2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Efficient Recovery Learning using Model Predictive Meta-Reasoning

Shivam Vats Maxim Likhachev Oliver Kroemer

Abstract— **Operating under real world conditions is challenging due to the possibility of a wide range of failures induced by execution errors and state uncertainty. In relatively benign settings, such failures can be overcome by retrying or executing one of a small number of hand-engineered recovery strategies. By contrast, contact-rich sequential manipulation tasks, like opening doors and assembling furniture, are not amenable to exhaustive hand-engineering. To address this issue, we present a general approach for robustifying manipulation strategies in a sample-efficient manner. Our approach incrementally improves robustness by first discovering the failure modes of the current strategy via exploration in simulation and then learning additional recovery skills to handle these failures. To ensure efficient learning, we propose an online algorithm called Meta-Reasoning for Skill Learning (MetaReSkill) that monitors the progress of all recovery policies during training and allocates training resources to recoveries that are likely to improve the task performance the most. We use our approach to learn recovery skills for door-opening and evaluate them both in simulation and on a real robot with little fine-tuning. Compared to open-loop execution, our experiments show that even a limited amount of recovery learning improves task success substantially from 71% to 92.4% in simulation and from 75% to 90% on a real robot.**

I. Introduction

It is common for robots to make mistakes while attempting a task due to noise in state estimation and actuation. For example, a robot may miss the handle or drop the key while trying to open a door due to an incorrect handle pose estimate. In practice, such mistakes are often handled with hand-engineered or heuristic behaviors and state machines. While practical for relatively simple tasks in controlled environments, this approach cannot scale to systems deployed in the real-world which can fail in a variety of different ways. Hence, there is a need for an algorithmic way to (1) discover potential failures and (2) quickly improve the robot when new failures are discovered.

To this end, we propose an approach to incrementally improve a robot's robustness by discovering potential failures in simulation and learning *recovery skills* that allow the robot to recover. While our approach can robustify against failures due to uncertainty in both execution and state estimation, we focus on the latter as it is more challenging. We assume the robot is given a nominal set of policies which can complete the task under ideal conditions. These could be hand-designed controllers or policies learned from human demonstrations. In reality, the state is rarely known perfectly but is estimated using an online state estimation module. Consequently, the robot may make mistakes during execution and enter a state from which it cannot continue— a *failure state*. We discover such failures in simulation by executing the nominal policies under a simulated state estimation model. Next, we cluster similar failures and learn recovery skills for each of the clusters that allow the robot to recover to the *precondition* of one of its nominal policies.

The Robotics Institute, Carnegie Mellon University
{svats, mlikhach, okroemer} @andrew.cmu.edu

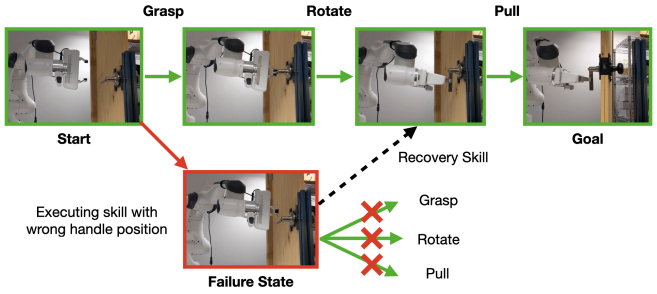

Fig. 1: To open a door, the robot has to (1) grasp the handle (2) rotate it and (3) pull the door. However, it can fail during any of these three stages due to incorrect state information and erroneously enter a failure state from which none of its skills can be applied. We propose an approach that (1) discovers such failures in simulation and (2) uses meta-reasoning to efficiently learn recoveries to the preconditions of the robot's existing skills.

There are multiple potential recoveries from every failure cluster, each corresponding to a precondition the robot could recover to. For n failure clusters and m preconditions, this results in a total of $n \times m$ potential recovery skills. Since attempting to learn all of these recoveries is computationally expensive and redundant, prior works use heuristics to choose where to recover. Recoveries generated in such a way can be sub-optimal as these heuristics don't reason about the quality of the recovery. For example, a common heuristic is to recover to a previous state upon detecting a failure. However, it is preferable to recover closer to the goal in terms of execution cost. On the other hand, it is not known *a priori* if recovering close to the goal is feasible. To this end, we propose **Meta**-**Re**asoning for **Skil**l **L**earning (MetaReSkill), an algorithm that builds a predictive model of improvement in skill performance and decides online which skills to devote training resources to such that the overall task performance improves maximally.

Our main contributions are (1) a hierarchical reinforcement learning-based framework for learning recovery skills to handle failures due to state uncertainty. Compared with open-loop execution, this improves success at opening doors from 71% to 92.4% in simulation and from 75% to 90% on a real robot. (2) a novel meta-reasoning algorithm MetaReSkill for sample-efficient recovery learning. Not only does MetaReSkill improve performance significantly faster than round-robin in all our experiments, but it also achieved the best performance of round-robin using only **70%** of the training budget in $3/5$ of the experiments. Additional details,

experiments and videos are available at https://sites.google.com/view/recoverylearning/home.

II. Related Work

Recovery from Failures: Robotic systems are usually deployed with hand-designed recovery behaviors in the anticipation of failures. Common recovery strategies include retrying the previous step [1], [2], backtracking [3] and hand-designed corrective actions [4], [5]. To execute a recovery, it is important to first detect [6], [7], [8], [9] what kind of failure has happened or is about to happen. Pastor et al. [10] propose Associative Skill Memories which associate stereotypical sensory events with robot movements. Parashar et al. [11] propose an architecture for robot assembly which uses meta-reasoning to identify the cause of a failure and repair the knowledge that caused the failure. In all these works, the recovery behaviors are either manually designed, which limits their scalability, or are generated using a heuristic, which limits their complexity and quality. By contrast, we learn recovery behaviors with reinforcement learning which offers the possibility of learning complex recoveries. Pacheck et al. [12] encode the robot's capabilities in Linear Temporal Logic which allows them to suggest additional skills that would make an infeasible task feasible. However, they do not deal with failures due to state uncertainty. More recently, there has been interest in learning a policy to recover to a safe state [13], [14]. However, these works learn a single recovery policy, assume full observability and do not focus on the efficiency of learning.

Meta-Reasoning: In cognitive science, meta-reasoning [15] refers to processes that monitor the progress of human reasoning activities (*metacognitive monitoring*) and allocate time and effort devoted to cognitive tasks (*metacognitive control*). It is a key component of human intelligence which allows the human mind to solve a wide range of problems using a fixed amount of computation and limited experience [16], [17], [18]. Meta-reasoning has been used in artificial intelligence to design agents that operate in resource constrained environments. One such approach is to compute the value of computation [16] for every computation that could be executed and pursue the computation with the highest value.

Hierarchical Reinforcement Learning: Hierarchical reinforcement learning (HRL) [19] is a powerful approach for difficult long-horizon decision making problems. It enables autonomous decomposition of the problem into tractable subtasks, often building hierarchies of states and policies. A number of prior works propose algorithms for skill discovery and learning [20], [21], [22] with the goal of covering the whole state space By contrast, we seek to discover and cover the part of the state space most relevant to the task. Second, we focus on resource efficiency due to the difficulty and high complexity of skill learning in robotics.

III. Background

Options Framework: We model each robot skill as an option as per the options framework [23], [24]. Each option consists of three components: (a) a robot *control policy* π (b) an *initiation set* which defines the states from which the option can be executed and (c) a *termination condition* which defines the states in which the option must terminate. In continuous spaces, the initiation set is typically estimated using a binary precondition classifier, called the precondition [25]. The skill precondition $\rho(s) : \mathcal{S} \rightarrow [0, 1]$ is a function that returns the probability that the skill can be successfully executed at a given state.

Recovery Skill: A recovery skill is an option that brings the system to a state from which one of its nominal skills can be executed. Formally, let $\Pi = \{\pi_1, \cdots, \pi_k\}$ be the set of nominal skills and let $\{\rho_1, \cdots, \rho_k\}$ be their preconditions. We say that the robot has reached a failure state if none of the preconditions is satisfied. A recovery skill (figure 1) drives the robot to a safe state where at least one of the preconditions is satisfied so that the robot can complete the task.

Acting under Uncertainty: The problem of acting under partial or uncertain state information is optimally solved by formulating it as a Partially Observable Markov Decision Process (POMDP) [26], [27]. A POMDP is defined by the tuple $\langle S, A, T, R, \Omega, O, \gamma \rangle$, where S is the underlying state space and Ω is the observation space. In this formulation, the robot acts on its state belief b, which is a probability distribution over all possible current states. However, solving a POMDP exactly is intractable in manipulation [28]. Instead, we use a common heuristic technique of using a state estimator to maintain a belief over world states based on observations and actions, while the robot acts on the most likely state [29].

IV. Problem Setup

We are interested in solving a manipulation task defined by a distribution of start states $\mathcal{D}$ and a goal indicator function $f_{goal} : \mathcal{S} \rightarrow \{0, 1\}$. The robot incurs costs based on its actions and a penalty of c_{fail} if it ends up in a dead-end or is unable to complete the task in T timesteps. We are given a set of nominal control policies $\{\pi_1, \cdots, \pi_k\}$ and a high level policy Π which chooses among these control policies. We assume that they can reliably complete the task under no uncertainty. Our goal is to improve the robustness of the robot by discovering and learning additional recovery skills that can handle failures due to state uncertainty. Formally, we seek to maximize the expected return of the high-level policy on the task distribution:

$$\mathbb{E}_{\tau \sim \mathcal{D}} \sum_{\substack{t=0, s_0=\tau \\ a_t \sim \pi(s_t), \pi \sim \Pi(s_t)}}^{T} R(s_t, a_t) \tag{1}$$

where $R(s, a) = -c(s, a) - c_{fail} 1_{failure}$ and s_t is the most likely state at time t as determined by a state estimator.

A. Hierarchical Reinforcement Learning Framework

Instead of reasoning with low-level ground states, which are high dimensional, we build a compact symbolic skill graph $G = (V, E)$. Each vertex in this graph is a symbolic state corresponding to a set of continuous states defined by its precondition $\rho : \mathcal{S} \rightarrow \{0, 1\}$. There exists an edge between two vertices $u, v \in V$ if there is a skill whose precondition

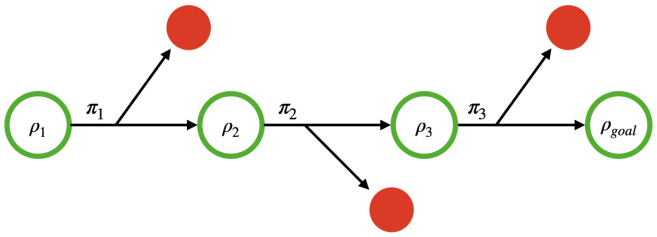

Fig. 2: Failure Discovery: We execute the nominal skills under a simulated state estimator to induce failures (shown in red). These failure states $\in S$ are clustered into failure modes using a Gaussian Mixture Model (GMM). This GMM is used as a failure classifier during execution.

contains u and its effect is contained in v. We initialize this graph as a chain with vertices $V = \{\rho_1, \cdots, \rho_k, \rho_{goal}\}$ corresponding to the preconditions of the nominal skills and edges E corresponding to the nominal policies. Let π_{ij} be the skill from ρ_i to ρ_j and q_{ij} be its probability of success. If π_{ij} fails then we assume it ends up in an absorbing failure state $Fail$ incurring a penalty of c_{fail}. π_{ij} can be learnt using off-the-shelf RL algorithms where ρ_i is the initial state distribution and ρ_j is the goal condition. While we could use ρ_j to define a binary reward function for RL, this is usually impractical for high-dimensional domains. Fortunately, ρ_j can also be used to define a dense reward, for example, by computing the distance to the decision boundary or using the probability $\rho_j(s)$ as the reward. Finally, the high level policy Π for choosing which skill to execute at every symbolic state can be computed using Value Iteration as the skill graph is discrete.

Precondition Chaining: The preconditions of the nominal skills can either be hand-designed or learnt. We use an approach similar to skill chaining [20], [21] to learn the preconditions backwards from the goal. This involves two steps:

1) We collect successful trajectories by executing the nominal policies in simulation. Let $\{s_1, \cdots, s_k, s_{goal}\}$ be one such trajectory consisting of only the start and end states of each policy and s_{goal} is a state that satisfies the goal function. For every policy π_i, we learn a corresponding positive distribution $\mathcal{D}_i^+$ over its start states S_i^+.
2) We train the precondition classifiers backwards from the goal. To learn the precondition ρ_i, we sample states in the vicinity of $\mathcal{D}_i^+$ and execute π_i from there. We verify its success using ρ_{i+1} (ρ_{goal} for ρ_k). This helps us gather informative negative samples S_i^- and additional positive samples which are crucial for learning a tight decision boundary. The precondition classifier ρ_i is trained using S_i^+ and S_i^-.

V. Approach

Our approach consists of two steps- failure discovery and learning recovery skills using meta-reasoning.

A. Failure Discovery

We procedurally generate failure states in simulation by executing the nominal skills under noisy state information. Concretely, let s be the true current state and o be a noisy observation. While the true state is known to us in simulation, we provide only the observation to the nominal skills. Because of the mismatch between o and s, the skill may not work as intended and the robot may end up in a new state s' with observation o'. If none of the existing skills is applicable at s', we record s' as a failure state as the robot would not be able to recover from it even if it could observe the true state. Note that we do not record o' as it may not even be a valid world state. While a recovery for s' does not allow the robot to deal with its current observation o', it will be useful when the robot observes $o \approx s'$. We propose two failure discovery strategies:

1) **Pessimistic Discovery:** The robot executes its nominal policies open-loop under simulated high state uncertainty. This strategy discovers a larger and more diverse set of failures than what may actually be encountered during execution. While this makes recovery learning computationally more expensive, it doesn't require a model of the state estimator.
2) **Early Termination:** The robot executes its nominal policies using observations from a simulated state estimator and terminates if none of the preconditions are satisfied. This strategy discovers a more accurate failure distribution and is preferable if a model of the state estimator is available.

Let S_{fail} be the set of failures discovered. We cluster S_{fail} into n failure clusters $\{\rho_1^f, \cdots, \rho_n^f\}$ and add them as states to our symbolic skill graph. For failures discovered using the early termination strategy, the size of a failure cluster corresponds to the likelihood that the robot will end up in that failure. Both of these failure discovery strategies lead to recoveries that provide significant improvement in performance over heuristic recovery strategies in our experiments.

B. Meta-Reasoning for Skill Learning (MetaReSkill)

The robot may recover to one of the $k+1$ preconditions $\{\rho_1, \cdots, \rho_k, \rho_{goal}\}$ from every failure cluster. However, many of these recoveries are redundant or infeasible. Instead of trying to learn all of them, our algorithm identifies and prioritizes the most promising recoveries.

Value of Failures (VoF): We define the Value of Failures to measure the performance of the robot at its failure states. Consider a failure cluster ρ_i^f in figure 3 at the start of recovery learning. ρ_i^f is not connected to any precondition as the success probability q_{ij} of all the recoveries is 0. Hence, the value of this failure cluster is $V(\rho_i^f) = -c_{fail}$. With further training, the value improves to

$$\max_j q_{ij} V(\rho_j) - (1 - q_{ij}) c_{fail}$$

ρ_j's are high value states as nominal skills can be executed reliably from there. To take into account multiple failure modes, we take a weighted sum of the values of all the

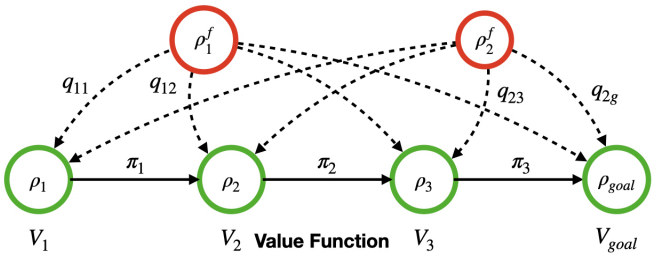

Fig. 3: Optimistic Recovery Learning: We learn recoveries using the most likely state heuristic, i.e., we optimistically assume the state becomes fully observable after the robot ends up in a failure state. We can potentially learn a recovery (shown in dotted lines) to each of the preconditions. At the start of recovery learning, none of these recoveries have been learnt and the robot always incurs a penalty c_{fail} for failing. After some training, the recoveries are partially learnt and have success probabilities q_{ij}. Note that it is preferable to recover to preconditions closer to the goal as they have a higher value than those away from the goal.

failure clusters:

$$VoF = \sum_i \frac{|\rho_i^f|}{\sum_j |\rho_j^f|} V(\rho_i^f) \tag{2}$$

where $|\rho_i^f|$ is the number of ground states in the cluster ρ_i^f.

Intuitively, a high VoF implies that failures during execution are less problematic as the robot is highly confident of recovering. Learning recoveries that optimize the VoF improves robustness to failures. A good first meta-strategy is to train all the recoveries in a round-robin manner as we do not know *a priori* what the best recovery from every failure mode will be. While this works well in the initial stages of learning, we observed in our experiments that it is quite inefficient as the VoF quickly saturates (figure 5). To address this issue, we propose a meta-reasoning algorithm that tracks the progress of all the recoveries and chooses which failure modes to focus on and which precondition to recover to such that the VoF improves maximally with high probability in every training episode.

Model of Task Performance: The key idea of MetaReSkill is to build a model of improvement in task performance by estimating the *rate of improvement* (ROI) of individual skills. We use confidence interval estimation to compute optimistic upper bounds Δq_{ij}^U of the ROI Δq_{ij} of the success probabilities q_{ij} of all the recoveries. This provides an optimistic estimate of how much a recovery could improve after another round of training.

Let θ be a parameter we wish to estimate. An α-confidence interval [30] for θ is an interval (L, U) such that θ is contained in the interval with confidence α. This also implies that U is an upper bound on θ at least with confidence α. Let $\theta_1, \cdots, \theta_w$ be a random sample of the parameter. Under the assumption that the underlying population is normally distributed, the mean μ of the distribution lies in the following interval with probability α:

$$\overline{\theta} - t_{(1-\alpha)/2,w-1} \frac{s}{\sqrt{w}} \leq \mu \leq \overline{\theta} + t_{(1-\alpha)/2,w-1} \frac{s}{\sqrt{w}}$$

where $t_{\alpha,w}$ is the upper α percentage point of the student's t-distribution with w degrees of freedom and s is the standard error.

Algorithm 1 Meta-Reasoning for Skill Learning

procedure TRAIN
  $Q_{ij} \leftarrow$ queue of max size $w, \forall i, j$
  for $1 \leq t \leq B$ **do**
    if all policies have been trained $\geq K$ times **then**
      for all i, j **do**
        $T \leftarrow$ transition matrix of skill graph
        $T(i,j) \leftarrow q_{ij}^U$
        $J_{ij}^U \leftarrow$ OBJECTIVE(T, R)
      $(i^*, j^*) \leftarrow \arg\max J^U$
    else
      $(i^*, j^*) \leftarrow$ least trained policy
    train recovery (i^*, j^*) for η episodes
    $q \leftarrow$ estimate new success rate of $\pi_{i^*j^*}$
    $q_{best} \leftarrow \max(q, q_{i^*j^*})$
    $q_{i^*j^*} \leftarrow q_{best}$, Q_{ij}.insert(q_{best})
    $q_{i^*j^*}^U \leftarrow$ COMPUTEUCL(i^*, j^*)
  return $\pi_{ij}, \forall i, j$
procedure OBJECTIVE(T, R)
  $V \leftarrow$ VALUEITERATION(T, R)
  return $\sum_i \frac{|\rho_i^f|}{\sum_j |\rho_j^f|} V(\rho_i^f)$ ▷ Value of Failures
procedure COMPUTEUCL(i, j)
  $\Delta Q_{ij} \leftarrow$ compute forward differences of Q_{ij}
  $n \leftarrow |\Delta Q_{ij}|$
  $\Delta q_{ij}^U \leftarrow \overline{\Delta Q_{ij}} + t_{(1-\alpha)/2,n-1} \frac{s}{\sqrt{n}}$ ▷ UCL of ROI
  $q_{ij}^U \leftarrow q_{ij} + \Delta q_{ij}^U$ **return** q_{ij}^U

We compute the upper confidence limit (UCL) of Δq_{ij} using only the w most recent forward differences of q_{ij} as the rate of improvement is a non-stationary quantity (w is a domain-dependent hyper-parameter). For every recovery with current success probability q_{ij}, $q_{ij}^U = q_{ij} + \Delta q_{ij}^U$ is an optimistic upper bound on its success probability after an additional round of training. Let J_{ij}^U be the VoF computed by replacing q_{ij} with q_{ij}^U in the transition matrix of the recovery learning graph 3. J_{ij}^U is then an optimistic prediction of the VoF if we were to train π_{ij} for another round. Our algorithm greedily picks a recovery for training that promises the highest VoF in the next round. We initialize MetaReSkill with K rounds of round-robin to estimate the UCL. Priors on Δq_{ij}, if available, can further speed up learning.

VI. EXPERIMENTS

We evaluate our approach on the task of door opening under noisy handle position information both in simulation and in the real world. The goal is to open a door by at least 0.3 rad with the Franka Panda robot under high initial state uncertainty.

Simulation Environment: We adapt the door environment from the MuJoCo-based robosuite [31], [32] framework to match our real door. The world state is 18 dimensional and includes the robot's joint angles and poses of the door and the handle. The initial state uncertainty is sampled from $\mathcal{N}(0, \sigma = 2cm)$ each in the x, y and z positions of the handle.

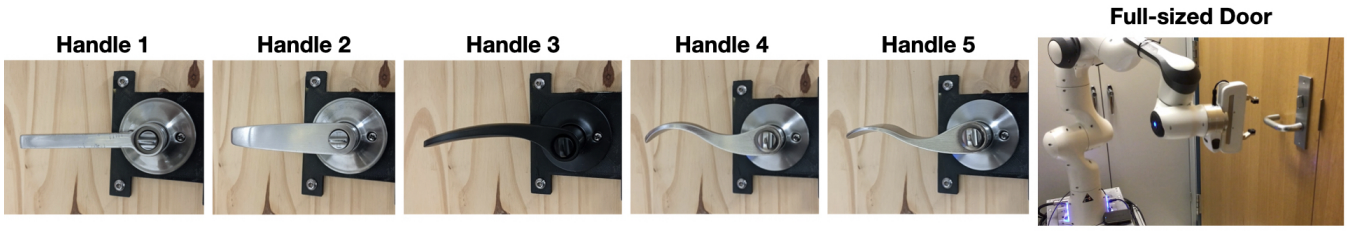

Fig. 4: Handles Used in Evaluation: We compare our approach with open-loop execution on 5 different lever latch handles and a full-sized door with a real robot. Only handle 1 and a small door was used during training; handles 2-5 and the full-sized door are unseen.

	Success Rate (%)	Cost (m)
Recovery-skills (Ours)	**92.4**	0.95 ($\pm$ 0.34)
Retry	66.9	0.80 ($\pm$ 0.02)
Recover-to-prev	75.5	0.86 ($\pm$ 0.19)
Recover-to-start	73.6	0.87 ($\pm$ 0.26)
No-recovery	64.4	0.80 ($\pm$ 0.02)
Open-loop	71.0	0.80 ($\pm$ 0.02)

TABLE I: Simulation results: We compare recovery skills trained with our approach using 150 REPS queries with heuristic recovery strategies. Our approach significantly improves the success rate. The statistics are averaged over 5 sets of recovery skills learnt with different seeds, each evaluated 200 times using a simulated state estimator and a limit of 10 skills per evaluation.

	Open-loop (%)	Recovery (%)
Handle 1 (Train)	75 (15/20)	**90** (18/20)
Handle 2 (Test)	50 (5/10)	**80** (8/10)
Handle 3 (Test)	80 (8/10)	**80** (8/10)
Handle 4 (Test)	80 (8/10)	**90** (9/10)
Handle 5 (Test)	60 (6/10)	**90** (9/10)
Full-sized door (Test)	30 (3/10)	**50** (5/10)

TABLE II: Success Rate on a Real Robot: Learnt recovery skills significantly outperform open-loop execution on a real robot across 5 different handles. Open-loop fails almost half the time on handles 2 and 5 which are, respectively, the smallest and the thinnest of the 5 handles. By contrast, the learnt recoveries use a caging grasp to re-grasp the handle close to the handle's axis of rotation and are robust to these variations. We also test recovery skills on a full sized door where the success rate of our approach drops to 50% due to the slip-prone cylindrical handle of the door.

We design 3 open-loop skills to be executed in sequence - REACHANDGRASPHANDLE, ROTATEHANDLE and PULLHANDLE- as the nominal skills. Each skill consists of one or more 7D waypoints that the robot tries to reach using task space impedance control, where each target consists of a gripper open/close state and a 6D end-effector pose. These skills are able reliably to open doors in simulation and the real world if accurate state information is available.

Symbolic Skill Graph: We train the preconditions of the nominal skills by precondition chaining using a total of 1223 positive and negative samples. Each precondition is a generative classifier with the positive distribution $\mathcal{D}^+$ learnt as a Gaussian distribution and the negative distribution $\mathcal{D}^-$ learnt as a Gaussian Mixture Model. The 3 nominal skills result in 4 symbols for the start, subgoals, and the goal.

Recovery Skill: Each recovery skill π_{ij} is a parameterized skill [33] that uses a regression model to predict robot actions $\theta \in \Theta$ based on the start state s. We use k-nearest neighbours regression to predict a 21D vector, a sequence of three 6D poses with respect to the initial end-effector pose and gripper open/close states, as robot actions. Collecting data of the form (s, θ) for training this regression model involves sampling a start state s from the failure mode ρ_i^f and computing the robot action parameters θ for recovery to ρ_j using Relative Entropy Policy Search (REPS) [34]. For learning a recovery to precondition ρ, REPS uses the reward function $R(s) = 0.1 \log f_{\mathcal{D}^+}(s) + 10\rho(s)$, where $f_{\mathcal{D}^+}$ is the probability density function of the corresponding $\mathcal{D}^+$. Each REPS query takes about 2 minutes to solve on our Intel® Core™ i7-9700K CPU.

A. Evaluation of Learnt Recovery Skills

We first evaluate the effectiveness of our overall approach using the pessimistic failure discovery strategy we described earlier. We execute the nominal skills 1000 times for failure discovery to collect a total of 1400 failure states which we group into 6 clusters using the Gaussian Mixture Model (GMM) [35]. Common failure modes include the robot missing the handle and the robot slipping while pulling the handle due to an improper grasp. We learn recovery skills in simulation using a budget of just 150 REPS queries. With 24 potential recovery skills, this means that each recovery policy can get only 6 data-points on average.

Evaluation in Simulation: We simulate a state estimator by assuming that the standard deviation of the noise distribution halves after every robot action. As we show in table I, learnt recoveries are significantly better than heuristic recovery strategies in improving success rate. Recovering from failures during door opening often requires the robot to (1) carefully move the handle so as not to weaken the grasp and (2) avoid collision with the environment. Heuristic recoveries are unable to account for this and hence perform poorly. Compared to open-loop execution, our approach substantially improves task success rate from 71% to 92.4%. This indicates that (1) the failures discovered using our pessimistic failure discovery do cover a number of failures encountered by the robot when using a state estimator and (2) recoveries can be reliably learnt with a generic reward function defined using preconditions which promises better scalability than reward shaping.

Evaluation on a Real Robot: We transfer the preconditions, failure classifier, nominal skills and recovery skills learnt in simulation to a real Franka Panda robot and run experiments with (a) 5 different lever latch handles on a

small door and (b) a full-sized door in our building with a cylindrical handle (figure 4). We fine-tune only the gains of the impedance controller on the real robot by increasing their values to achieve similar tracking as in simulation. As in simulation, the robot is controlled by a Cartesian-space impedance controller that executes each skill open-loop. We evaluate the recovery skills learnt in simulation under an *idealized state estimator*. The ground-truth handle position is known to us but at $T = 0$, we only provide a noisy handle position estimate to the robot, where, noise $\sim \mathcal{N}(0, \sigma = 2cm)$. The robot executes its nominal skill using this noisy state information. At $T = 1$, by the time the robot finishes executing the first skill, we assume that the state estimator has converged to the ground-truth. Hence, the robot has access to the accurate handle position at this point. The robot uses its preconditions to check if any of its nominal skills can be executed. If so, it executes the remaining nominal skills. If not, it uses the failure classifier to identify the failure mode and execute the best recovery from that mode.

We compare our approach (RECOVERY) with open-loop execution (OPEN-LOOP) of nominal skills (table II). The success rate of OPEN-LOOP is sensitive to the handle and varies from $50 - 80\%$ By contrast, RECOVERY improves the success rate to $80 - 90\%$ consistently across all of the 5 handles even though it was trained only for handle 1. Both OPEN-LOOP and RECOVERY struggle at the full-sized door due to the slippery cylindrical handle. However, our approach still does significantly better than OPEN-LOOP which only succeeded in $3/10$ attempts. We expect recovery performance to improve with further training and by the use of a good state estimator which will enable closed-loop behavior. Importantly, our approach did not induce any additional failures which indicates good transfer of the preconditions and failure classifier learnt in simulation.

B. Evaluation of MetaReSkill

In this evaluation, we assume that we have access to an accurate model of the state estimator so that we can estimate the failure distribution accurately. We discover failures using our early termination discovery strategy along with a simulated state estimator that halves the standard deviation of the noise distribution after every robot action. We discover a total of 2000 failure states which we group into 5 clusters using the GMM. These failures are less diverse than the failures discovered by pessimistic discovery and are more concentrated near the door handle. We compare round-robin learning of recoveries with MetaReSkill in figure 6 using 5 different seeds. We use $\alpha = 0.95$ to compute the confidence interval, window size w of 3 and query REPS for a new data-point in every round, i.e. $\eta = 1$. We initialize the UCL estimates by training every recovery twice in a round-robin order. Not only does MetaReSkill improve significantly faster than round-robin, but also it converges to a better objective in all the trials. In $3/5$ trials, MetaReSkill used only **70%** of the training budget to achieve the best objective achieved by round-robin, i.e., 1 hour earlier. This shows that it can make better use of training resources to improve robustness.

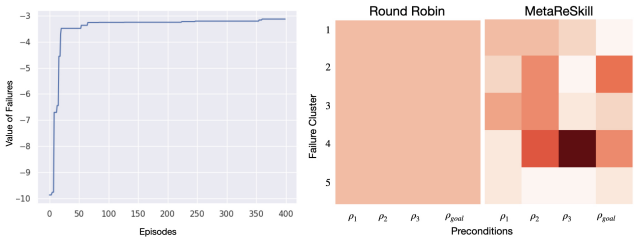

Fig. 5: *(left)* **VoF Saturates:** After quick initial improvement, the objective saturates when recoveries are trained in a round-robin order. *(right)* **Allocation:** We show how many rounds each of the 20 recoveries were trained for by Round-robin and MetaReSkill. Each recovery is identified by the pair (i, j), where, i is the failure cluster it is meant for and j is the precondition it recovers to. Round-robin trains all of them equally while MetaReSkill prioritizes a small number of promising recoveries that improve the VoF by the most.

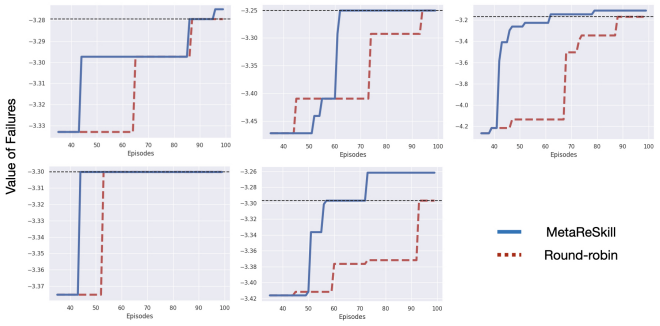

Fig. 6: We compare MetaReSkill with round-robin learning of recoveries over 100 REPS training episodes with 5 different seeds. MetaReSkill is initialized by training each recovery twice initially in a round-robin manner. Hence, we see a difference in performance from episode 40 when MetaReSkill kicks in. MetaReSkill immediately focuses on the most promising recoveries which allows it to optimize the VoF significantly faster, also converging to a better VoF in every trial.

VII. CONCLUSION

We propose a scalable algorithmic framework to efficiently robustify a given manipulation strategy against failures due to state uncertainty. Our method consists of discovering failures in simulation by evaluating the given strategy under simulated state uncertainty and then using meta-reasoning to efficiently train recovery skills. Our algorithm Meta-Reasoning for Recovery Learning (MetaReSkill) monitors the progress of all potential recovery skills during training and adaptively chooses which recoveries to train to best improve the robot's robustness. Compared to baselines, the learnt recovery skills significantly improve task success both in simulation and in the real world. We also find that MetaReSkill makes a much better use of computation than round-robin skill learning. In our future work, we would like to combine failure discovery in the real world with discovery in simulation to further improve robustness. Finally, we are also interested in investigating if we can provide theoretical bounds on the performance of MetaReSkill.

ACKNOWLEDGMENTS

This work was in part supported by ARL grant W911NF-18-2-0218 and ONR grant N00014-18-1-2775.

REFERENCES

[1] F. Ebert, S. Dasari, A. X. Lee, S. Levine, and C. Finn, "Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 983–993.

[2] T. Matsuoka, T. Hasegawa, and K. Honda, "A dexterous manipulation system with error detection and recovery by a multi-fingered robotic hand," in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, vol. 1. IEEE, 1999, pp. 418–423.

[3] A. S. Wang and O. Kroemer, "Learning robust manipulation strategies with multimodal state transition models and recovery heuristics," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 1309–1315, 2019.

[4] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, "Contact-reactive grasping of objects with partial shape information," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 1228–1235.

[5] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, J. Ichnowski, E. Novoseller, M. Hwang, M. Laskey, J. E. Gonzalez, and K. Goldberg, "Untangling dense non-planar knots by learning manipulation features and recovery policies," in *Robotics Science and Systems (RSS)*, 2021.

[6] A. Rodriguez, D. Bourne, M. Mason, G. F. Rossano, and J. Wang, "Failure detection in assembly: Force signature analysis," in *2010 IEEE International Conference on Automation Science and Engineering*. IEEE, 2010, pp. 210–215.

[7] D. Park, Z. Erickson, T. Bhattacharjee, and C. C. Kemp, "Multimodal execution monitoring for anomaly detection during robot manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 407–414.

[8] S. Luo, H. Wu, S. Duan, Y. Lin, and J. Rojas, "Endowing Robots with Longer-term Autonomy by Recovering from External Disturbances in Manipulation Through Grounded Anomaly Classification and Recovery Policies," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 101, no. 3, 2021.

[9] P. A. Zachares, M. A. Lee, W. Lian, and J. Bohg, "Interpreting contact interactions to overcome failure in robot assembly tasks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3410–3417.

[10] P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal, "Towards associative skill memories," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 309–315.

[11] P. Parashar and A. K. Goel, "Meta-reasoning in assembly robots," *Systems Engineering and Artificial Intelligence*, pp. 425–449, 2021.

[12] A. Pacheck, G. Konidaris, and H. Kress-Gazit, "Automatic encoding and repair of reactive high-level tasks with learned abstract representations," in *Robotics Research: the 18th Annual Symposium*, 2019.

[13] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.

[14] A. Wilcox, A. Balakrishna, B. Thananjeyan, J. E. Gonzalez, and K. Goldberg, "Ls3: Latent space safe sets for long-horizon visuomotor control of sparse reward iterative tasks," in *Conference on Robot Learning*. PMLR, 2022, pp. 959–969.

[15] R. Ackerman and V. A. Thompson, "Meta-reasoning: Monitoring and control of thinking and reasoning," *Trends in cognitive sciences*, vol. 21, no. 8, pp. 607–617, 2017.

[16] S. Russell and E. Wefald, "Principles of metareasoning," *Artificial intelligence*, vol. 49, no. 1-3, pp. 361–395, 1991.

[17] M. M. Cox and A. Raja, "Metareasoning - thinking about thinking," in *Metareasoning*, 2011.

[18] T. L. Griffiths, F. Callaway, M. Chang, E. Grant, and F. Lieder, "Doing more with less: meta-reasoning and meta-learning in humans and machines," *Current Opinion in Behavioral Sciences*, vol. 29, pp. 24–30, 2019.

[19] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–35, 2021.

[20] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *Advances in neural information processing systems*, vol. 22, pp. 1015–1023, 2009.

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

[22] A. Bagaria and G. Konidaris, "Option discovery using deep skill chaining," in *International Conference on Learning Representations*, 2019.

[23] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[24] 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.

[25] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Symbol acquisition for probabilistic high-level planning," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[26] K. J. Åström, "Optimal control of markov processes with incomplete state information," *Journal of mathematical analysis and applications*, vol. 10, no. 1, pp. 174–205, 1965.

[27] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[28] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[29] M. T. Spaan, "Partially observable markov decision processes," in *Reinforcement Learning*. Springer, 2012, pp. 387–414.

[30] W. W. Hines, D. C. Montgomery, and D. M. G. C. M. Borror, *Probability and statistics in engineering*. John Wiley & Sons, 2008.

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

[32] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[33] B. Da Silva, G. Konidaris, and A. Barto, "Learning parameterized skills," *arXiv preprint arXiv:1206.6398*, 2012.

[34] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, 2010.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.