Kinodynamic Reconfiguration Planning with Physics Simulations

***Abstract*—In this work, we introduce an algorithm to solve the reconfiguration planning problem, which requires moving objects to achieve a goal. Existing approximate solutions rely on the availability of a predefined set of primitives, like pushing, sweeping, or grasping. We reformulate the problem as a kinodynamic motion planning problem and utilize a physics simulator for rolling out the forward model, releasing us from the burden of producing primitives. We demonstrate the ability to generate plans using a quasistatic physics model for the forward model rollouts. Additionally, we demonstrate the ability to generate plans using the Box2d physics engine.**

I. Introduction

As we develop robots intended to operate in human environments, the capability to work amongst clutter becomes increasingly important. Consider a robot assistant tasked with getting a snack from the pantry. The shelves are cluttered and unorganized. Several cans of soup and a box of cereal all block the way. The robot must solve a plan to move these objects out of the way in order to grasp the desired snack.

Humans solve this problem easily, pushing and sliding objects to clear a path to the goal. Items blocking the way are fearlessly rearranged with little attention paid to where they end up. In this work, we aim to incorporate these reconfiguration strategies into manipulation planning.

Many works have show the importance of *reconfiguration planning* [5] [15] [17] [20]. Allowing the robot to change the environment makes impossible tasks possible and difficult tasks easier. However, reconfiguration planning has been shown to be NP-hard [21], combining the complexity of motion planning with that of combinatorial search over objects to move.

Existing approximate solutions fall into two categories. The first set uses heuristics such as monotonicity [6] [18] or allowing a fixed number of touches per object [4] to gain tractability. Then, the problem reduces to backtracking from the goal to generate an explicit order to move objects. The second uses randomized planning to select the order to move objects in the scene [3].

Each of these solutions rely on the availability of a predefined set of high-level motion primitives such as straight line pushes, sweeping of the arm or pick-and-place of an object. In addition, they require the planner to explicitly reason about moving objects in the scene one at a time. These restrictions impose two key limitations on the system.

First, the use of motion primitives places a great burden on the designer to develop high-level actions that are both expressive and efficient, transferring, in some ways, computational complexity on to the designer. Second, reducing the problem to an ordered list of moved objects limits the expressiveness of the planner to combinations of high-level primitives. Consider the example shown in Fig.1. Although it is possible to carefully move each of the blue objects aside, then move the green object to its destination, there are far more efficient solutions if we utilize arbitrary pushing actions, which allow to move multiple objects at once.

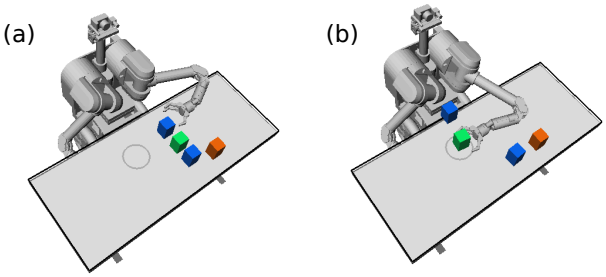

Fig. 1. A solution to the reconfiguration planning problem. Here the green object must be moved inside the circle, where it can be grasped by the right arm. The blue objects are movable while the orange object is not. (a) The start configuration. (b) The final configuration. The blue objects are swept aside by the arm of the robot.

An alternative to using these primitive based methods are kinodynamic motion planners [8] [11]. These problems search an action space for a motion that goes from the start state to a goal state while obeying both obstacle constraints and local differential constraints. Randomized kinodynamic planners have been shown to quickly produce plans in high-dimensional spaces [13], trading off efficiency for speed.

In this work we offer the following contributions.

First, we reformulate the reconfiguration planning problem as a kinodynamic motion planning problem (Sec. III). This enables us to plan forward in the full state space of the robot and movable objects, releasing us from the burden of developing primitives.

Second, we show that all our planner needs is an off-the-shelf physics simulator, treated as a black box, for rolling out the forward model. We present some initial experiments showing results using a custom developed quasistatic simulator and the off-the-shelf Box2d physics engine [1] in Sec. IV.

II. The Reconfiguration Planning Problem

A. Terminology

We use the standard terminology from Lavalle [12]. We work with a robot in a configuration space $q \in \mathcal{C}^R$. Our world is populated with obstacles that must be avoided and also with M movable objects which can be contacted and moved

around. Each movable object is endowed with a configuration space $o^i \in \mathcal{C}^i, i = 1...M$. We define the state space X as the cartesian product space of the robot and objects, given by $X = \mathcal{C}^R \times \mathcal{C}^1 \times ... \times \mathcal{C}^M$. We denote a state $x \in X$ by $x = \left(q, o^1, ..., o^M\right)$.

Following Lavalle [12] and Simeon et al. [16], we define the free state space $X_{free} \subseteq X$ as the set of all states where the robot and objects are not penetrating themselves, the obstacles or each other. Note that this allows *contact* between entities, which is critical for manipulation.

B. The problem

Our task is to find a *feasible* path $\tau : [0, 1] \rightarrow X_{free}$ starting from a state $\tau(0) \in X_{free}$ and ending in a *goal region* $\tau(1) \in X_G \subseteq X_{free}$.

Feasible plans. The state x evolves nonlinearly based on the physics of the manipulation as its time derivative $\dot{x} = f(x, a)$ where $a \in \mathcal{A}$ is an action from the set of control actions that can be applied to the robot. A plan is feasible if for all $t \in [0, 1]$ there exists an $a_t \in \mathcal{A}$ such that $\dot{\tau}(t) = f(\tau(t), a_t)$, i.e. there exist robot actions that can actually perform the plan.

For example, in Fig.1, $\mathcal{C}^R = \mathbb{R}^7$, the set of joint configurations for the left arm, and $\mathcal{C}^i = SE(2)$, the set of poses in the plane. We define the set of actions, $\mathcal{A}$, as the set of joint velocities for the left arm. A feasible plan can be quite complicated, making and breaking multiple contacts with movable objects and reconfiguring them to achieve the goal.

Goal region. In most manipulation problems, the goal is often underspecificed. For example, in [17] only the robot's goal is specified. In many other problems [7] the task is to move a specific object to a specific place (or a set of places). We denote this object as the *goal object* G with its configuration space $g \in C^G$ and its goal as the set $\mathcal{G} \subseteq C^G$. We define $X_G \subseteq X_{free}$ as the set of all states with the goal object in $\mathcal{G}$.

Much of the planning complexity arises because the system is under actuated, with one robot having to move several objects, and nonlinear, due to the physics of manipulation. In Sec. III, we describe how we can combine black box physics simulations with ideas from randomized kinodynamic planning to produce fast, feasible plans.

III. Forward Search with Trajectory Rollouts

We utilize a kinodynamic RRT [13] to perform a forward search through our space. Alg.1 shows the basic algorithm. A kinodynamic RRT differs from a traditional RRT in the implementation of the tree extension. Rather than solving the two-point boundary value problem, the kinodynamic RRT algorithm samples k actions, propagates them forward under a transition function $T : X \times A \rightarrow X$ and selects the best using a distance metric defined on the state space. The result is a plan that obeys both kinematic and dynamic constraints.

In this work, we utilize the algorithm to ensure our plans adhere to the constraints on object motion induced by pushing actions. We implement the transition function using a physics simulator. The general solution places no requirements on the underlying physics model used by the simulator.

Input: The start state x_0, integer k
Output: The path
$tree \leftarrow \{nodes = \{x_0\},\ edges = \emptyset\}$
$path \leftarrow \{\}$
while *not* `ContainsGoal(`$tree$`)` **do**
  $x_{rand} \leftarrow$ `SampleConfiguration ()`
  $x_{near} \leftarrow$ `NearestChild (`$tree$`,` x_{rand}`)`
  $\{a_1, \ldots, a_k\} \leftarrow$ `SampleActions (`k`)`
  $(x_{new}, a) \leftarrow$ `T (`$(x_{near}, a_1), a_1$`)`
  for $i = 2 \ldots k$ **do**
    $x_i \leftarrow$ `T (`x_{near}, a_i`)`
    if `Dist (`x_i, x_{rand}`)` $<$ `Dist (`x_{new}, x_{rand}`)` **then**
      $(x_{new}, a) \leftarrow (x_i, a_i)$
  $tree.nodes \cup \{x_{new}\}$
  $tree.edges \cup \{((x_{near}, x_{new}), a)\}$
$path \leftarrow$ `ExtractPath (`$tree$`)`

Algorithm 1: The kinodynamic RRT algorithm using a transition function T

IV. Experiments

We implemented the algorithm by extending the Open Motion Planning Library (OMPL) framework [19]. To prove the feasibility of our proposed method, we run several experiments in simulation. For each experiment, we test against four scenes of varying difficulty.

We run the planner several times on each scene and collect results. A trial is considered successful if a solution is found within 300 seconds. In all experiments, we give the goal-set a radius of 0.1 meters. We set the number of control samples $k = 10$ for all experiments.

For each experiment we report two metrics:

- Success rate - The percentage of trials that produced a successful plan in less than 300 seconds
- Plan time - The average time to generate a successful plan.

A. Quasistatic Physics Model

In our initial experiments, we consider a simple single contact quasistatic pushing model with Coulomb friction [14]. We assume friction between the object and the underlying surface is finite, and that the pressure distribution between the object and the surface is also finite. Under these assumptions, we can analytically derive the nonlinear motion of an object when pushed by the hand [9] [10]. We model only robot-object contact, and invalidate any action that introduces object-object contact.

We plan for a 7-DOF manipulator, like the arm of the robot shown in Fig.1. The state space of each movable object is modeled in $SE(2)$. During planning, we constrain the motion of the arm such that the end-effector remains parallel with the pushing surface.

Tab.I shows the results for these experiments.

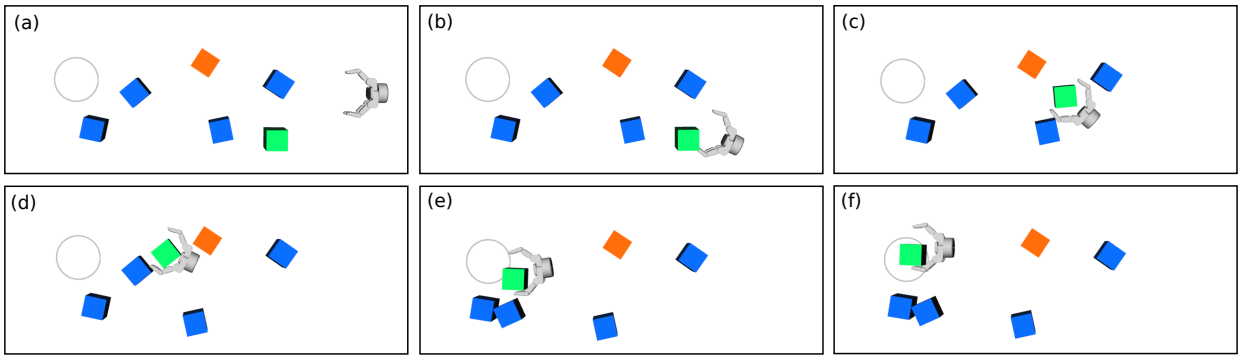

Fig. 2. An example solution for planning with Box2d.

Scene	One	Two	Three	Four
Plan time (s)	12.37	21.25	27.18	20.94
Success rate (%)	80	100	80	20

TABLE I
RESULTS FOR EXPERIMENTS PLANNING FOR A 7-DOF ARM.

Scene	One	Two	Three	Four
Plan time (s)	5.69	9.51	32.23	46.28
Success rate (%)	100	100	100	100

TABLE II
RESULTS FOR EXPERIMENTS USING A BOX2D PHYSICS SIMULATOR.

B. Box2d Physics Model

In the next set of experiments, we utilize the Box2d physics engine for trajectory rollouts. Use of Box2d as our physics simulator releases us from two key limitations present in the quasistatic simulations reported in Sec. IV-A. First, we can allow object-object interaction, as these are easily modeled by the physics engine. The result is the ability to generate plans where the manipulator pushes one movable object into another, creating a chain of moving objects.

Second, we can now model dynamic interactions. To model full dynamics, we must double the size of the state space, tracking both position and velocity for the manipulator and each movable object. Instead, in these results, we chose a semi-dynamic approach. For a semi-dynamic interaction, we allow impulsive manipulation, but require that all objects come to rest within a specified amount of time T_{max} after the action was executed. This allows our state space to track only the position of the manipulator and movable objects, while simulating a fully dynamic transition between states. This enables the planner to also find solutions in low friction environments such as objects on ice.

The choice of our simulator restricts us to two dimensional environments, so in these experiments we model the manipulator as a free-floating end-effector in SE(2).

Tab.II shows results of experiments on the four scenes. Fig.2 shows several keyframes from one solution found by the planner.

V. CONCLUSION AND DISCUSSION

In this work, we have presented a formulation of the reconfiguration planning problem as a kinodynamic motion planning problem, and presented a randomized kinodynamic planner to solve the problem. We have shown the use of two different physics simulators to generate valid reconfiguration plans. In addition, we have shown the ability to generate plans for manipulator in $SE(2)$ and in higher-dimensional state spaces.

The results in Sec. IV-B show that we can use the dynamic modeling available in open-source physics simulators to model complex dynamic interactions between objects and the manipulator. We restricted our results to semi-dynamic interactions. In future work we wish to examine the problem of planning with full dynamics as well.

The use of the Box2d simulator restricts us to two-dimensional evironments. The use of a higher-dimensional physics simulator, such as Bullet [2], would allow us to plan dynamic interactions for manipulators in higher-dimensional spaces, such as the one used for experiments in Sec. IV-A.

Finally, the results we presented were obtained using only a simulation environment. In the immediate future, we plan to execute the generated trajectories on an actual manipulator. This will allow us to understand whether the simulations we achieve during trajectory rollouts are accurate enough to allow successful execution.

REFERENCES

[1] Box2d. http://box2d.org, 2010 (accessed August 2014).
[2] Bullet physics library. http://bulletphysics.org, (accessed August 2014).
[3] Jennifer Barry, Kaijen Hsiao, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Manipulation with multiple action types. In *ISER*, 2012.
[4] Akansel Cosgun, Tucker Hermans, Victor Emeli, and Mike Stilman. Push planning for object placement on cluttered table surfaces. In *IEEE/RSJ IROS*, 2011.
[5] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. In *RSS*, 2011.
[6] Mehmet Dogar and Siddhartha Srinivasa. A planning

framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots*, 2012.

[7] Mehmet R. Dogar, Kaijen Hsiao, Matei Ciocarlie, and Siddhartha S. Srinivasa. Physics-based grasp planning through clutter. In *RSS*, 2012.

[8] Bruce Donald, Patrick Xavier, and John Reif. Kinodynamic motion planning, 1993.

[9] S. Goyal, A. Ruina, and J. Papadopoulos. Planar sliding with dry friction. part 1. limit surface and moment function. *Wear*, 1991.

[10] Robert D. Howe and Mark R. Cutkosky. Practical force-motion models for sliding manipulation. *IJRR*, 1996.

[11] R. Kindel, D. Hsu, J. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. In *IEEE ICRA*, 2000.

[12] Steven M. Lavalle. *Planning Algorithms*. 2006.

[13] Steven M. Lavalle and James J. Kuffner Jr. Randomized kinodynamic planning. *IJRR*, 2001.

[14] Kevin Lynch and Matthew T. Mason. Stable pushing: Mechanics, controllability, and planning. In *WAFR*, 1995.

[15] D. Nieuwenhuisen, A. Stappen., and M. Overmars. An effective framework for path planning amidst movable obstacles. In *WAFR*, 2008.

[16] Thierry Simeon, Jean-Paul Laumond, Juan Cortes, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. 2004.

[17] M. Stilman and J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *IEEE-RAS Humanoids*, 2004.

[18] M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour. Manipulation planning among movable obstacles. In *IEEE ICRA*, 2007.

[19] I. Sucan, M. Moll, and L. Kavraki. The open motion planning library. *IEEE Robotics and Automation Magazine*, 2012.

[20] Jur van den Berg, Mike Stilman, James Kuffner, Ming Lin, and Dinesh Manocha. Path planning among movable obstacles: a probabilistically complete approach. In *WAFR*, 2008.

[21] Gordon Wilfong. Motion planning in the presence of movable obstacles. *Annals of Mathematics and Artificial Intelligence*, 1991.