Learning Robust Manipulation Strategies with Multimodal State Transition Models and Recovery Heuristics

Austin S. Wang[1] and Oliver Kroemer[2]

***Abstract*— Robots are prone to making mistakes when performing manipulation tasks in unstructured environments. Robust policies are thus needed to not only avoid mistakes but also to recover from them. We propose a framework for increasing the robustness of contact-based manipulations by modeling the task structure and optimizing a policy for selecting skills and recovery skills. A multimodal state transition model is acquired based on the contact dynamics of the task and the observed transitions. A policy is then learned from the model using reinforcement learning. The policy is incrementally improved by expanding the action space by generating recovery skills with a heuristic. Evaluations on three simulated manipulation tasks demonstrate the effectiveness of the framework. The robot was able to complete the tasks despite multiple contact state changes and errors encountered, increasing the success rate averaged across the tasks from 70.0% to 95.3%.**

I. INTRODUCTION

Robots operating in unstructured environment will inevitably make mistakes. Mistakes are not desirable, but they are only a major problem if the robot cannot detect and recover from them autonomously. Usually the potential errors that a robot may encounter will not be known a priori, and thus the robot will have to learn strategies for recovering from errors on its own. Learning to recover from errors is however a challenging problem, as the robot needs to consider a variety of different recovery options when learning a robust policy. The robot may be able to continue from the current situation and try to reach the original goals via a different path, or it may need to backtrack an unknown number of steps to reattempt a part of the task. Detecting errors and performing recovery actions will allow the robot to perform the overall task more reliably.

Contacts play an important role in performing manipulation tasks and recovering from errors. Contacts constrain motions and changes in contact states often correspond to subgoals or errors in manipulation tasks, e.g., contacts for grasping or accidental collisions. By monitoring for changes in the contact state, the robot can detect subgoals and errors more reliably. Distinct contact states are modeled as a discrete set of *contact modes*, which will be leveraged in our framework to learn a more accurate representation of the task dynamics.

We propose a framwork that increases the robustness of a policy learned from demonstration. The robot optimizes the policy based on a task representation that was learned from executions of the initial demonstrated skills. As illustrated in Fig. 1, the framework is initialized with a demonstration of the desired task. The robot then repeatedly executes the demonstrated skills to detect different types of errors. The robot uses the resulting data, together with additional contact mode information extracted at the end of each skill, to construct a state transition model. The robot subsequently optimizes the policy based on the generated task model. To further increase the robustness of the policy, the robot utilizes the model structure to generate additional recovery skills over time. The resulting policy maximizes the success rate of the tasks by skipping unnecessary skills and autonomously recovering from errors.

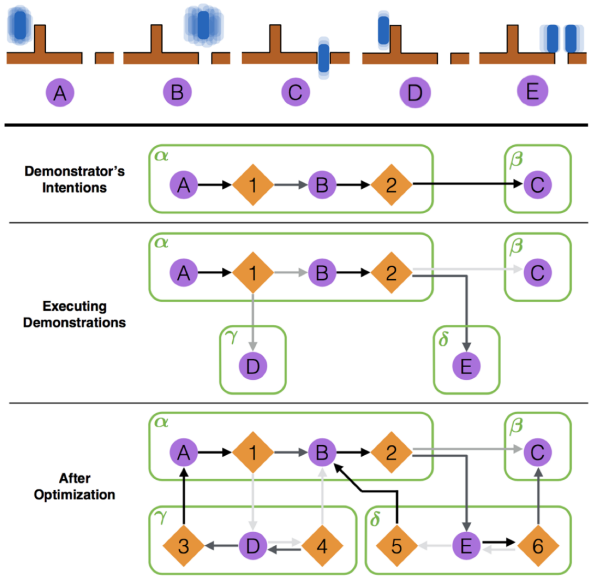

Fig. 1: Illustration of the task structure for an insertion task. The purple circles are distributions of states with similar properties, the orange diamonds are skills, the green blocks are contact modes, and the arrows are transitions and skill selections (darker lines indicate higher probabilities). **(Top)** A demonstration of inserting the of the blue peg into the hole. **(Middle)** Executing the demonstrated skills cause unintended collisions, resulting in additional erroneous contact modes γ and δ. **(Bottom)** Optimizing the policy and generating recovery skills results in additional skills 3 and 6 for transitioning between state distributions to recover from errors.

We evaluated the proposed framework on three simulated manipulation tasks with a Sawyer robot. The experiments

*This work received funding from Amazon through the Amazon Research Award program.

[1]Austin S. Wang is with Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

[2]Oliver Kroemer is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

showed that incorporating the contact modes and recovery skills, instead of simply executing the demonstrated skills as given, lowered the mean task *failure* rate from 30.0% to 4.7%.

II. RELATED WORK

The proposed framework learns a policy for sequencing skills to perform manipulation tasks. Skill chaining was proposed by Konidaris et al. [1] as a framework that starts by learning skills to reach the goal states and then learns additional skills to reach the initiation sets of the current skills. In this manner, the framework grows skill trees from the goal outward. Robots often learn to perform complex tasks using hierarchical policies or sequences of skills. Some approaches assume that the individual skills are provided to the robot [2], [3], while others focus on segmenting and learning skills [4], [5]. The high-level policies for selecting skills have been learned using reinforcement learning [6], [7], [8] and imitation learning [9], [10].

Contact modes are often used to model distinct contact states in manipulation tasks. The term modes is adopted from hybrid systems literature [11], where they correspond to distinct continuous system dynamics that the system can jump between. The modal structure of a task may be learned [12], [13] or predefined. A number of previous planning approaches exploit the contact mode structure of manipulation tasks [14], [15], [16], [17]. This structure can also be used as a basis for learning skills for transitioning between contact modes [12]. Common mode transition skills include grasping and placing. The detection of mode transitions can be used to reduce uncertainty [18], or to determine if a subgoal or error has occurred [13]. In our framework, the robot uses observations of the contact modes as auxiliary information for modeling the dynamics of the task.

The high dimensional state spaces and highly nonlinear dynamics in manipulation tasks have always posed a challenge for planning manipulation strategies. Koval et al. [19] used a lattice in configuration space and performed planning by solving regions relevant to the task online. Other methods, similar to our approach, create simplified task representations by exploiting the structure of contact-rich manipulation tasks, e.g. aggregating similar states together [20], or creating abstract models based on transitions between sets of states [21]. Our approach to creating the state transition model was inspired by [22], which showed that the state space for sequencing skills depends on the pre- and post- conditions of the skills.

The improved robustness of the proposed framework is the result of the robot incorporating additional recovery skills over time. The additional skills allow the robot to recover from errors and skip unnecessary skills in the task execution. Niekum et al. [4] proposed a framework for learning low-level skills and high-level policies for selecting skills from demonstrations. Their framework also allowed the robot to identify errors in the task execution and incorporate additional demonstrations of recovery skills to create a more robust overall policy. Su et al. [13] proposed a method for detecting when a skill failed to reach its goal state. The robot would then reattempt the same skill to recover from the error. Phillips-Grafflin et al. [20] recovered from errors by assuming reversibility and repeating skills multiple times until they succeeded. In the proposed framework, the robot automatically generates additional skills for recovering to different states.

III. STATE TRANSITION MODELS

The goal of the proposed framework is to learn a high-level policy for chaining together low-level skills. The robot will begin by learning a state transition model based on the data collected while attempting the task using the initial demonstrated skills.

We assume a quasi-static environment, i.e. the state of the system can be fully defined by the poses of the arm and the manipulated objects after each skill execution. The task is then modeled as a continuous-state Markov Decision Process (MDP). The states are defined as the poses of the end-effector relative to the manipulated object, and are assumed to be fully observable.

The state transition distributions are only dependent on the current state and the executed actions. However, in the presence of contact mode switches, those distributions are hard to model using the pose information alone, as the effects of skills are strongly affected by the low-level action constraints imposed by the contact mode. A slight shift in pose might be the difference between contact or no contact, thus executing the same skill might have a completely different effect. Hence, contact mode observations will be used to provide additional structure to the transition model.

A. Exploratory Data Generation

To extract the structure of the manipulation task, we begin by providing the robot with a sequence of skills for performing the task. In our evaluations, the skills are defined as straight line Cartesian end-effector movements, and each skill a is parameterized by an end-effector goal pose g relative to the observed object frame. Although one could use more complex skill representations, e.g., DMPs [23], straight line movements are often well suited for low-level skills [13]. Due to stochastic transitions and contact constraints, the end-effector pose after executing a will not be the same as g. The robot will encounter a variety of states as it explores the task, including erroneous states that the demonstrator had not intended as part of the task.

Each state will correspond to a contact mode which will impose a certain set of local dynamics and constraints. The robot can observe these constraints by applying low-level actions and observing the results. Hence, at the start and end of each skill execution, the robot performs a sequence of local 3D perturbations in the end-effector pose Δx_d, which we assume to be reversible, and observes the resulting changes in the end-effector position Δx. Note that Δx_d is the controller input in the form of the change in desired end-effector position. The robot then fits a linear model to this data $\Delta x = A\Delta x_d$. In our experiments, the off-diagonal

elements tended to be close to zero, so the diagonal elements of the matrix A are used as contact mode observations $o^m = \text{diag}(A)$. In the future, we will explore more expressive representations of the local constraints. The auxiliary contact mode observation is then appended to the state to form an augmented state feature $z_t =< s_t, o_t^m >$.

The robot attempts the task multiple times using the initial sequence of demonstrated skills. For each execution of a skill a_t the robot receives an experience in the form of the tuple $e_t =< z_t, a_t, r_t, z_t' >$, with the current augmented state z_t, the skill executed a_t, the reward received r_t, and the next augmented state z_t'. The experiences are stored in a buffer and used to learn state transition models in an offline fashion.

B. Transition Model

Given the exploratory dataset, the next step is to model the state transition distributions for the skills. Our skills have the property of acting as funnels in the state space: given a state within an *input region* of the state space, the skill execution will result in a transition to a state within a corresponding *output region* of the state space. Sequentially selecting skills can thus be seen as chaining together funnels to reach desired states [1]. A funnel i is associated with an action a_i, an input region ϕ_i, and an output region ψ_i. In our model, input regions are represented as unimodal gaussian distributions $\phi_i(z) = \mathcal{N}(z|\mu_i, \Sigma_i)$. Due to the unpredictable nature of hybrid systems (e.g. a pushed bottle may slide or fall over), output regions are modeled as multimodal gaussian mixtures $\psi_i(z') = \sum_j w_j \mathcal{N}(z'|\mu'_{ij}, \Sigma'_{ij})$. The state transition model is then formed in a locally weighted fashion:

$$P(z'|z, a) = \frac{\sum_i \psi_i(z')\phi_i(z)\delta(a_i, a)}{\sum_j \phi_j(z)\delta(a_j, a)}$$

where a_i is the skill associated with the ith funnel, and δ is the Kroenecker delta function that only takes a value of one if the two inputs are the same and zero otherwise. The input and output regions can thus be seen as a model of the skill's preconditions and corresponding postconditions [22].

Contact modes are discrete in nature, and thus an ideal model would be to incorporate them as discrete variables. These discrete mode values would be functions of the continuous state. However, the discontinuous mapping between the states and the contact modes is unknown and hard to learn, which is why the continuous contact mode observations are used in our model. The auxiliary mode observations allow the robot to better distinguish between regions of the state space with different contact constraints. Intuitively, we can decompose the input and output distributions as $\phi_i(z_t) = \phi_i^s(s_t)\phi_i^m(o_t^m)$ and similarly $\psi_i(z_t') = \psi_i^s(s_t')\psi_i^m(o_t^{m\prime})$, where $\phi_i^s(s)$ and $\psi_i^s(s')$ capture the effects of high-level skills and $\phi_i^m(o^m)$ and $\psi_i^m(o^{m\prime})$ capture the effects of low-level actions. Both of these transition types are important for creating a robust transition model.

Given the general form of the transition model, the next step is to extract the input and output regions. The robot learns a model of the regions by clustering the exploratory data points for the individual skills. The components of the transition model are constructed by a clustering of states, similar to the symbolic state generation procedure proposed by Konidaris et al.[22]. The clustering approach for the individual skills is illustrated in Fig. 2 and Algorithm 1. The robot begins by clustering the skill's samples according to the next states z'. The clustering is performed using DBSCAN [24]. The algorithm clusters together any samples z_i, z_j within a distance $D_z(z_i, z_j) = \|s_i - s_j\|_2 + \left\|o_i^m - o_j^m\right\|_2 < \epsilon_s$, where ϵ_s is a length scale hyperparameter with a value of 2. This process results in a set of output clusters $\hat{\psi}$. The samples for each of these output clusters is then clustered again based on the initial states z using DBSCAN. The resulting clusters $\hat{\phi}$ correspond to distinct regions of the state space, and each of these clusters should be associated to one input Gaussian and one output Gaussian. The final step in our clustering approach merges clusters of samples with similar initial state distributions z. In particular, two state distributions ϕ_i, ϕ_j are merged if $D_\phi(\phi_i, \phi_j) < \epsilon_\phi$, where D_ϕ is implemented with the Bhattacharyya distance [25] and $\epsilon_\phi = 0.05$.

The resulting clusters of samples are subsequently used to compute parameters of the input distributions $\phi_i(z)$ using maximum likelihood. To avoid slight mismatches between input distributions ϕ_i across skills, we compare the distance measure D_ϕ between all pairs of input distributions regardless of the associated skill, and redefine the input distributions using their combined samples for any pairs ϕ_i, ϕ_j that satisfies the same merging condition $D_\phi(\phi_i, \phi_j) < \epsilon_\phi$. The multimodal output distributions $\psi_i(z')$ could in theory also be approximated using the clustered samples.

Algorithm 1 Learning Funnels in State Space

Experiences: $< z_t, a_t, r_t, z_t' >$, $t \in \{1, ..., N\}$
Set of skills (action space): A where $a_t \in A \; \forall t$
Distance measure between states: $D_z(z_i, z_j)$
Distance measure between distributions: $D_\phi(\phi_i, \phi_j)$

$\Phi_{\text{result}} \leftarrow \emptyset$
for all $a \in A$ **do**
  $\hat{\Phi} \leftarrow \emptyset$
  // Cluster output states
  $\Psi \leftarrow$ CLUSTER($\{z_i'|a_i == a\}$, D_z)
  for all $\psi \in \Psi$ **do**
    // Cluster respective initial states
    $\hat{\hat{\Phi}} \leftarrow$ CLUSTER($\{z_j|z_j' \in \psi\}$, D_z)
    $\hat{\Phi} \leftarrow \hat{\Phi} \cup \hat{\hat{\Phi}}$
  // Merge similar initial state clusters
  $P \leftarrow$ CLUSTER($\{\hat{\phi}|\hat{\phi} \in \hat{\Phi}\}$, D_ϕ)
  $\Phi_{\text{result}} \leftarrow \Phi_{\text{result}} \cup \{ \bigcup_{\hat{\phi} \in \Phi} \hat{\phi} | \Phi \in P \}$
return Φ_{result}

function CLUSTER(X, D)
  Cluster all $x \in X$ using DBSCAN with $D(x_i, x_j)$
  return $\{Y_1, Y_2, ..., Y_N\}$, where each Y_i is a set containing all x's belonging to the ith cluster

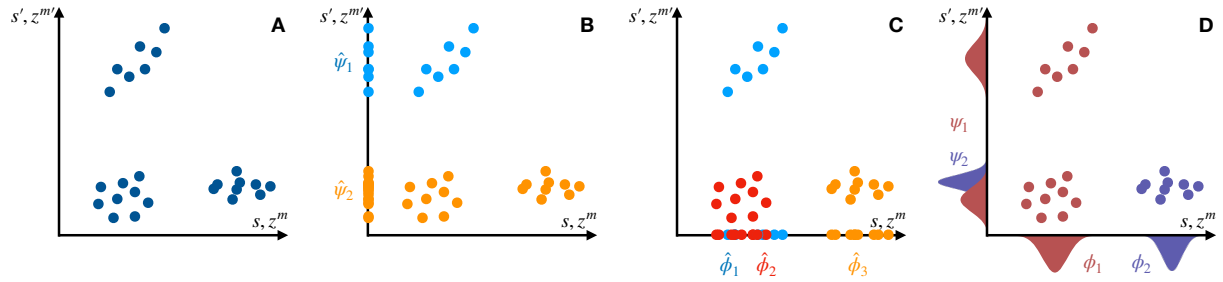

Fig. 2: Conceptual illustration of the clustering process. (**A**) Raw data. (**B**) Cluster output states. (**C**) Cluster initial states corresponding to each output cluster. (**D**) Merge initial states.

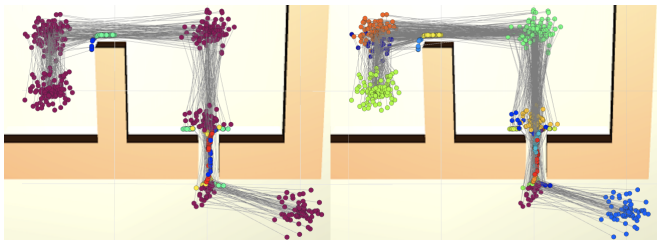

Fig. 3: Visualization of the clustering process. **(Left)** Experience poses and contact modes. Colors indicate different contact modes. **(Right)** Sets of samples resulting from the clustering of distributions. Colors indicate different distribution modes.

However, in practice, these distributions do not need to be computed explicitly to learn the policy, as we will explain in the following section.

An example of the clustering can be seen in Fig. 3 for a basic maze task. The resulting clusters allow the transition distribution to capture both the high-level skill transitions and the low-level constraints corresponding to the different contact modes.

IV. POLICY LEARNING

Given the state transition model, the next step is to learn a policy for selecting skills given the current states and contact mode observations. To further improve the robustness of the policy, we present a heuristic for incorporating additional recovery skills over time.

A. Learning a Policy for Selecting Skills

To learn a policy for selecting skills, we associate an expected reward r_i with each skill a_i and input region ϕ_i pair, which is obtained by taking the weighted average over experiences

$$r_i = \frac{\sum_t r_t \phi_i(z_t)\delta(a_i, a_t)}{\sum_t \phi_i(z_t)\delta(a_i, a_t)}$$

where we use t to iterate over all experiences. We then model the reward function as

$$r(z, a) = \frac{\sum_i r_i \phi_i(z)\delta(a_i, a)}{\sum_j \phi_j(z)\delta(a_j, a)}$$

where we use i and j to iterate over learned funnels.

Using this reward function and state transition model, the action value function can be written as:

$$\begin{aligned} Q(z, a) &= r(z, a) + \gamma \int_{\mathbb{Z}} P(z'|z, a) \max_{a'} Q(z', a') dz' \\ &= \frac{\sum_i q_i \phi_i(z)\delta(a_i, a)}{\sum_j \phi_j(z)\delta(a_j, a)} \end{aligned}$$

where $q_i = r_i + \gamma \int_{\mathbb{Z}} \psi_i(z') \max_{a'} Q(z', a') dz'$. However, the Q-value function parameters q_i cannot be computed analytically because integrating over all possible output states is intractable. We therefore treat the next state z'_t as a sample from $\phi_i(z')$, and compute the parameters q_i using gradient descent with the update rule:

$$q_i \leftarrow q_i + \alpha[Q_t - Q(z_t, a_t)] \left. \frac{\partial}{\partial q_i} Q(z, a) \right|_{z_t, a_t}$$

where

$$Q_t = r_t + \gamma \max_{a'} Q(z'_t, a')$$

and

$$\left. \frac{\partial}{\partial q_i} Q(z, a) \right|_{z_t, a_t} = \frac{\phi_i(z_t)\delta(a_i, a_t)}{\sum_j \phi_j(z_t)\delta(a_j, a_t)}$$

Once the value estimates are computed, a softmax policy $\pi(a|z) = e^{Q(z,a)} / \sum_k e^{Q(z,a_k)}$ is used to select actions while continuing to explore state-action pairs in a structured manner. After every attempt of the task, the model is updated with the new data and value iteration is run until convergence.

B. Generating Recovery Skills

After obtaining a policy for the initial set of skills, the robot will sometimes perform unneccessary or erroneous skill executions. These executions may result in transitions to states that were not seen in the original demonstrations, and the initial skills are therefore also not well-suited for handling these situations. To increase the robustness and performance of the overall policy, the robot will need to acquire additional recovery skills. These skills should be generated in an incremental manner and exploit the prior structure of the task model.

We propose a skill generation heuristic for backtracking to previous states and skipping immediate successor states. These skills will allow the robot to effectively undo the effects of erroneous skill executions. At each skill selection step, the robot attempts to generate and execute a new skill

using the proposed heuristic with a probability of 0.1 in our evaluations.

Our skill generation heuristic is based on the funnel inputs $\phi_i(z_t)$. We define the jth funnel to be a *successor* of the ith funnel if

$$\frac{\sum_t \phi_j(z'_t)\phi_i(z_t)}{\sum_t \sum_k \phi_k(z'_t)\phi_i(z_t)} > h$$

with a threshold $h = 0.1$, which corresponds to the probability of transitioning to z'_t when at z_t marginalized over different skills. Conversely, if the jth funnel is a successor of the ith funnel, then the ith funnel is a *predecessor* of the jth funnel.

Given the current augmented state z_t the current funnel is given by $\arg\max_i \phi_i(z_t)$. The robot then defines a set of candidate funnels that include the current funnel's predecessors and its successors' successors. The predecessor candidates allow the robot to back track one step, while the successors' successors will allow the robot to skip erroneous or redundant actions.

Candidate funnels are removed from the candidate set if the robot already has a skill for transitioning from the current funnel to the candidate one. Candidates are also removed if the value of the current funnel q_i is larger than the discounted value of the candidate q_c, i.e., $q_i > \gamma q_c$. The robot then selects the candidate with the highest value $\tau = \arg\max_c q_c$ and generates a corresponding skill with the goal parameters given by the target funnel's mean $g = \mu(\phi_\tau)$.

As the robot generates new skills, the sets of successors and predecessors will also change and new candidates will be created as a result. For example, if the robot generates a skill for reaching a predecessor, then this funnel will also become part of the successors. The robot will subsequently attempt to create skills for reaching its successor's as part of the successor's successor rule. Similarly, if the robot manages to skip a skill and reach a successor's successor, the robot will then attempt to generate skills for skipping this next skill as well. In this manner, the heuristic allows the robot to incrementally generate skills for improving robustness and performance based on the task structure.

V. EXPERIMENTS

In this section, we explain how we evaluate the proposed approach and discuss the results.

A. Experiment Setup

Three simulated tasks were used to evaluate the performance of the proposed framework using the Mujoco physics engine [26]. First, the robot has to escape a 2D maze as shown in Fig. 1 where the rod end-effector remains at a fixed height above the table. The second task involved a key-shaped end-effector and a T-shaped hole. The goal of the task is to insert the key into the vertical part of the hole and then slide it to the end of the horizontal slot. In the final task, the robot is equipped with a gripper. The robot has to grasp a drawer handle, pull the handle down to release a latch, and then pull the drawer open. The task environments

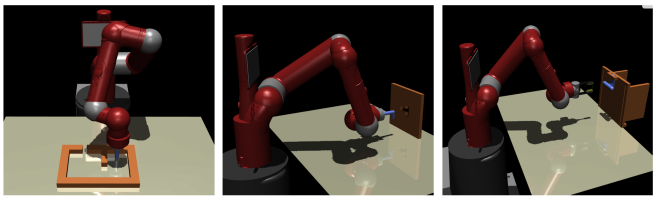

Fig. 4: Task Environments. **(Left)** Maze. **(Middle)** Keyhole. **(Right)** Drawer.

are referred to as Maze, Keyhole, and Drawer respectively, as shown in Fig. 4.

For each of the tasks we define a goal region G: the rod's tip is outside of the maze, the key is at the end of the horizontal slot, and the drawer is more than 10cm open. After each skill execution, the robot receives a reward $r = -d - [s \notin G]\delta t$, where δt is the time duration of the skill, d is the distance travelled during the execution of the skill, and s is the resulting state. A 60-second time limit is also imposed on all tasks, at which point episodes terminate automatically. The arm is controlled using an Cartesian impedance controller. We simulated action noise using a zero-mean Gaussian with a standard deviation of 1cm.

To evaluate the effects of incorporating recovery skills, the robot was initially provided with a sequence of 6, 4, and 6 skills to perform each of the tasks respectively. The robot then generated an MDP model with state transition estimates of the task as described in Section III. The robot was then given an additional 40, 100, and 100 episodes for each of the Maze, Keyhole, and Drawer tasks respectively. During these episodes, the robot was allowed to generate additional skills using the heuristic described in Section IV. We designate checkpoints for evaluation right after learning the initial MDP model and after every 25 episodes of skill generation. At each checkpoint, we evaluate performance by freezing the policy and attempting the task 100 times. No new skills were generated and no additional training data was acquired during these evaluation attempts. The success rates are shown in the top row of Fig. 5. The success rates are averaged over 10 runs of the experiment. Fig. 6 shows the success rates as a function of time.

To evaluate the effects of incorporating the contact modes, we reran the experiment without using the contact mode information z^m in the state clustering process. Results of running the same experiments are on the right of Fig. 5.

B. Discussion

As shown in the left column of Fig. 5, implementing the additional recovery skills increased the mean success rate of the three tasks from 70.0%, 62.0%, and 78.0% to 96.1%, 94.6%, and 95.1% respectively. Perfect success rates were even achieved in at least one experiment run for all three tasks using the recovery skills. Fig. 7 illustrates an episode of the execution of the drawer task in which the arm was able to recover from multiple erroneous states by utilizing newly acquired recovery skills. The importance of contact mode information is emphasized in the right column of

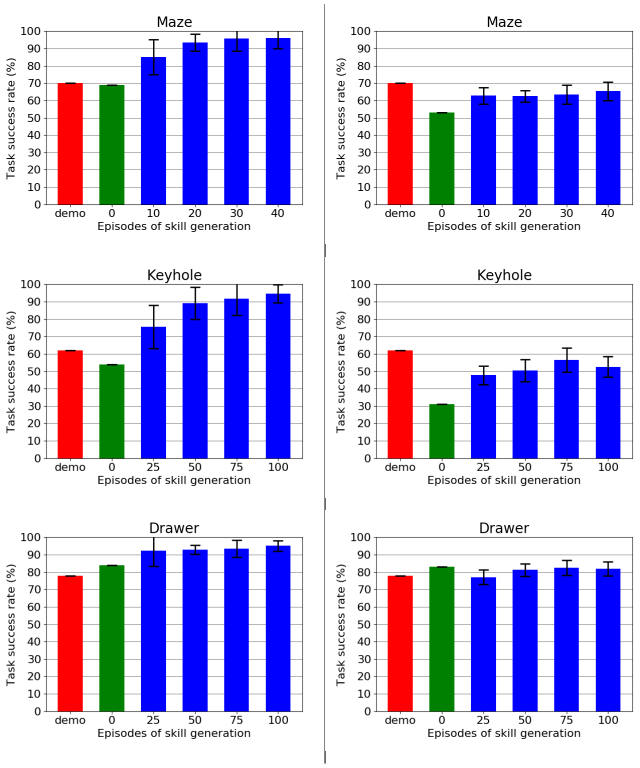

Fig. 5: Success rates for the three simulated manipulation tasks **(Left)** with contact mode information and **(Right)** without contact mode information. The success rate is evaluated over 100 test episodes and averaged over 10 training trials. Results from executing the demonstration skills, using the initial MDP policy without recovery skills, and using the MDP policy augmented with recovery skills are colored as red, green, and blue respectively.

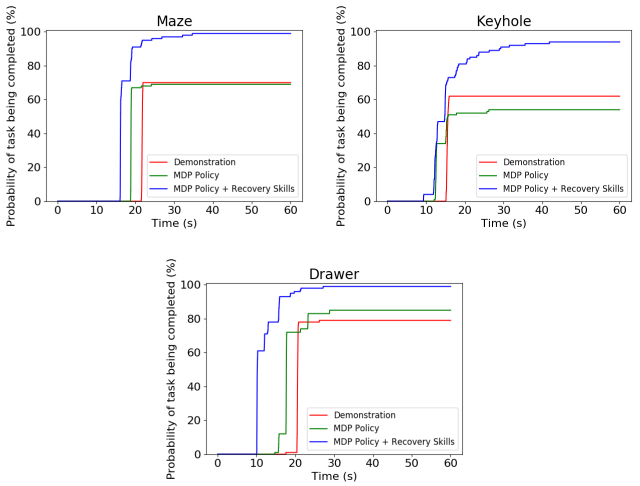

Fig. 6: Cumulative plot of success rate over time. The percentage corresponds to how often the robot solves the task before the given time. The discrete jumps in percentages are due to the fact that skills usually require a constant amount of time to execute, and thus episodes involving the same combination of skills will terminate at approximately the same time.

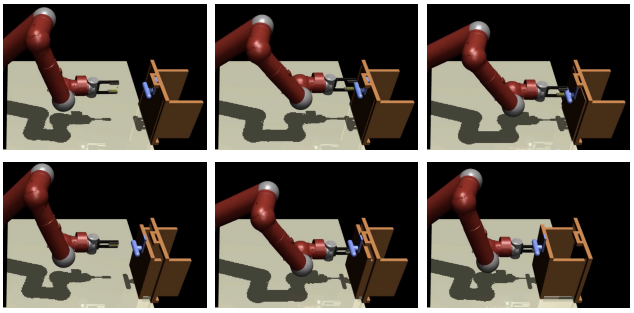

Fig. 7: Visualization of an execution of the drawer task. In the fourth image, the handle slipped away during the process of pulling the drawer out, but since the latch is already out of the socket, the manipulator uses learned recovery skills to swoop back to pull the drawer out without grabbing it.

Fig. 5, where the resulting performance exhibits no obvious improvement over naively executing the demonstration skills.

More insight into the learned strategies can be gained by looking at how long it takes for the manipulator to complete the tasks in Fig. 6. Executing the fixed demonstration skill sequence results in the episodes terminating at the same time regardless of whether the task has been completed. Directly after forming the policy based on the state transition model, the algorithm often finds shortcuts by directly executing the next skill. However, there are times when the shortcut is more risky and leads to a decrease in success rate, as in the Keyhole case. After generating recovery skills, the agent not only discovers more shortcuts and optimizes paths better due to the additional data, but also generates skills to recover from errors, which results in much higher success rates and shorter completion times.

The set of distributions ϕ_i's is currently based on exploring the task with the initial demonstration skills. The performance could therefore potentially be further increased by continuously adapting the state distributions over time as the robot acquires more data.

VI. CONCLUSIONS

We introduced a framework for learning more robust policies for performing contact-based manipulation tasks under uncertainty. An initial set of skills is used to explore the task and subsequently extract a suitable state abstraction based on observed transitions and contact modes. We proposed a heuristic for generating additional recovery skills based on the learned task model. The robot subsequently learned a policy for selecting skills using value iteration. The framework was successfully evaluated on three simulated manipulation tasks and lowered the failure rate averaged over all three tasks from 30.0% to 4.7%.

In the future, we will explore modeling the tasks as Partially Observable Markov Decision Processes (POMDPs) [27] to handle observation uncertainties. We will also explore contact mode estimation methods based on sensory signals during skill executions [13], or by actively perturbing the end-effector to maximize information gain.

REFERENCES

[1] G. Konidaris and A. G. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," in *Advances in Neural Information Processing Systems (NIPS)*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1015–1023.

[2] P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal, "Towards associative skill memories," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012, pp. 309–315.

[3] F. Meier, E. Theodorou, F. Stulp, and S. Schaal, "Movement segmentation using a primitive library," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 3407–3412.

[4] S. Niekum, S. Chitta, A. G. Barto, B. Marthi, and S. Osentoski, "Incremental semantically grounded learning from demonstration," in *Robotics: Science and Systems (RSS)*, 2013.

[5] D. H. Grollman and O. C. Jenkins, "Incremental learning of subtasks from unsegmented demonstration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 261–266.

[6] J. Mugan and B. Kuipers, "Autonomous learning of high-level states and actions in continuous environments," *IEEE Transactions on Autonomous Mental Development (TAMD)*, vol. 4, no. 1, pp. 70–86, March 2012.

[7] C. Daniel, G. Neumann, O. Kroemer, and J. Peters, "Hierarchical relative entropy policy search," *Journal of Machine Learning Research (JMLR)*, vol. 17, no. 93, pp. 1–50, 2016.

[8] B. C. da Silva, G. Baldassarre, G. Konidaris, and A. Barto, "Learning parameterized motor skills on a humanoid robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 5239–5244.

[9] S. Manschitz, J. Kober, M. Gienger, and J. Peters, "Learning to sequence movement primitives from demonstrations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 4414–4421.

[10] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *International Journal Robotics Research (IJRR)*, vol. 31, no. 3, pp. 360–375, Mar. 2012.

[11] A. van der Schaft and J. Schumacher, *An Introduction to Hybrid Dynamical Systems*, ser. Lecture Notes in Control and Information Sciences. Springer, 2000.

[12] O. Kroemer, C. Daniel, G. Neumann, H. van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1503 – 1510.

[13] Z. Su, O. Kroemer, G. Loeb, G. Sukhatme, and S. Schaal, "Learning to switch between sensorimotor primitives using multimodal haptic signals," in *From Animals to Animats 14: 14th International Conference on Simulation of Adaptive Behavior*, vol. 9825, 2016, pp. 170–182.

[14] J. R. Chen and B. J. McCarragher, "Programming by demonstration - constructing task level plans in hybrid dynamic framework," in *IEEE International Conference on Robotics and Automation (ICRA). Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, pp. 1402–1407 vol.2.

[15] G. Lee, T. Lozano-Prez, and L. P. Kaelbling, "Hierarchical planning for multi-contact non-prehensile manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 264–271.

[16] A. Jain and S. Niekum, "Efficient hierarchical robot motion planning under uncertainty and hybrid dynamics," *CoRR*, vol. abs/1802.04205, 2018.

[17] W. R. N. Guan, Charlie; Vega-Brown, "Efficient planning for near-optimal compliant manipulation leveraging environmental contact," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[18] A. Sieverling, C. Eppner, F. Wolff, and O. Brock, "Interleaving motion in contact and in free space for planning under uncertainty," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 4011–4073.

[19] M. C. Koval, D. Hsu, N. S. Pollard, and S. S. Srinivasa, "Configuration lattices for planar contact manipulation under uncertainty," *CoRR*, vol. abs/1605.00169, 2016. [Online]. Available: http://arxiv.org/abs/1605.00169

[20] C. Phillips-Grafflin and D. Berenson, "Planning and resilient execution of policies for manipulation in contact with actuation uncertainty," *CoRR*, vol. abs/1703.10261, 2017. [Online]. Available: http://arxiv.org/abs/1703.10261

[21] K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez, "Grasping pomdps," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 4685–4692.

[22] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[23] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *International Conference on Neural Information Processing Systems (NIPS)*, 2002, pp. 1547–1554.

[24] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96, 1996, pp. 226–231.

[25] T. Jebara, R. Kondor, and A. Howard, "Probability product kernels," *Journal of Machine Learning Research (JMLR)*, vol. 5, pp. 819–844, 2004.

[26] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[27] M. T. J. Spaan, *Partially Observable Markov Decision Processes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 387–414.