Affordance Wayfields for Task and Motion Planning

Troy McMahon[1], Odest Chadwicke Jenkins[1] and Nancy Amato[2]

Abstract— **Affordances provide a natural means for a robot to describe its agency as actions it can perform on objects. Further, affordances can enable robots to reason complicated, multi-step tasks that involve proper use of a diversity of objects. This paper proposes the concept of affordance wayfields for representing manipulation affordances as objective functions in configuration space. Affordance wayfields quantify how well a path, or sequence of motions, will accomplish an afforded action on an object. Paths that enact affordances can be located by performing a randomized form of gradient descent over affordance wayfields. Incorporating obstacles, or other constraints into wayfields allows our method to adaptively generate valid motions for executing afforded actions. We demonstrate that affordance wayfields can enable robots, such as the Michigan Progress Fetch mobile manipulator, to solve complex real-world tasks such as assembling a table, or loading and unloading objects from a storage chest.**

I. Introduction

Autonomous robots capable of fulfilling high-level end-user requests could revolutionize many applications across society. Eldercare, manufacturing, and interplanetary exploration are but a few of the domains where mobility and manipulation could be game-changing. However, completing objectives in general domains requires robots to perceive scenes in diverse environments, plan over multiple sets of tasks, and execute low-level motion plans for each step in a task. For tasks involving dexterous manipulation, existing methods are often limited in their ability to generalize, often resulting in one-off systems restricted to laboratory environments. These limitations are increasingly critical as we face the complexity of common human environments filled with uncertainty—e.g., due to occlusions, physical contact between objects, etc.

Affordances have been proposed as an interface for bridging the gap between high level task planners and low level motion planners. They provide a natural means of expression for describing and reasoning about the robot's environment and the actions available. Affordances can be used as planning operators within existing task planners (e.g. STRIPS [2]).

To apply affordances to manipulation task planning, we require the ability to generate low level motion plans (i.e., paths, trajectories and ultimately motor controls) to execute actions that satisfy affordances. Affordance templates [5] represent an affordance as a set of waypoints corresponding to a sequence of end-effector poses. Actions are enacted by iterating through these waypoints with paths between waypoints being generated by a motion planning method, such as a PRM [13], RRT [17] or RRT* [12].

There are a number of limitations to affordance templates and, more generally, affordances defined as end-effector waypoints. First, they only allow us to control the pose of the end-effector at the waypoints. Such affordance models also provide no guarantees about the position of the rest of the robot or about the motions between waypoints. If any of the waypoints are blocked by obstructions, or outside of the reach of the robot, then the action fails. And Lastly, end-effector waypoint representations are also difficult to set up for complicated motions that have many steps, and consequently require many waypoints.

To address these shortcomings, we introduce the concept of **affordance wayfields**. Similar to the control basis representation of Grupen et al. [24], wayfields are a gradient field representation of affordances that guide generated motions to satisfying configurations. We express wayfields as a set of critical regions that the robot must pass through to perform a task. We use these regions to formulate an objective function mapped over configuration space based on the workspace poses of objects associated with the affordance. Affordance wayfields allow for planning over the entire path of motion of the robot, whereas affordance templates only allow for control at waypoints. Motion properties, such as constraints, can be enforced by giving undesired configurations a high cost. Extending existing methods, such as STOMP [11] and CHOMP [34], our affordance wayfield representation can also account for obstacles through a linear combination of manipulation actions with gradient fields for an arbitrary number of obstacles. Our planner is also advantageous in that it allows us to specify the desired end of the motion as a region (either in workspace or in C-space). In contrast, both STOMP and CHOMP require the user to specify a single configuration as the goal.

Figure 1 illustrates how affordance wayfields can be applied to an affordance action of closing a desk drawer. To close the drawer, the robot needs to move its end-

[1]Department of Electrical Engineering Computer Science, Robotics Institute, University of Michigan, 2260 Hayward Street, Ann Arbor, MI, 48109-2121 {tamcm,ocj}@umich.edu [2]Parasol Laboratory, Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843-3112, amato@cse.tamu.edu

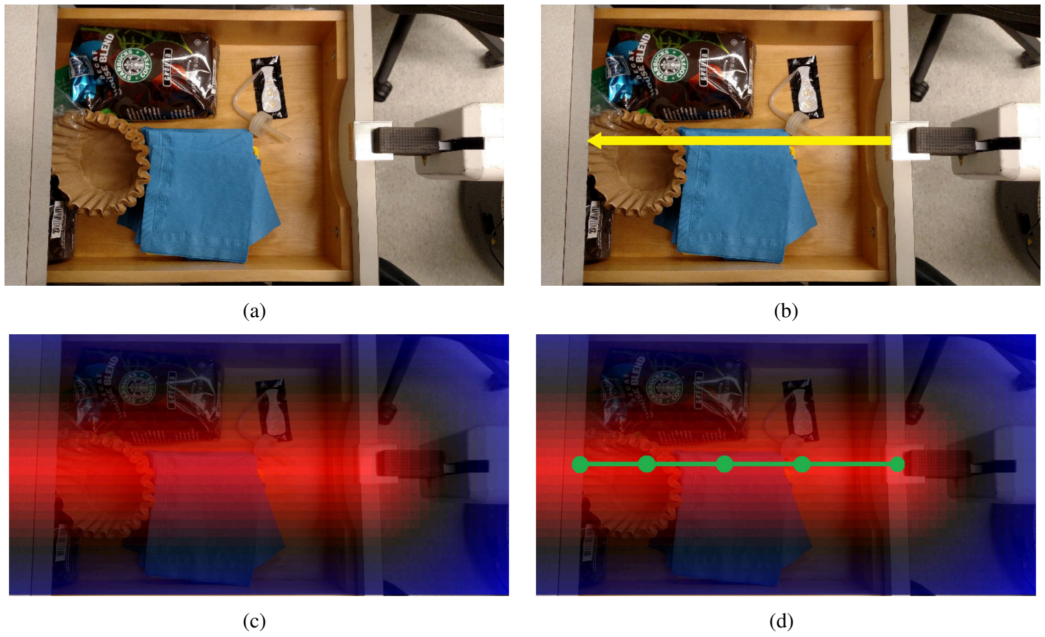


Fig. 1: Affordance wayfield illustration: (a) The robot perceives an open drawer and elects to execute a close affordance on it. (b) The robot needs to move its end-effector horizontally along the yellow arrow. (c) The robot overlays a predefined wayfield associated with closing the drawer on the environment. (d) It then uses gradient decent planning to generate a physical trajectory for performing the affordance. The path of the end-effector during this trajectory is shown in green.

effector horizontally along the yellow arrow (Figure 1(b)). The wayfield for this action is defined by the distance between the robots gripper and the example path (Figure 1(c)), with each configuration of the arm being evaluated based on the position of the end-effector in the field. The wayfield is aligned with the position and orientation of the drawer in the environment, and a configuration space trajectory (Figure 1(d)) is generated using a gradient descent planner.

In the remainder of this paper, we define the concept of an affordance wayfield and its use in task and motion planning for manipulation. We present a framework and implementation for generating robot motion using affordance wayfields. We demonstrate the efficacy of affordance wayfields for manipulation tasks performed by a Michigan Progress Fetch mobile manipulation robot. Our experiments show how affordance wayfields can solve complex, multi-step problems, such as assembling a table or loading and unloading objects from a storage chest.

II. Related Work

Affordances and Affordance Templates: In his seminal work, J.J. Gibson's psychological "theory of affordances" suggests that organisms perceive their environment in terms of their ability to interact with it [4]. Applying this theory to robotics, prior work has examined how to learn affordances for pushing and grasping objects [3], tool use [31], and navigation [20]. More generally, various researchers have modeled affordances probabilistically in terms of the likely effect of robot behavior [27], as planning operators that can be used in extended sequences of manipulation tasks [16], or in collections that can be acquired through intrinsically motivated reinforcement learning [6], [7]. Defining Task goals in terms of affordances also serves as a form of *semantic mapping*. Affordances provide a suitable and compact means of describing a robot's environment that can be understood by a human operator. While semantic mapping has traditionally been investigated in the context of autonomous navigation [18], it provides enormous potential for manipulation systems that must recognize and interact with objects in a 3D environment [26], [23], [8], [32], [33], [21].

Motion Planning for Constrained Systems: Motion planning is the problem of locating valid paths or trajectories for execution of an action in an environment. Sampling based motion planners, such as PRMs [13] and RRTs [17] have been successfully applied to a wide array of problems. Asymptotically optional methods such as PRM* and RRT* [12] converge to shortest paths, or to minimal paths according to a given cost function.

Manipulation planning introduces constraints into the motion planning problem, and many affordance actions specifically require constrained actions. For example, our benchmark task of opening a drawer requires motions where the gripper is constrained to the direction of motion of the drawer. Reachable volumes [19] addresses this problem by computing regions of space the robot's joints must be in to satisfy the constraints, and then sampling the joints in these regions. Reachable volumes have have been applied to a wide variety of problems such as manipulation and planning for high degree of freedom problems. But these methods are limited because their complexity scales with the geometric complexity of constraints, making them unsuited for many of the highly complex constraints encountered in planning for manipulation affordances.

Gradient based motion planners, such as STOMP [11] and CHOMP [34] locate minimal paths within a cost field representation of an environment. They are able to generate collision free paths by representing obstacles using a cost functions such as the one proposed by Schulman et. al. [28]. These methods are also able to generate constrained motions by incorporating the constraints into the cost field. But these methods are limited in that they require an initial path with predefined start and end configurations, which makes them unsuited for problems with multiple potential goals. They also have the potential to converge to local minimums, which makes them very sensitive to the initial path. Unlike our aims for affordance wayfields, STOMP and CHOMP do not directly consider the case of task and motion planning in their formulation.

Combined task and motion planning has received increasing attention in recent robotics research. Srivastava et al. [30] define an interface layer between task and motion planners for abstraction across geometric and configuration spaces. Raman and Kress-Gazit [25] describe an approach for compiling and analyzing task plans as logical descriptions into robot motion. Similarly, Dantam et al. [1] provide probabilistically complete planning algorithm for satisfiability over logical operators. Kaelbling and Lozano-Perez [10] perform hierarchical belief space planning over sets of known operators. Similar to *trajopt* [29], affordance wayfields combines task and motion planning holistically into a single optimization. The resulting planner is a simple extension of gradient descent optimization. For such optimization, affordance wayfields offer a new form of cost function that incorporates notions of object affordances.

III. Affordance Wayfields

We define an affordance wayfield to be a value function over the space of possible paths through configuration space. The value given by this function describes how well the path will perform the affordance, with lower values indicating that the path is closer to the affordance action. Our envisioned use of affordance wayfields takes a similar approach to that of the Affordance Template Library [5]. At run time, the robot

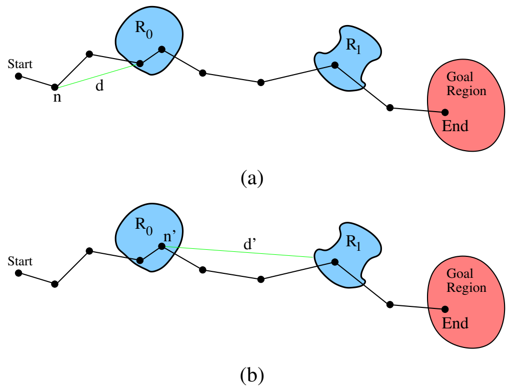


Fig. 2: Wayfield Cost Computation: (a) The cost for node n is the distance, d, between n and the next region, R_0. (b) The cost for node n' is d', which is the distance between n' and R_1.

enacts an affordance by transforming the wayfield onto the existing scene based on the position of the objects associated with the affordance. The robot then computes an obstacle field that encodes collision with obstacles, as well as any other validity constraints. Finally, it computes a motion plan by applying gradient descent planning on the wayfield and obstacle field.

A. Wayfield Formulation

There are many potential ways to define the cost function of an affordance wayfield. We formulate wayfields as an ordered set of regions of C-space that the path must travel through. The cost of each node is the distance between it and the next region that the path travels through, and the cost of a path to be the sum of these costs over all nodes (see Figure 2 and Algorithm 1). This method is beneficial because it can accommodate complex actions that move through the same region of space multiple times.

Our formulation is generic with regards to the geometric representation of the regions. We only assume that the representation provides a means of determining if a node is in a region and a method for computing the distance between a node and a region.

For many applications, only the path of the end-effector is relevant with regards to the affordance. In these cases, it would be more natural to define the regions in workspace and the path cost as the distance of the end-effector to the next region. Similarly, the regions in C-space can be defined from satisfying regions of end-effector poses in workspace. Distance between a node and a region can then be defined as the distance of an end-effector position and the region in workspace. When converting workspace regions to C-space we require all regions to intersect with the robot's reachable volume. If there is no intersection then the robot cannot reach the region and it is not fesiable for the robot to perform

Algorithm 1 PathCost(R,N)

```
input:
$R$: ordered set of wayfield satisfying regions
$N$: ordered set of path nodes in configuration space
output:
$\text{cost}_N$: scalar cost of path
begin
$r = 0$
$\text{cost}_N = 0$
for $n \leftarrow 0 \dots |N|$ do
  $\text{cost}_n \leftarrow$ distance($n$,$R_r$)
  $\text{cost}_N \leftarrow \text{cost}_N + \text{cost}_n$
  if $n$ is inside region $r$ then
    if $r = |R|$ then
      break
    $r \leftarrow r + 1$
return $\text{cost}_N$
end
```

Algorithm 2 GradientDescentPlanner(N)

```
input:
$N$: ordered set of path nodes in configuration space
begin
while $\neg$StoppingConditions($N$) do
  for all $n \in N$ do
    $n$ =GradientDescentUpdateNode($n$)
return $N$
end
```

the required motion. This limitation also exists for affordance templates in that it is inpossiable to generate a motion for a template if the robot cannot reach one of the template's waypoints. Requiring intersections between the regions and the robot's reachable volume also ensures that the regions in workspace will map to non-empty regions in C-space

The waypoint representation used by affordance templates is a special case of our method in which all of the regions are single points (i.e. the waypoints). As with affordance templates we presently require affordance wayfields to be generated by hand. Fortunately, our framework allows for wayfields to be reused such that it is generated only once for a particular affordance.

B. Obstacle Fields

To accommodate obstacles, we generate a potential field [14] to compute obstacle cost. Based on the work of Schulman et al. [9], this field defines the obstacle cost ObstacleCost(n) of a configuration n to be the translational distance required to move the robot to a free configuration. The resulting path will minimize collision with obstacles while adhering as closely as possible to the motion requirements of the affordance. We define the obstacle cost of a path N to be the sum of this cost over all nodes along the path:

$$\text{ObstacleCost}(N) = \sum_{n \in N} ObstacleCost(n) \qquad (1)$$

C. Transitional Cost

The third factor in our model for wayfield planning is the cost of transition between the nodes along the path. For consistency, we use the same cost function as was used by the CHOMP planner [34] as the sum of the transition costs of all adjacent nodes along the path.

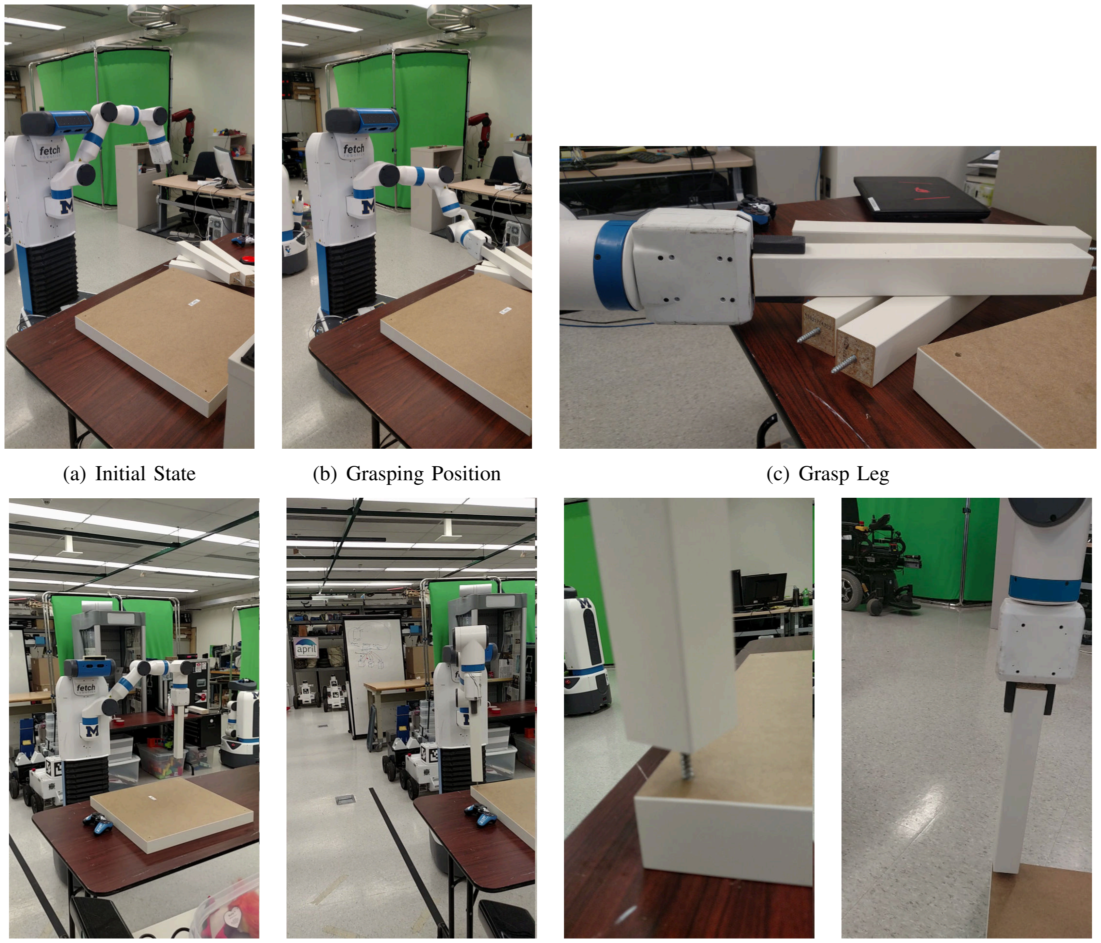
D. Wayfield Matching

To do motion planning with wayfields, we need to match the wayfield to the physical position of the objects the affordance is acting on. For example, in Figure 1 we need to match the "close-drawer" wayfield to the position and orientation of the drawer. The waypoint architecture presented in [5] provides the user with an interface for the user to manually specify the position of these objects, with the waypoints being mapped to the real world based on these positions. We provide an equivalent interface for specifying object positions and map the wayfields to the real world based on these positions. There is ongoing work to automate the process of mapping waypoints and wayfields; however such methods are outside of the scope of this paper.

E. Gradient Descent Planning for Affordance Wayfields

Gradient descent planners (Algorithm 2) start with an initial path that is represented by a sequence of nodes in C-space. Normally, this initial path is a straight line between start and end configurations. Gradient descent is performed on a cost function formed by three terms: 1) the cost given by the wayfield, 2) the cost given by the obstacle field, and 3) the cost of transition between nodes. Our method is different from previous gradient fields in that one of the values, the wayfield cost, is evaluated over the entire path. Previous gradient field formulations primarily use costs that depend only on the position of a node and its predecessor/successor on the path. This assumption makes computing an exact gradient nontrivial. We therefore use an approximate gradient descent (Algorithm 3) in place of an exact method. This method steps a node, n, by randomly generating samples that are near n, then moving n to the position of the sample which gives the lowest total path cost.

One issue we need to consider is the weighting between the three costs. These weightings are denoted as w_P, w_O and w_T in Algorithm 3. We used an increasing weight factor that initially weights the wayfield cost



(a) Initial State (b) Grasping Position (c) Grasp Leg

(d) Lift (e) Position Leg (f) Insert (g) Screw

Fig. 3: Table leg grasp affordance (a,b,c), positioning affordance (d,e,f) and screwing affordance (g)

highly, then gradually increases the weighting of the obstacle and transition costs. We could easily adjust this process based on priorities that are specific to our application.

Our algorithm terminates when one of the following conditions is met: 1) The path goes from having a node in all regions to omitting a region, or 2) The number of iterations reaches some maximum threshold. The first condition specifies that our method terminates when it reaches an iteration where the path goes from having a node in all regions to omitting a region. If this condition is met we stop and return the previous iteration, which includes nodes in all of the regions. Conceptually, this gives us a path that goes through all of the regions while adhering to the obstacle and transition constraints as closely as possible. The second condition guarantees termination by stopping after a certain number of iterations. We use a threshold of 100 iterations in our current implementation.

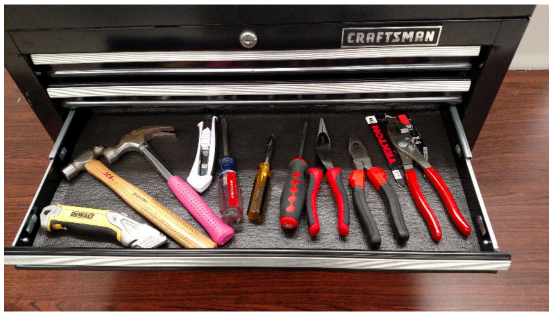
As with other state of the art gradient descent motion planners, ours may converge to a locally optimal solution–the problem of local minimums is inherent to gradient descent planners. Fortunately, in robotic path planning globally optimal solutions are rarely required and a locally optimal solution that accomplishes the required motion planning task is considered adequate. In practice, we found that our planner was able to consistently produce solutions that successfully accomplish the desired actions.

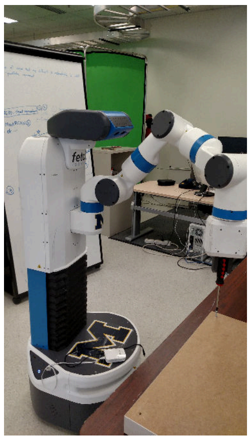
IV. Results

We have implemented a prototype manipulation system demonstrating planning by affordance wayfields. This system was used with a Michigan Progress Fetch robot for scenarios involving tool use, such as table assembly and loading a toolbox. The results presented show the efficacy of affordance wayfields for task and motion planning for a variety of actions beyond pick-and-place. Visualizations of our results are included in this paper's video attachment.

Algorithm 3 ApproximateUpdateNode(N,n)

```
input:
$N$: ordered set of path nodes in configuration space
$n$: node to be update
const:
$D$: number of degrees of freedom in C-space
$\delta$: set of scaling factors over degrees of freedom
begin
$cost_N = w_P$*PathCost($N$)+$w_O$*ObstacleCost($N$)
   +$w_T$*TransitionCost($N$)
for $i \leftarrow 1 \ldots |N|$ do
    $N' \leftarrow N$
    $n'_i \leftarrow n$
    for $j \leftarrow 1 \ldots D$ do
        $n'_i \leftarrow n'_i + \delta_j$*rand(-1,1)
    replace $n$ with $n'_i$ in $N'$
    $cost_{N'} = w_P$*PathCost($N'$)+$w_O$*ObstacleCost($N'$)
       +$w_T$*TransitionCost($N'$)
    if $cost_{N'} < cost_N$ then
        $n \leftarrow n'$
        $N \leftarrow N'$
        $cost_N \leftarrow cost_{N'}$
return $N$
end
```


(a) Tool Chest


(b) Position

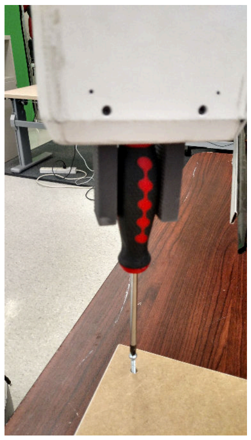

(c) Drive Screw

Fig. 4: Affordances for objects in the drawers.

A. Table Assembly

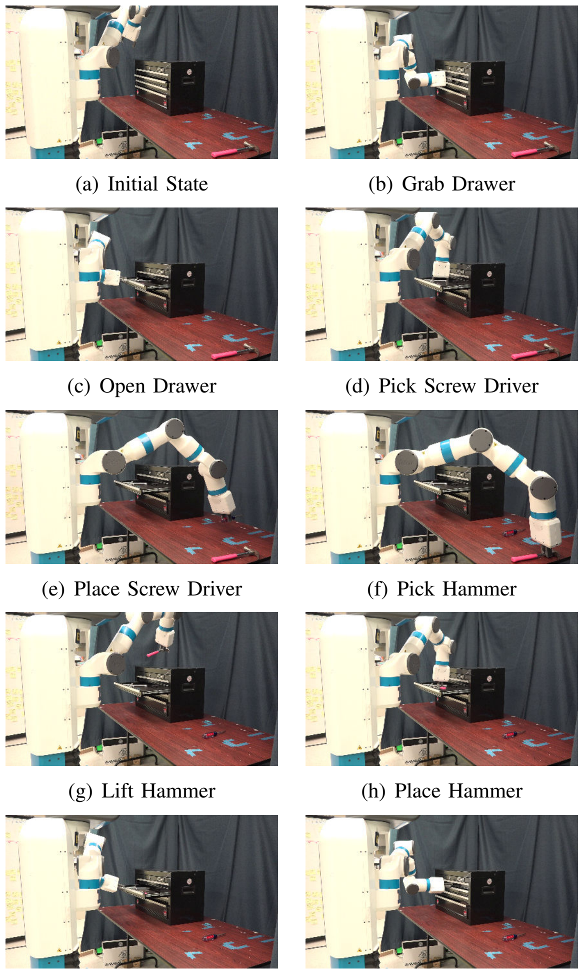
We first apply affordance wayfields to a table assembly task (Figure 3) where the robot needs to pick up each of the table's legs and screw them into the bottom of a tabletop. This problem is analogous to those studied in [22] and [15]. The first wayfield enacts a grasp affordance on the table leg, as pictured in Figures 3(a)-3(c). In order to solve this problem, our grasp planner must be aware of how the afforded action could be executed in the context of a sequential task. For example, it is only possible screw the leg into the base if the robot is grasping it by the end without the screw as shown in Figures 3(c). We define a grasping wayfield that attracts the robot's grippers to the end of the table leg, then closes the grippers. The state of the art affordance templates [5] cannot perform this affordance because they have no mechanism for restricting the robot's orientation. This makes it impossible to require that the robot's hand be aligned with the table leg in order to achieve the required grasp.

We next define a positioning affordance wayfield which positions the table leg over its screw hole, as shown in Figures 3(d)-3(f). This affordance attracts the robots wrist joint to a region directly above the screw hole and the end of the gripper to a position directly below the wrist so that the endpoint of the table leg is just above the position of the screw hole (Figure 3(e)). Again, affordance templates are not able to perform this affordance because they provide no means of restricting the direction of the leg/grasper to be aligned with the screw hole.

We then define a screw affordance wayfield which screws the leg into the tabletop (Figure 3(g)). This wayfield consists of circular motion about the approach axis of the robot's wrist while constraining the other joints to be fixed. As the gripper rotates, the leg is screwed into the base. The affordance template waypoint architecture cannot accommodate a screw affordance because they have no mechanism for specifying a motion that rotates the robot's gripper while constraining the position of the rest of the robot. Performing the grasping, positioning and screwing affordance in sequence will screw one of the table legs into the tabletop. We can therefore assemble the table by applying these affordances to each of the table legs.

B. Toolbox

We next explore a tool chest environment (Figures 4 and 5). This environment consists of a tool chest with multiple drawers that store tools the robot needs to manipulate (Figure 4(a)). As an example, consider a scenario where the robot is tasked with fastening in a screw. To accomplish this task the robot needs



(a) Initial State (b) Grab Drawer (c) Open Drawer (d) Pick Screw Driver (e) Place Screw Driver (f) Pick Hammer (g) Lift Hammer (h) Place Hammer (i) Close Drawer (j) Goal State

Fig. 5: Tool chest environment: The robot performs an open drawer affordance (b,c). It then performs a pick affordance (d) and a place affordance (e) to remove the screw driver from the drawer. Next it performs a pick affordance (e,f) and a place affordance (g,h) to put the hammer into the drawer. Finally, it performs a close drawer affordance(i,j).

perform an open-drawer affordance on the drawer in which the screwdriver is located. It must then perform a pick affordance on the screw driver followed by a positioning and screwing affordance (Figures 4(b) and 4(c)). Finally, it needs to put the screw driver back in the drawer by performing a place affordance followed by a close-drawer affordance.

To accommodate such scenarios, we define a drawer-open affordance and a drawer-close affordance that can be performed on the drawer of the chest. We also define a pick affordance and a place affordance for the objects in the drawer, as well as a set of object specific affordances such a drive-screw affordance for the screw driver.

The drawer-open affordance consists of a wayfield that attracts the robot's grasper towards the drawer's handle (Figure 5(b)) and an open-drawer affordance that moves the gripper outward in order to open the drawer (Figure 5(c)). Because the wayfields are centered on the location of the affordance objects (in this case the drawer) we can use these wayfields to open any of the chest's drawers.

The drawer-close affordance consists of a wayfield that attracts the robot's gripper towards the open drawer (Figure 5(i)) and a second wayfield that moves the gripper towards the chest in order to close the drawer (Figure 5(j)). Unlike the open affordance, the robot does not need to close its grippers before pushing the drawer shut. The existing affordance template architecture only allows the position of the robot to be specified at waypoints. It cannot accommodate open-drawer of close-drawer affordances because it does not provide a mechanism for constraining the robot's motion to be along the axis of the drawer.

The pick affordance (e.g. Figure 5(f)) consists of a wayfield that attracts the end-effector of the robot towards specified grasping positions on the object. Once the robot reaches these positions it closes its gripper, grasping the object. The place affordance consists of an attraction region over the entire inside of the open drawer. This wayfield attracts the robot's grasper to the inside of the drawer where it opens its grippers and release the object. This wayfield demonstrates our method's ability to accommodate regions as goals.

We are also able to define wayfield affordances that are specific for the different objects in the drawer. As an example, we define a drive screw affordance that is associated with the screw driver. This affordance consists of a wayfield that attracts the robot's end-effector towards the position and orientation of a screw followed by a wayfield defining a circular motion similar to that of the table leg screw affordance.

V. Conclusion

This paper presents a novel framework for motion planning with manipulation affordances. It proposes the concept of wayfields which provide template motions for affordances. We show how affordance wayfields can be combined with obstacle fields in order to produce collision free paths. We propose a randomized gradient descent planner to generate paths from affordance wayfields. This planner is advantageous over other gradient descent methods in that it can handle complex paths that traverse the same region of space multiple times. Our experiments demonstrate that affordance wayfields can solve problems such as assembling a table or loading tools into a tool-chest.

References

[1] Neil Dantam, Zachary K. Kingston, Swarat Chaudhuri, and Lydia E. Kavraki. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems*, 2016.

[2] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[3] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning about objects through action: Initial steps towards artificial cognition. In *IEEE International Conference on Robotics and Automation*, Taipei, May 2003.

[4] J. J. Gibson. The theory of affordances. In *Perceiving, acting and knowing: toward an ecological psychology*, pages 67–82, Hillsdale, NJ, 1977. Lawrence Erlbaum Associates Publishers.

[5] S. Hart, P. Dinh, and K. Hambuchen. The Affordance Template ROS Package for Robot Task Programming. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[6] S. Hart and R. Grupen. Intrinsically motivated affordance learning. In *2009 Workshop on Approaches to Sensorimotor Learning on Humanoids at the IEEE Conference on Robots and Automation (ICRA)*, Kobe, Japan, 2009.

[7] S. Hart and R. Grupen. Intrinsically motivated affordance discovery and modeling. In Gianluca Baldassarre and Marco Mirolli, editors, *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 279–300. Springer Berlin Heidelberg, 2013.

[8] Evan Herbst, Peter Henry, and Dieter Fox. Toward online 3-d object segmentation and mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[9] Jonathan Ho Alex Lee Ibrahim Awwal Henry Bradlow Jia Pan Sachin Patil Ken Goldberg Pieter Abbeel John Schulman, Yan Duan. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33:1251–1270, 2014.

[10] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[11] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *International Conference on Robotics and Automation*, Shanghai, China, 05/2011 2011.

[12] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *CoRR*, abs/1105.1186, 2011.

[13] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug 1996.

[14] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[15] Ross A. Knepper, Todd Layton, John Romanishin, and Daniela Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. *2013 IEEE International Conference on Robotics and Automation*, pages 855–862, 2013.

[16] N. Krüger, J. Piater, F. Wörgötter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen, B. Hommel, A. Agostino, D. Kragic, J. Eklundh, V. Kruger, and R. Dillmann. A formal definition of object action complexes and examples at different levels of the process hierarchy. http://www.paco-plus.org, 2009.

[17] J. J. Kuffner and S. M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, San Francisco, California, April 2000.

[18] B. Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[19] T. McMahon, S. Thomas, and N. M. Amato. Sampling-based motion planning with reachable volumes: Theoretical foundations, May 2014.

[20] J. Modayil and B. Kuipers. Autonomous development of a grounded object ontology by a learning robot. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, 2007.

[21] Venkatraman Narayanan and Maxim Likhachev. Perch: Perception via search for multi-object recognition and localization. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016.

[22] Scott Niekum, Sachin Chitta, Andrew G. Barto, Bhaskara Marthi, and Sarah Osentoski. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, 2013.

[23] Richard Alan Peters II, Kimberly A. Hambuchen, and Robert E. Bodenheimer. The sensory ego-sphere: a mediating interface between sensors and cognition. *Autonomous Robots*, 26(1):1–19, 2009.

[24] Robert Platt Jr, Andrew H Fagg, and Roderic A Grupen. Null-space grasp control: theory and experiments. *IEEE Transactions on Robotics*, 26(2):282–295, 2010.

[25] Vasumathi Raman and Hadas Kress-Gazit. Towards minimal explanations of unsynthesizability for high-level robot behaviors, 11 2013.

[26] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robot. Auton. Syst.*, 56(11):927–941, November 2008.

[27] E. Şahin, M. Çakmak, M.R. Doğar, E. Uğur, and G. Üçoluk. To afford of not to afford: A formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 4(15):447–472, 2007.

[28] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *Int. J. Rob. Res.*, 33(9):1251–1270, August 2014.

[29] John Schulman, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization.

[30] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[31] A. Stoytchev. Toward learning the binding affordances of objects: A behavior-grounded approach. In *Proceedings of the AAAI Spring Symposium on Developmental Robotics*, Stanford University, 2005.

[32] Zhiqiang Sui, Lingzhu Xiang, Odest C. Jenkins, and Karthik Desingh. Goal-directed robot manipulation through axiomatic scene estimation. *International Journal of Robotics Research*, 36(1):86–104, 1 2017.

[33] Zhiqiang Sui, Zheming Zhou, Zhen Zeng, and Odest Chadwicke Jenkins. Sum: Sequential scene understanding and manipulation. *IROS*, 2017.

[34] Matthew Zucker, Nathan D. Ratliff, Anca D. Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M. Dellin, J. Andrew Bagnell, and Siddhartha S. Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *I. J. Robotics Res.*, 32:1164–1193, 2013.