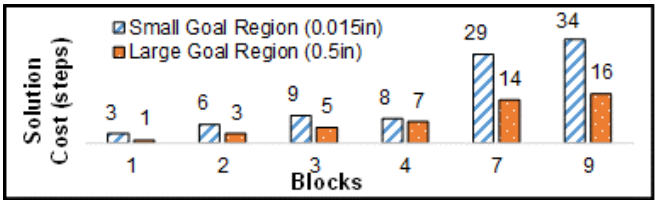


Fig. 8. The effect of goal tolerance on solution length.

with the number of particles, although, as the number of particles increases, they will tend to be more diffused and might actually require a longer plan. We would expect the number of particles required to achieve a reliable planner to grow exponentially with the number of objects. For the 7-block problem, we capped the running time for an expansion at 5000 search nodes; none of the particle-based planners found a solution under this constraint. The reported times for the failed searches show the amount of time it took to reach the 5000 search node constraint.

We evaluated the robustness of the solutions found by the different planners using the physics simulator. We tested each of the 10 plans 1000 times and added simulated noise as used in learning. We reported the percentage of tests in which all objects were in their goal region at termination in Figure 6. The learned CBST planner produced 100 percent accurate results, whereas the on-line simulation approaches produced many non-conformant plans. The deterministic planner is successful a little more than half the time in the 4-block case and a little more than 10% of the time in the 7-block case, showing the need to consider the effect of stochastic actions. In the 4-block case, the particle-based planners improve slowly as function of the number of particles and approach respectable levels of accuracy for 50 particles. In the 7-block case the running time to construct a plan with more than one particle was prohibitive.

B. Detailed CBST experimentation

Planner performance: We tested the CBST planner for four arrangements with 2, 3, 4, and 7 blocks. We varied the goal tolerances (the dimension of the goal regions) between ± 0.1 inches and ± 0.5 inches. The initial placement uncertainty was, unless stated otherwise, ± 0.2 inches in x and y and ± 15 degrees rotation. When a plan was obtained, we simulated it 1000 times. We found that whenever a plan was found, all the simulations satisfied the goal. However, for tight goal conditions, especially for larger assemblies, the search can exceed our limit of 5000 search nodes.

The search performance, as measured by number of search nodes expanded, is affected primarily by the quality of the heuristic used. In this experiment, the search node limit was 5000. In the table in Figure 7, we see that using the "inadmissible" heuristic cuts the number of expanded nodes substantially, at the cost of longer plans. The domination test has a small beneficial effect given this heuristic, but a very large effect when a weaker or no heuristic is used.

The effect of uncertainty: Figures 8 and 9 show the effects of goal tolerance and initial placement uncertainty on plan length. Tighter goals and higher initial uncertainty both increase the length of the required plans. Once the required plans exceed a length of about 16, the search process exceeds the allowed number of search nodes (5000); in such cases, there is no corresponding result plotted in the graphs.

Real robot experiments: An example arrangement sequence found by the planner, as well as its execution using a real PR2 robot, are shown in Figure 3; the accompanying video shows many more examples. We obtained plans for 5 different assemblies (with 2, 3, 4, 7 and 9 blocks) and executed each one on the robot 5 times. The placement uncertainty was ± 0.2 inches and ± 15 degrees rotation; the goal tolerance was ± 0.2 inches. We saw one execution failure (in a 2-block arrangement) during the 25 assemblies, for a 96% success rate.

The execution failure appears to be due to an initial placement outside of the modeled placement uncertainty bounds. Increasing the modeled placement uncertainty could decrease this type of failure, at the expense of increasing the planning and execution times for the typical case. This is an unavoidable trade-off in conformant planning that could be ameliorated by moving to a belief-space replanning paradigm that adds some sensing, such as the final position of the paddle after a push.

Conclusion and future work: One key question is: what fidelity of the transition model is needed for planning?

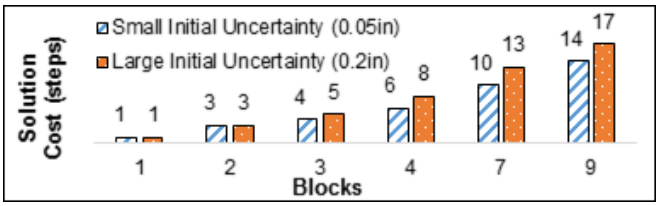


Fig. 9. The effect of initial place uncertainty on solution length. Small uncertainty is ± 0.05 in, large uncertainty is ± 0.2 in; angle uncertainty is ± 15 deg in both cases.

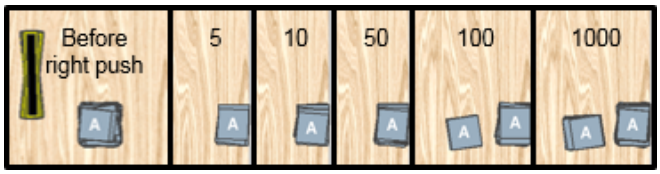


Fig. 10. Approximation of the belief state with different number of particles.

Figure 10 shows a push action that usually pushes the block to the wall, but in some cases misses the block with the paddle due to uncertainty in the vertical offset. In this example, it took over 50 simulations just to see the unlikely result. How should a robot deal with this? Doing on-line simulations to detect unlikely outcomes would be computationally difficult. Learning from on-line simulations is more promising, but representing and planning with a complex posterior distribution is problematic. Following the thread of this work, we could learn a relatively simple model and use sensing and replanning to detect prediction failures. We believe that the replanning approach is most practical, but probing the tradeoff in fidelity of prediction vs sensing is important.

We have shown how a belief-space transition model can be acquired from off-line physics-based simulations and used to plan reliable planar push-arrangements in the presence of substantial uncertainty. We have compared this work with on-line physics-based simulation methods and found results showing much higher accuracy with significantly decreased search time in the box arrangement domain. Clearly, there are cases where the detailed approach described here will not work, such as a cylindrical robot pushing a cylindrical object. Future work will explore whether some of the basic ideas, especially learning factored models, can be generalized more broadly.

References

[1] M. Mason, "The mechanics of manipulation," *Robotics and Automation*, 1985.

[2] T. Lozano-Perez, M. Mason, and R. Taylor, "Automatic synthesis of fine-motion strategies for robots," *IJRR*, 1984.

[3] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," *RSS*, 2010.

[4] M. Mason, *Mechanics of robotic manipulation*. MIT press, 2001.

[5] K. Lynch and M. Mason, "Stable pushing: Mechanics, controllability, and planning," *IJRR*, 1996.

[6] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," *RSS*, 2011.

[7] J. King, J. Haustein, S. Srinivasa, and T. Asfour, "Nonprehensile whole arm rearrangement planning on physics manifolds," *ICRA*, 2015.

[8] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-based grasp planning through clutter," *RSS*, 2012.

[9] N. Kitaev, I. Mordatch, S. Patil, and P. Abbeel, "Physics-based trajectory optimization for grasping in cluttered environments," *ICRA*, 2015.

[10] J. Scholz and M. Stilman, "Combining motion planning and optimization for flexible robot manipulation," *International Conference on Humanoid Robots*, 2010.

[11] S. Elliott, M. Valente, and M. Cakmak, "Making objects graspable in confined environments through push and pull manipulation with a tool," *ICRA*, 2016.

[12] T. Meriçli, M. Veloso, and H. Akın, "Push-manipulation of complex passive mobile objects using experimentally acquired motion models," *Autonomous Robots*, 2015.

[13] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. Pokorny, A. Dragan, and K. Goldberg, "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations," *CASE*, 2016.

[14] M. Erdmann and M. Mason, "An exploration of sensorless manipulation," *Robotics and Automation*, 1988.

[15] M. Koval, J. King, N. Pollard, and S. Srinivasa, "Robust trajectory selection for rearrangement planning as a multi-armed bandit problem," *IROS*, 2015.

[16] A. Johnson, J. King, and S. Srinivasa, "Convergent planning," *IEEE RA-L*, 2016.

[17] M. Kopicki, S. Zurek, R. Stolkin, T. Moerwald, and J. L. Wyatt, "Learning modular and transferable forward models of the motions of push manipulated objects," *Autonomous Robots*, 2017.

[18] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," *CoRR*, 2016.

[19] O. Kroemer, S. Leischnig, S. Luettgen, and J. Peters, "A kernel-based approach to learning contact distributions for robot manipulation tasks," *Autonomous Robots*, 2017.

[20] N. Kushmerick, S. Hanks, and D. Weld, "An algorithm for probabilistic planning," *Artificial Intelligence*, 1995.

[21] R. Goldman and M. Boddy, "Expressive planning and explicit knowledge." *AIPS*, 1996.

[22] C. Papadimitriou and J. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of Operations Research*, 1987.

[23] C. Yu, J. Chuang, B. Gerkey, G. Gordon, and A. Ng, "Open loop plans in POMDPs," Stanford University CS Dept, Tech. Rep., 2005.

[24] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[25] A. Anders, L. Kaelbling, and T. Lozano-Perez, "Planning robust strategies for constructing multi-object arrangements," MIT CSAIL, Tech. Rep., 2017.

[26] E. Catto, "Box2D: A 2D physics engine for games," 2008.

[27] "Scikit-learn: Machine learning in Python," *JMLR*, 2011.