2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

Long-Horizon Manipulation of Unknown Objects via Task and Motion Planning with Estimated Affordances

Aidan Curtis* Xiaolin Fang* Leslie Pack Kaelbling Tomás Lozano-Pérez Caelan Reed Garrett†

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9812057

***Abstract*— We present a strategy for designing and building very general robot manipulation systems using a general-purpose task-and-motion planner with both engineered and learned modules that estimate properties and affordances of unknown objects. Such systems are closed-loop policies that map from RGB images, depth images, and robot joint encoder measurements to robot joint position commands. We show that this strategy leads to intelligent behaviors even without a priori knowledge regarding the set of objects, their geometries, and their affordances. We show how these modules can be flexibly composed with robot-centric primitives using the PDDLStream task and motion planning framework. Finally, we demonstrate that this strategy can enable a single policy to perform a wide variety of real-world multi-step manipulation tasks, generalizing over a broad class of objects, arrangements, and goals, without prior knowledge of the environment or re-training.**

I. Introduction

Our objective is to design and build robot policies that can interact robustly and safely with large collections of objects that are only partially observable, where the objects have never been seen before and where achieving the goal may require many coordinated actions, as in putting away all the groceries or collecting all the ingredients for a meal. Our goal is a *single* goal-conditioned policy that will generalize without specialized re-engineering or re-training to a broad range of novel objects, physical environments, and goals. In this paper, we describe a general strategy for designing such systems and a particular implementation called M0M (Manipulation with Zero Models).

The capabilities of M0M are illustrated in Figure 1. The goal is for all objects to be on a green target region. Importantly, *the system has no prior geometric models of objects and no specification of which, or even how many objects, are present in the world.* It takes as input RGB-D images, which it segments and processes to find surface and object candidates. Goals are specified via inputs to the policy using logical formulas; in this example,

$$\forall obj.\ \exists region.\ \texttt{On}(obj, region) \wedge \texttt{Is}(region, \texttt{green}).$$

This formula involves a spatial relation on object volumes, `On`, that the system can plan to achieve, by picking and placing, and perceptual properties such as color, (`green`), that the system can compute from the input images (see Section II). The goal does *not* reference any individual objects by name because, in our problem setting, the object instances have no unique identities and must be referenced by their properties. Instead, goals existentially and universally quantify over the perceived objects, varying substantially in number and properties across or within problem instances.

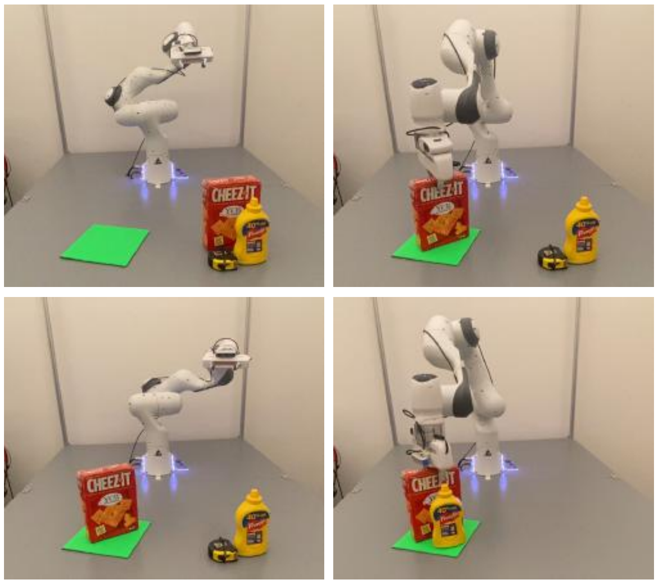

Fig. 1. The robot starts with one object (the cracker box) visible. The goal is for all perceivable objects to be on a green target region. Multiple re-perceiving and re-planning steps are necessary to achieve the goal.

Initially, two objects are hidden behind the tall cracker box so the robot cannot perceive them. Finding only a single object on the table, the robot first picks and places the cracker box on the green target region. After re-observing, the robot discovers two new objects in the scene. To achieve the goal formula, the robot re-plans and intentionally moves the cracker box to a temporary new placement to make room for the tape measure and mustard bottle. Finally, the robot plans a new placement for the cracker box that avoids collisions with the other two objects while also satisfying the goal. A video of this trial and many more can be found at `https://tinyurl.com/manipulation-m0m`.

This example demonstrates several critical properties of intelligent manipulation systems, which our method is the first to integrate: (1) Unlike previous manipulation planning systems, M0M does not require explicit knowledge about the specific objects or even object classes. (2) Unlike many methods for constructing task-specific policies, M0M can achieve any goal that can be represented in first-order logic over

*The first two authors contributed equally and are listed in alphabetical order. The authors are at CSAIL, MIT, USA: {`curtisa, xiaolinf,lpk,tlp,caelan`}`@mit.edu`. † also at NVIDIA. We gratefully acknowledge support from NSF grant 1723381; from AFOSR grant FA9550-17-1-0165; from ONR grant N00014-18-1-2847; from MIT-IBM Watson Lab and from the MIT Quest for Intelligence.

the provided vocabulary of relations. No images, physical demonstrations, or re-training episodes are needed for a new task. (3) M0M reasons about complex interactions between a high-level plan and low-level geometry (*e.g.* obstruction), demonstrating a level of reasoning that goes substantially beyond picking and placing objects at pre-specified poses. (4) Geometrically feasible plans are found under partial observability where objects are only partially visible or not visible at all but previously observed or referenced in the goal. This is accomplished using a combination of shape completion, state estimation, and visibility reasoning. (5) M0M incorporates feedback during plan execution to re-plan when errors occur, or more information about the state is revealed. A persistent memory is maintained across re-plans to better estimate the state and handle situations where objects become occluded.

Our approach leverages the planning capabilities of general-purpose task and motion planning (TAMP) systems [1]. These planners can construct long-horizon manipulation plans to achieve complex goals requiring tightly-knit discrete and geometric decision making. Prior to this work, a common criticism of TAMP was the presumed limitation that it requires known object models [2], [3], [4], [5]. In this paper, we refute this presumption by demonstrating the first TAMP system that does not require known object models. The key insight behind our approach is that such planners do not necessarily need a perfect and complete model of the world, as is often assumed; they only need answers to a set of "affordance queries", which can be answered by direct recourse to perceptual data. The different queries may use different representations of entities such as point clouds, meshes, or image properties, whichever is most appropriate for the particular query. The rest of the paper describes the design, implementation, and evaluation of M0M.

II. Approach

The M0M approach constructs a "most likely" estimate of the world state based on a segmentation of the current scene together with inferences based on the previous observations, actions, and the goal. It then solves for a multi-step motion plan to achieve the goal given that interpretation, executes one or more steps of the plan, re-observes the scene, determines whether the goal is satisfied, and if not, re-plans. This simple approach to handling partial observability is surprisingly effective in many cases but could be generalized to incorporate explicit belief-space planning [6], [7]. In this paper, we take the simpler approach in order to maintain focus on the perception and affordance queries.

The pseudocode for the M0M manipulation policy to achieve a specified goal $\mathcal{G}$ is displayed in Algorithm 1, and a system diagram shown in Figure 2. The policy is given a set of parameterized manipulation actions $\mathcal{A}$ and modules belonging to one of three categories: $\mathcal{M}_{\mathcal{R}}$ modules support robot-centric operations, $\mathcal{M}_{\mathcal{P}}$ modules compute perceptual representations, and $\mathcal{M}_{\mathcal{A}}$ modules compute object-centric affordances. Crucially, the planner only accesses the physical world through these modules (Section II-A).

On each decision-making iteration, the robot receives the current RGB-D image I from its camera and its current joint configuration q from its joint encoders. From each input RGB-D image, it segments out surface point clouds S and object point clouds O. The estimated state is updated based on the segmented objects, surface point clouds, robot configuration, and the previous action and the goal. This current estimated state along with the goal $\mathcal{G}$, actions $\mathcal{A}$, and modules $\mathcal{M}_R \cup \mathcal{M}_A$ form an implicit TAMP planning problem, where access to the state is limited to calls to the specified modules; this is solved by PLAN. This is a very general strategy, applicable to a variety of manipulation domains with different perception and affordance modules; here we specify the modules for a particular instantiation of M0M that we describe in the rest of the paper.

If $\mathcal{G}$ is reachable from the current state s, PLAN will return a plan π, which consists of a finite sequence of instances of the actions in $\mathcal{A}$. If the plan is empty (*i.e.* $\pi = [\ ]$), the current state s satisfies the goal and the policy terminates successfully. Otherwise, the robot executes the first action $\pi[0]$ using its position controllers and repeats this process by reobserving the scene.

Algorithm 1 The M0M policy

```
Assume: $\mathcal{A} = \{\texttt{pick}, \texttt{place}, \texttt{drop}, \texttt{push}\}$
Assume: $\mathcal{M}_R = \{\texttt{inverse-kinematics}, \texttt{plan-motion}\}$
Assume: $\mathcal{M}_P = \{\texttt{segment}, \texttt{shape-complete}, \texttt{mesh}\}$
Assume: $\mathcal{M}_A = \{\texttt{predict-grasps}, \texttt{predict-placement}, \texttt{predict-cfree}, \texttt{detect-attribute}\}$
 1: procedure EXECUTE-POLICY($\mathcal{G}$)                          ▷ Goal $\mathcal{G}$
 2:     $h \leftarrow [\ ]$                                  ▷ Observation and action history
 3:     while True do
 4:         $I, q \leftarrow$ OBSERVE()                  ▷ RGB-D image $I$, robot conf $q$
 5:         $S, O \leftarrow$ PROCESS($I, \mathcal{M}_{\mathcal{P}}$)          ▷ Surfaces $S$, objects $O$
 6:         $s \leftarrow$ UPDATE-STATE($I, S, O, q, h, \mathcal{G}$)
 7:         $\pi \leftarrow$ PLAN($s, \mathcal{G}, \mathcal{A}, \mathcal{M}_R \cup \mathcal{M}_A$)
 8:         if $\pi =$ None then return False ▷ Failure: unreachable
 9:         if $\pi = [\ ]$ then return True              ▷ Success: $s \in \mathcal{G}$
10:         $a \leftarrow \pi[0]$                        ▷ Select first action to execute
11:         EXECUTE-ACTION($\mathcal{R}, a$)
12:         $h \leftarrow h + [\langle I, q, a\rangle]$
```

This policy does not just map most recent observations to actions, which would fail in several common situations. For example, if the task is to have a large box stacked on top of a small box, the small box will be occluded after the action, and thus a state derived from solely the latest observation after stacking would not actually satisfy the goal. In M0M we use inferences based on the previous action history and the goal to update the state estimate. In particular, the system uses the predicted effects of actions to update the state, keeping track of any object that is expected to be occluded. A depth map will be rendered based off the state estimated. If something unexpected happens, leading to a difference between predicted depth and actual depth, data association will be done to match the missing or added points. This strategy enables the system to "track" objects that predictably become invisible as a result of an action (see Figure 3), or accidentally move due to motion failure.

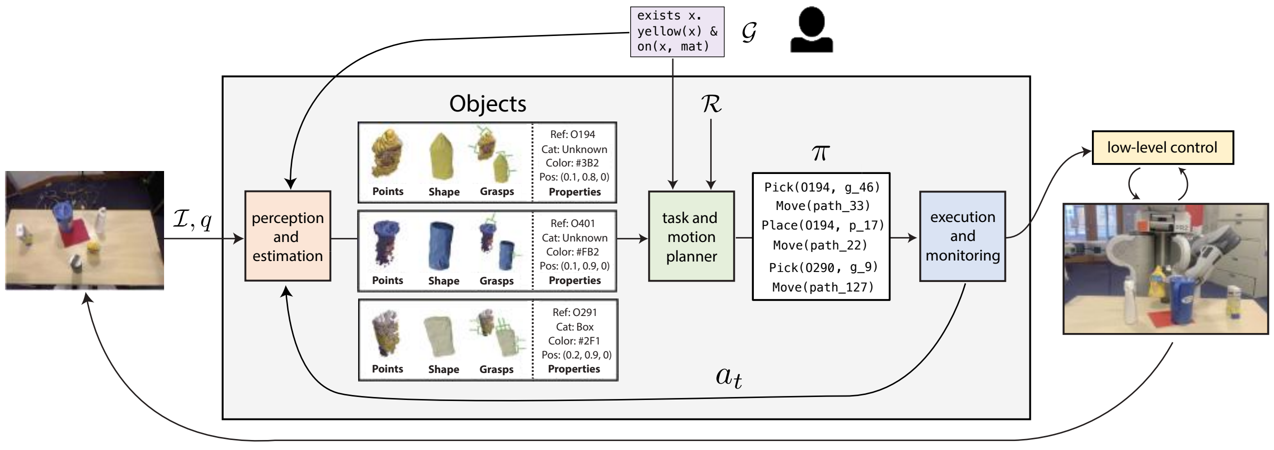

Fig. 2. Structure of the goal-conditioned M0M policy, which maps RGB-D and robot configuration observations to robot position-controller commands.

Additionally, if no visible objects have a property that is needed to satisfy the goal, then planning and the policy would fail. To combat this, the system records which parts of the current scene have not been observed and thus could contain additional objects. Objects with the missing goal properties are optimistically hypothesized to be in the occluded space and are added to the estimated state. However, because their existence is highly uncertain, the planning model contains action preconditions requiring that the volume occupied by hypothesized objects be unoccluded prior to picking them. Thus, the system will plan information-gathering actions to reveal the occluded space. After execution, the system will receive an observation with new information, enabling it to update its hypotheses and re-plan.

A. Modules

The models and modules used by the policy are outlined here, and more concretely described in Section III.

Domain model $\mathcal{A}$: a domain description for planning, that is, a description of the manipulation operations, including a characterization of preconditions and effects of the actions, as well as a description of how the provided modules can be used for picking parameters such as grasps and trajectories. In this paper, we focus on `pick`, `place`, `drop`, and `push` actions, but other operations such as `pour` can be directly incorporated [8], [9]. This single domain description is used for all the objects, arrangements, and goals.

Robot model $\mathcal{R}$: A URDF description of the robot's kinematics that is used for motion planning and simulation and a corresponding position controller for the robot.

Robot-centric modules $\mathcal{M}_R$: These standard robotics modules can be engineered due to the availability of an accurate robot model $\mathcal{R}$: `inverse-kinematics` finds robot configurations given a gripper pose, and `plan-motion` plans collision-free trajectories between two robot configurations using a primitive robot-object collision-prediction module.

Perceptual modules $\mathcal{M}_P$: M0M requires several different modules that produce representations from perception: `segment` takes an RGB-D image as input and produces a set of object hypotheses $O = \{o_1, \ldots, o_{n_o}\}$, each of which is characterized by a partial (colored) point cloud, and a set of approximately horizontal surfaces (such as tables, shelves, parts of the floor) $S = \{s_1, \ldots, s_{n_s}\}$ that could serve as support surfaces for placing objects; `shape-complete` takes an object point cloud and produces a predicted point cloud for o, primarily used for reasoning about collision-avoidance and containment; `mesh` takes an object point cloud as input and produces a volumetric mesh that includes the points. Associated with each entity is an arbitrary reference coordinate frame, such as the sensor's frame. When we speak of the pose of an entity we mean a transform p relative to the reference coordinate frame of o. This notion of a pose is used internally but has no semantics outside of the system.

Affordance modules $\mathcal{M}_A$: These modules predict *action affordances* of objects: ways that the robot can act on or use the objects, and they use different representations for their computations. Some use conservative over-estimates of the input point cloud to find volumes for avoiding collisions, some use tight approximations of local areas to find candidate grasps, while others use learned networks operating on the whole input to compute such affordances. The `predict-grasps` module enumerates a possibly infinite sequence of transforms between the robot's hand and the reference coordinate frame of o such that, if the robot were to reach that relative pose with the gripper open and then close it, it would likely acquire a stable grasp of o. The `predict-placement` module generates stable orientations of o in its reference frame. The `predict-cfree` module predicts whether the robot configurations along a trajectory collide with an object at a given pose. The `detect-attribute` outputs a list of properties of o, used in goal specifications, which can include object class, aspects of shape, color, *etc.*

III. Implementation

We implemented an instance of M0M on both a PR2 and a Franka Panda robot. The perceptual modules all make use of RGB-D images gathered from a Kinect 1 sensor mounted on the PR2's head or a RealSense depth camera on the Panda arm. The `inverse-kinematics` module is compiled from the robot URDF using IKFAST [10], and `plan-motion` uses RRT-CONNECT [11] to find a path in the robot's configuration space that avoids collisions. When running on different robots, only the robot model $\mathcal{R}$ needs to be changed. Below we describe specific module implementations.

PDDLStream: Our implementation of M0M is based on PDDLstream [12], an existing open-source domain-independent planning framework for hybrid domains. PDDLstream has been previously used to solve a rich class of manipulation problems; however, in previous applications, all objects in the environment were assumed to be known exactly. Other TAMP frameworks that provide a similar interface between perceptual operations and the planner through, for example, suggesters [13] could also be used as the basis of an implementation of M0M.

The domain model $\mathcal{A}$ is provided in the form of Planning Domain Definition Language (PDDL) operator descriptions. The modules are provided as a set of *samplers* (referred to as *streams*) that produce candidate values of continuous quantities, including joint configurations, grasps, object placements, and robot motion plans that satisfy the constraints of the problem. The innovation required to support M0M was to develop operator descriptions and streams that do not require knowledge of complete, accurate mesh models and that can operate directly on the output of the perception modules. The design of PDDLstream facilitated the shift from known models to online computed affordances since the streams mediate all access to the continuous current-state estimate. Finally, the modularity of PDDLstream enables independent learned and engineered samplers to be fluidly and automatically composed during planning.

Because the PDDLstream planning engine is responsible for querying the perceptual streams, it will automatically decide online which operations are relevant to the problem and how many generated values are needed, avoiding unnecessary advance computation. As a result, the planner will not perform computationally expensive perceptual operations on images and point clouds to, for example, predict properties and grasps, unless the property has been identified as relevant to the problem.

Goal specifications, even those with quantifiers, can be directly and automatically encoded in a PDDL formulation using *axioms* (*i.e.* derived predicates), which are logical inference rules [14], [15], [12]. Intuitively, an axiom has the same precondition and effect structural form as an action but is automatically derived at each state, enabling PDDLstream to efficiently reason about complex goal conditions involving multiple quantifiers, such as the ones present in M0M.

Segmentation of objects and surfaces: We use category-agnostic segmentation to identify rigid collections of points that collectively move as an object when manipulated. In our experiments, we use UOIS-net-3D [16], a pretrained network that takes RGB-D images as input and returns a segmentation of the scene into an assignment of points to objects. We then remove the points assigned to the table,

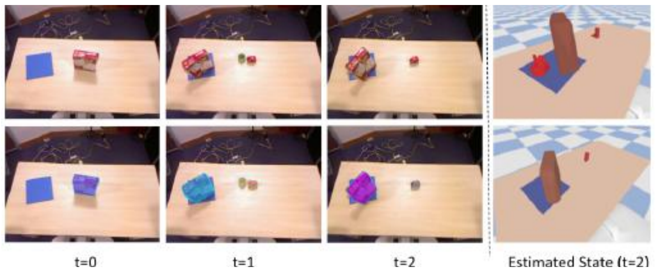

Fig. 3. *Left*: Observed images (top row) and predicted segmentation masks (bottom row) during the system's execution in Figure 5 (task 5). *Right*: The estimated state with (top) and without (bottom) memory. The cup is not detected at t=2 due to occlusion, but M0M is able to keep track of it by updating the state based on the previous state and action executed. See the accompanying video for additional demonstrations of the system's memory.

apply DBSCAN, a distance-based clustering method, to the segmented point cloud produced by UOIS-net-3D in order to reduce under-segmentation and noise in the point cloud, and finally use post-processing to filter out degenerate clusters. We apply RANSAC [17] to the non-object points for table and region estimation. Figure 3 displays the segmentation masks predicted by UOIS-net-3D.

Shape estimation: Both the `predict-placements` and `predict-cfree` stream operations make use of shape estimation, which takes in a partial point cloud as input and predicts a completed volumetric mesh. We again use a combination of learned and geometric methods. For the learned approach, we train MSN [18] on our own synthetic dataset, which is composed of 14 YCB object models, each rendered in PyBullet from approximately 200 randomly-sampled viewpoints, generating the partial point clouds. MSN takes as input a partial point cloud and predicts a completed point cloud. Our geometric method augments the partial point cloud by connecting it to the projection of the visible points onto a table plane. This simple heuristic is motivated by the intuition that the base of an object must be large enough to stably support the visible portion of an object and is particularly useful given a viewpoint that tends to observe objects from above. We filter the result by back-projecting predicted points onto the depth image and pruning any visible points that are closer to the camera than the observed depth value. Figure 4 visualizes the estimated meshes produced by four of the shape-estimation strategies in an uncluttered scene with a diverse set of objects. We use the combination of these methods in the system experiments.

Mesh interpolation: While some downstream operations use the point cloud directly, others require a volumetric mesh. The simplest way to obtain a mesh is to take the convex hull of the points; however, this can substantially overestimate the volume when the object is non-convex, which can lead to planning failures when attempting to grasp non-convex objects such as bowls. Instead, we produce the final volume by computing a "concave hull" in the form of an alpha shape [19], a non-convex generalization of a convex hull, from the union of the visible, network-predicted, and projected points. Collisions along a trajectory are checked

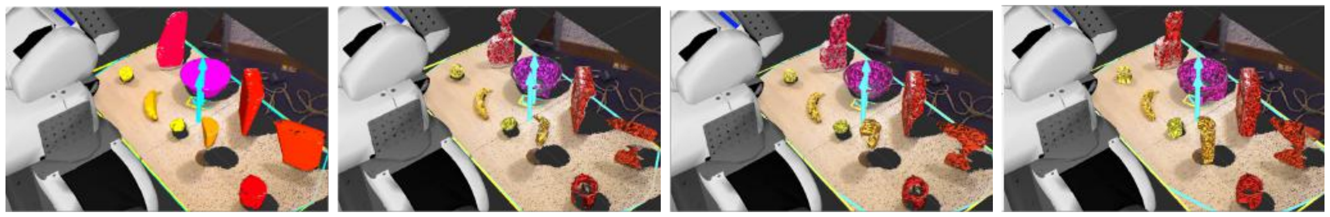

Fig. 4. RViz visualizations of estimated shapes overlaid on top of raw point cloud data. From left to right: a) the convex hull of the visible points only, b) the concave hull of the visible points only, c) the concave hull of the MSN-completed and visibility-filtered visible points, and d) the concave hull of the MSN-completed, projected, and visibility-filtered visible points.

in the `predict-cfree` stream using PyBullet [20]. To enable efficient collision checking, we build an additional representation that approximates the mesh as the union of several convex meshes, implemented by V-HACD [21].

Grasp affordances: Grasp affordances are transformations between the robot's hand and an object's reference frame such that, if the robot's hand were at that pose and closed its fingers, it would acquire the object in a stable grasp. They are purely local and do not take reachability, obstacles, or other constraints into account. We tested three grasping methods for implementing the `predict-grasps` stream, each of which takes a partial point cloud or estimated mesh as input. GPD [22] first generates grasp candidates by aligning one of the robot's fingers to be parallel to an estimated surface in the partial point cloud and then scores these candidates using a convolutional neural network, which is trained on successful grasps for real objects. GraspNet [23] uses a variational autoencoder (VAE) to learn a latent space of grasps that, when conditioned on a partial point cloud, yields grasps. We also developed a method that performs shape estimation using the methods described above and then finds antipodal contact points on the mesh. Qualitatively, we found that GPD and our method outperformed GraspNet both in speed and accuracy, with our method having a slight edge over GPD. Because of this, we use employ our method in M0M for the system experiments in Section IV.

Object properties: The `detect-attribute` stream outputs color properties, based on aggregate color statistics computed from the segmented RGB images and optionally category information from Mask R-CNN [24] trained on our synthetic dataset.

IV. Experiments

In order to evaluate the system's reliability, we experimented with M0M on five challenging manipulation tasks with five repeated trials per fixed goal on a real PR2 robot (see Figure 5). Because the real power of M0M is its generalizability to novel goals, we also applied the system to over 25 individual problem instances, highlighting particular capabilities of the system. See Figure 6 for two examples and `https://tinyurl.com/manipulation-m0m` for full videos of M0M solving all of these problems.

The experimental tasks are characterized below in terms of the logical goal and qualitative properties of the initial state, which the robot can only observe using its sensors:

Task	Iterations	Estimate	Plan	Execute	Success
1	2.0	29.6s	18.5s	16.1s	5/5
2	3.0	34.0s	37.4s	23.6s	5/5
3	2.0	39.6s	41.6s	44.6s	5/5
4	2.4	47.7s	18.9s	28.3s	5/5
5	3.0	36.8s	28.3s	40.5s	4/5

TABLE I
REAL-WORLD FULL-SYSTEM TASK COMPLETION EXPERIMENTS.

1) Manipulating arbitrary objects: A single object is placed arbitrarily on the table and must be placed on a blue region. The five objects we used across the trials were a bowl, a real power drill, a plastic banana, a cup, and a tennis ball.

2) Placing multiple objects: Two large objects (*e.g.* a mustard bottle and a toy drill) must end on a blue region.

3) Obstructed placing: An object with a target property (*e.g.* the most red object) needs to be in the target region. The target region is obstructed by other objects.

4) Obstructed picking: A target object needs to be in the target region, but is surrounded by potentially obstructing objects.

5) Occluded objects: All objects on the table need to be in the target region, but some are initially hidden.

We performed experiments consisting of five repeated real-world trials for the five tasks, obtaining the results shown in Table I. Note that the goal formula is the *only input* that changes for each task. The column *Iterations* refers to the average number of combined estimation and planning iterations that were performed per trial. The system always takes at least two iterations to achieve the goal and then validate that the goal is in fact satisfied. The system performs more than two iterations when the perception module identifies a new object due to under-segmentation, an action is aborted due to a failed grasp, or an action has unanticipated effects.

The columns *Estimation*, *Planning*, and *Execution* report the average time spent perceiving, planning, and executing per iteration. Each module was implemented in Python to flexibly support multiple implementations of each module. During planning, a majority of the time is spent checking for collisions when the robot is planning free-space motions. The overall runtime could be reduced by simultaneously planning motions for later actions while executing earlier actions [7]. The column *Successes* reports the number of times out of five trials that the system terminated having identified that it

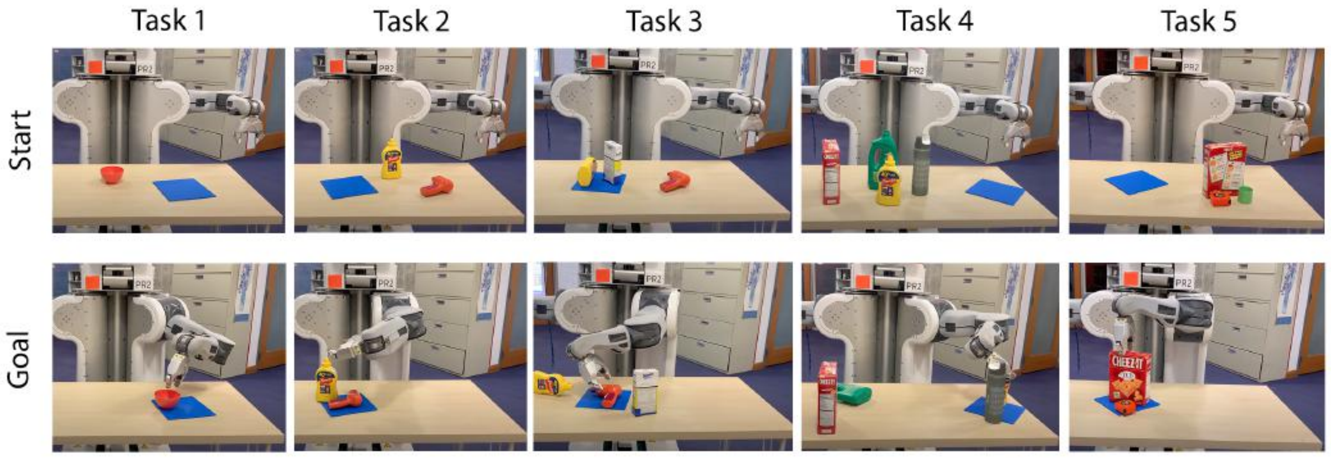

Fig. 5. Visualizations of the initial state and an achieved goal state for one trial per each of the tasks used in the quantitative experiments (Table I).

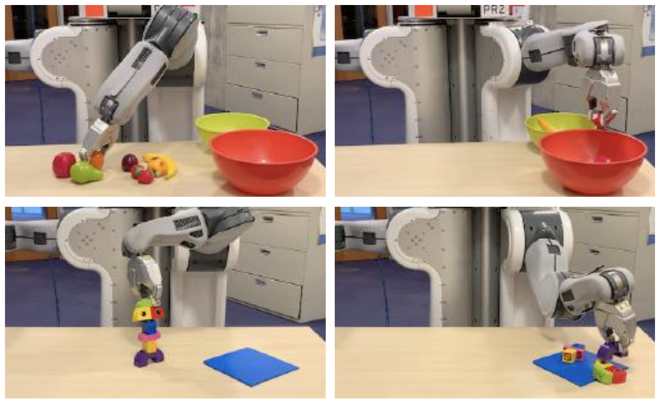

Fig. 6. *Top:* The goal is for each object to be in the bowl that is closest in color to to them. *Bottom:* The goal is for all objects to be on a blue target region. The Lego object is made up of a set of weakly-attached bricks and thus can break when manipulated. When the robot attempts to pick up the Lego, it breaks into three pieces, effectively creating two new objects. Still, the robot is able to perceive these new objects and re-plan to ensure that each one of them ends up on the blue target region.

achieved the goal. Our system was able to achieve the goal on every trial except for a single trial that was a part of *Task 5*, in which the cracker box contacted a tennis ball, causing it to roll out of the reach of the robot. These results show that this single system can perform a diverse set of long-horizon manipulation tasks robustly and reliably.

V. Related Work

Existing TAMP systems that have been demonstrated in real-world settings, including our prior work, require known object instance 3D mesh models that can be accurately aligned to the observed data using human-calibrated fiducials or pose estimators, which restricts their applicability to known environments, often with only a few unique object instances [13], [25], [26], [27], [12], [3]. Even several extensions to TAMP-based systems that actively deal with some uncertainty from perception (such as substantial occlusions) [6], [28], [7] require observations in the form of poses of known objects. This pose registration process is critical for these approaches for identifying human-annotated affordances, such as grasps and placement surfaces and representing collision volumes during planning. In this paper, we show that one can also fulfill these queries using only the observed point clouds, without explicit prior object models.

An alternative strategy for constructing manipulation policies or planning affordances is to learn them via supervised, imitation, or reinforcement-learning methods in simulation or real-world settings [29], [30], [31], [32], [4], [33]. These approaches are attractive because they appear to require less human engineering, but they make heavy demands for simulated or real-world training, posing a substantial development burden. In addition, these learned policies are often narrowly focused on a single "task" and require re-training to deal with even closely related tasks.

The most closely related works to ours[34], [2], [35] involve manipulation planning without prior shape models. Many of the modules of our system, *e.g.*, grasping and shape estimation, are analogous to theirs; the key difference is in the generality of the planners. Gualtieri *et al.* [34] uses a specialized arrangement planner for each task, while Qureshi *et al.* [2] learns a rearrangement policy that converts an input arrangement to a given target one. Neither of these systems can address the more complex goals enabled by a general-purpose task and motion planner.

VI. Conclusion

M0M can perform purposeful manipulation for a general class of unknown objects with arbitrary shapes, arrangements, and goals while operating directly from perceptual data. Importantly, the system is designed in a modular fashion so that different modules, both learned and engineered, can be used for perceptual tasks such as segmenting the scene or choosing grasps on detected objects. Furthermore, new manipulation operations, such as pouring and tossing, can be added and immediately combined with existing operations to achieve new goals.

REFERENCES

[1] C. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Perez, "Integrated task and motion planning," *Annual Review: Control, Robotics, Autonomous Systems*, vol. 4, 2021.

[2] A. H. Qureshi, A. Mousavian, C. Paxton, M. C. Yip, and D. Fox, "Nerp: Neural rearrangement planning for unknown objects," *CoRR*, vol. abs/2106.01352, 2021. [Online]. Available: https://arxiv.org/abs/2106.01352

[3] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs," *ArXiv*, vol. abs/2012.07277, 2020.

[4] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei, "Deep affordance foresight: Planning through what can be done in the future," vol. abs/2011.08424, 2020.

[5] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," *ArXiv*, vol. abs/1912.03628, 2020.

[6] L. Kaelbling and T. Lozano-Perez, "Integrated task and motion planning in belief space," *IJRR*, vol. 32, 2013.

[7] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox, "Online Replanning in Belief Space for Partially Observable Task and Motion Problems," in *ICRA*, 2020.

[8] Z. Wang, C. R. Garrett, L. Kaelbling, and T. Lozano-Perez, "Learning compositional models of robot skills for task and motion planning," *IJRR*, 2020.

[9] R. Holladay, T. Lozano-Pérez, and A. Rodriguez, "Planning for multi-stage forceful manipulation," in *ICRA*, 2021.

[10] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf

[11] J. J. Kuffner Jr. and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[12] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers," in *ICAPS*, 2020.

[13] L. Kaelbling and T. Lozano-Perez, "Hierarchical task and motion planning in the now," *ICRA*, 2011.

[14] E. P. D. Pednault, "ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus." *Kr*, vol. 89, pp. 324–332, 1989.

[15] S. Thiébaux, J. Hoffmann, and B. Nebel, "In defense of PDDL axioms," *Artificial Intelligence*, vol. 168, no. 1-2, pp. 38–69, 2005.

[16] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "Unseen object instance segmentation for robotic environments," in *arXiv:2007.08073*, 2020.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *CACM*, vol. 24, no. 6, 1981.

[18] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, "Morphing and sampling network for dense point cloud completion," in *AAAI*, vol. 34, no. 07, 2020.

[19] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, 1983.

[20] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," 2016. [Online]. Available: http://pybullet.org

[21] M. Müller, N. Chentanez, and T.-Y. Kim, "Real time dynamic fracture with volumetric approximate convex decompositions," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–10, 2013.

[22] M. Gualtieri, A. T. Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," *IROS*, 2016.

[23] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," *CoRR*, vol. abs/1905.10520, 2019.

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[25] 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," *ICRA*, 2014.

[26] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *IJCAI*, 2015.

[27] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. Kavraki, "An incremental constraint-based framework for task and motion planning," *IJRR*, vol. 37, 2018.

[28] D. Hadfield-Menell, E. Groshev, R. Chitnis, and P. Abbeel, "Modular task and motion planning in belief space," *IROS*, 2015.

[29] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *IJRR*, vol. 39, 2020.

[30] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning (CoRL)*, 2019.

[31] A. Hundt, B. Killeen, H. Kwon, C. Paxton, and G. D. Hager, ""good robot!": Efficient reinforcement learning for multi-step visual tasks via reward shaping," *CoRR*, vol. abs/1909.11730, 2019. [Online]. Available: http://arxiv.org/abs/1909.11730

[32] D. Kalashnkov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Mt-opt: Continuous multi-task robotic reinforcement learning at scale," *arXiv*, 2021.

[33] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, "Neural task programming: Learning to generalize across hierarchical tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3795–3802.

[34] M. Gualtieri and R. W. Platt, "Robotic pick-and-place with uncertain object instance segmentation and shape completion," *IEEE Robotics and Automation Letters*, vol. 6, pp. 1753–1760, 2021.

[35] C. Mitash, R. Shome, B. Wen, A. Boularias, and K. Bekris, "Task-driven perception and manipulation for constrained placement of unknown objects," *Robotics and Automation Letters*, vol. 5, 2020.