arXiv:2305.02807v1 [cs.RO] 4 May 2023

Learning Failure Prevention Skills for Safe Robot Manipulation

Abdullah Cihan Ak¹, Eren Erdal Aksoy² and Sanem Sariel¹

Abstract-Robots are more capable of achieving manipulation tasks for everyday activities than before. But the safety of manipulation skills that robots employ is still an open problem. Considering all possible failures during skill learning increases the complexity of the process and restrains learning an optimal policy. Beyond that, in unstructured environments, it is not easy to enumerate all possible failures beforehand. In the context of safe skill manipulation, we reformulate skills as base and failure prevention skills where base skills aim at completing tasks and failure prevention skills focus on reducing the risk of failures to occur. Then, we propose a modular and hierarchical method for safe robot manipulation by augmenting base skills by learning failure prevention skills with reinforcement learning, forming a skill library to address different safety risks. Furthermore, a skill selection policy that considers estimated risks is used for the robot to select the best control policy for safe manipulation. Our experiments show that the proposed method achieves the given goal while ensuring safety by preventing failures. We also show that with the proposed method, skill learning is feasible, novel failures are easily adaptable, and our safe manipulation tools can be transferred to the real environment.

I. INTRODUCTION

Robots that are used in domestic environments require manipulation skills to perform various manipulation tasks [1]. Considering a kitchen environment, a robot can be assigned to cook various recipes such as soup or a cake. Primitive or compound motor skills such as stirring and pouring are needed to make these recipes. Existing learning methods enable learning such skills effectively [2], [3].It is also crucial to ensure that these skills are executed safely. However, even well-designed skills are prone to fail in the real world due to wrong assumptions, perceptual errors, or changing environmental conditions [4]. These may threaten the integrity of the workspace. For example, a robot stirring soup may spill the soup or slide the pan out of the stove which can be harmful to the people nearby, its workspace, and itself. Therefore, the capability of the robot should not be limited to performing manipulation skills only, they also need to monitor/detect potential failures and prevent them for safety. For that purpose, we propose learning failure prevention skills augmenting the safety of robotic manipulation.

Manipulation skills are often considered closed-loop sensorimotor control systems for robots to complete various tasks. With recent developments in reinforcement learning

No safety concerns

Skill Library

V.

Base Stir Skill

Fig. 1: A sample base skill, stir, may have been learned without taking into safety concerns. To make it safer, failure prevention skills are added in the library, and selected when needed.

(RL), even though robots can learn and use manipulation skills effectively for a limited number of objectives, they struggle when the number of objectives increases due to the curse of dimensionality. Therefore, RL-based skill learning rather focuses on achieving a given task without considering potential failures. Safe reinforcement learning approaches [5]–[8] apply learning to satisfy the goal while keeping the execution safe by estimating the risk. These approaches have the intention to prevent failures by learning to avoid unsafe states, however, they do not respond to scenarios with several failures as in unstructured environments. A previous work [9] addresses this issue but with a limited number of failures. In summary, even though robots have better accuracy and persistence in manipulation tasks, humans surpass robots in detecting and preventing failures. Arguably, an optimal skill policy can complete the task without failures, however, it will be limited to the known failures which are experienced during training. For a novel failure, the previous skill would become obsolete and should be learned again.

Representing a skill as smaller skills combined with a hierarchy [10]–[12] benefits from reduced complexity of the goal for optimal skill learning. While such skills can be predefined, it is also possible to learn smaller skills incrementally [13], [14]. Even though hierarchical approaches focus on learning several skills none of the learned skills does not have an explicit focus on ensuring the safety of the execution. For that purpose, the safety and the goal can be considered as two different objectives, and learning safe and robust skills can be expressed as a multi-objective reinforcement learning problem by decomposing the reward. Separating rewards and punishments can effectively perform both the task and failure prevention (contextual inhibition and spontaneous recovery) [15]. In the context of multi-objective reinforcement learning, positive rewards (reward) can be used to learn how to complete the task(main objective), whereas negative rewards (risk) can be introduced to avoid

This work is supported by a grant from the Scientific and Technological Research Council of Turkey (TUBITAK), Grant No. 119E-436.

¹Artificial Intelligence and Robotics Laboratory, Faculty of Computer and Informatics Engineering, Istanbul Technical University, Maslak, Turkey {akab, sariel}@itu.edu.tr

²School of Information Technology, Center for Applied Intelligent Systems Research, Halmstad University, Halmstad, Sweden

potential failures (safety objective). Many recent works [16]– [20] decompose the reward to learn to achieve the goal safely but they only respond to a single failure type. It can make a crucial difference to balance both rewards and risks whereas several objectives would result in suboptimal results for each objective. Especially, failures and their consequences are not limited and can vary in a dynamic scene, therefore, an increased number of objectives can make the multi-objective reinforcement learning problem unfeasible.

To address these problems, we propose a modular method where base skills are augmented by failure prevention skills, enhancing their safety. We group skills in the skill library into two categories according to their purposes: base skills and failure prevention skills. Skills with the purpose of reaching a goal to complete a task are defined as base skills (i.e., stirring or pouring). Skills with the purpose of preventing failures define failure prevention skills (such as prevention of sliding, overturning, or spilling). The latter are more reusable skills for augmenting different base skills. The proposed method has a hierarchy between skills and skill selection. The lower level of the hierarchy is a skill library which is composed of base skills and failure prevention skills. For each potential failure, a risk estimation model is defined to estimate the risk of the failure happen in-near future. Note that for the sake of simplicity, autonomous detection and identification of failures [4] is out of the scope of this work. Failure prevention skills are learned and added to the skill library for preventing potential failures. The higher level in the hierarchy rather involves a skill selection policy, triggering the optimal skill from the skill library to safely accomplish the task. Figure 1 illustrates this concept.

In the case of detecting a novel failure, the proposed method can easily be revised by only learning a novel failure prevention skill that mitigates this particular new failure type. Therefore, it becomes easier to adapt to new environmental conditions where novel failures could be encountered.

The proposed method is evaluated in a simulated environment and the skill library formed in the simulated environment is transferred to a real environment for real world evaluation. To the best of our knowledge, this is the first study that addresses learning reusable failure prevention skills to enable failure precautions into a skill library.

A. Contibutions

Our main contributions are as follows:

- the formulation and implementation of a modular and hierarchical method for safe robot manipulation;
- enabling learning reusable failure prevention skills as precautionary switching policies in a skill library for safe manipulation;
- adapting to novel failures and augmenting incrementally;
- real-world applicability by effective safety precautions in the physical world.

II. LEARNING FAILURE PREVENTION SKILLS FOR SAFE ROBOT MANIPULATION

Cognitive robots are equipped with either hand-coded or learned motor skills to achieve a given task. Even though these skills work well for controlled environments, in unstructured environments, their outcomes are not always as expected, and even worse, undesired/unsafe situations may occur due to failures in changing situations. To ensure safety, it is crucial for the robot to anticipate potential failures before they occur, and to prevent them if possible. In this work, we address this problem and formulate it as augmenting base robot skills with appropriate precautions to make them safer without changing them.



Fig. 2: Hierarchical Failure Prevention Model

We define a base skill, π_b , as a motor skill to achieve either a primitive or a compound action (e.g., pick and place objects, pour liquids, stir a bowl of ingredients, etc.). During performing this skill in a setting different than the trained one, failures are inevitable due to either wrong assumptions or unanticipated situations. The probability of a failure to occur is defined as a risk, (ρ^k , in the range from 0 and 1). The safety of the execution of π_b can be monitored by continually checking the occurrence of a finite set of risks: $\{\rho^0, \dots, \rho^n\}$ which can be designed previously or discovered online. The main problem that we address in this study asks; when the execution of a base skill π_b increases the risk of a failure (ρ^k) , to learn and activate necessary skills that enable transition to a safe state. We call this type of a complementary skill as failure prevention skill π_{p_k} that is responsible for reducing a corresponding risk. Note that, these failure prevention skills are generic skills that can be used to augment different base skills. Therefore, the problem asks for learning these failure prevention skills, and taking over control of these skills $(\pi_{p_0}, ..., \pi_{p_n})$ when necessary during the execution of the base skill such that the whole execution is safe. Thus, a dynamic chain of skills (i.e., a linear sequence of the base skill and the failure prevention skills in different orders or by different selections) are executed based on the circumstances of the world. This problem also asks for effective and efficient selection of the prevention skills during execution.

The skill library L of the robot includes both the base skills and failure prevention skills (Equation 1). L can be gradually built by adding novel skills online. Each learned skill extends the skill library to make it robust against each failure type that is observed.

$$L = \{\pi_{b_0}, ..., \pi_{b_m}\} \cup \{\pi_{p_0}, ..., \pi_{p_n}\}$$
(1)

In our solution to this problem, augmentation of a base skill π_b with safety precautions is done in three steps; observing failures and obtaining risk estimation models, learning failure prevention skills, and learning to select which skill to execute. In the first step, a failure is observed during the execution of the base skill π_b . In the second step, a corresponding novel failure prevention skill π_k is learned. In the third step, a skill selection policy π_{Ω} is used to select a skill from *L* using the estimated risk models. The resulting model is depicted in Figure 2. The following subsections describe this process in detail.

A. Learning a Base Skill

A base skill π_b can be hand-coded by an expert or learned by optimizing models such as Markov Decision Process (MDP) and Dynamic Movement Primitives (DMP) [21] using reinforcement learning (RL) or learning from demonstration (LfD). However, these models may not work as well as desired when the robot is exposed to reality as unexpected failures are likely in different settings other than the trained one. Therefore, a learned skill needs to be adapted to changes that occur in the environment. However, this adaptation may degrade the effectiveness of actual task performance (i.e., base skill) for the sake of adapting to changes. Therefore, we propose to augment the base skill to make it safer without changing it. This also ensures a more reusable solution to be used as a library of skills that can be used to augment other base skills as well.

B. Risk Estimation Models

A risk ρ^n is a binary safety estimate (*safe - risky*) against a failure, and presents the probability of a failure to occur in the near-future. In order to continue the execution safely, failure prevention decisions should be given based on risk estimations in real time.

Since the failure detection is not the main focus of this work, we use a rule-based risk estimation model which is expressed as a finite state machine $\rho^n = FSM(\chi^n, \kappa_a^n, \kappa_d^n)$ as illustrated in Figure 3. A risk ρ^n uses an observable parameter χ^n from the environment, and evaluates the risk using the activation threshold κ_a^n and the deactivation threshold κ_d^n where $\kappa_a^n \neq \kappa_d^n$. A safe state is transitioned to a risky state if χ^n is observed to be larger than κ_a^n . A risky state is transitioned to a safe state if χ^n is observed to be smaller



Fig. 3: Safe - Risky transition model

than κ_d^n . The interval between κ_a^n and κ_d^n prevents undesired fluctuations between states when the observed parameter is close to thresholds.

C. Learning Failure Prevention Skills

In our method, a failure prevention skill is modelled with MDP and learned to prevent a risky situation. The reward of a failure prevention skill is determined by the corresponding risk estimation as given in Equation 2. Since risk estimations are binary, the reward is sparse. During the training, the failure prevention skill policy is optimized to maximize the reward as it minimizes the risk.

$$R^{risk} = 1 - \rho^{risk} \tag{2}$$

Risky states are expected to be observed occasionally. To learn to prevent risky states, the robot should observe these cases quite often. However, this is not the case if it is not intended. In order to improve the sample efficiency of learning a failure prevention skill policy, the robot should be able to experience risky states more often. For that purpose, specific procedures are designed to create different failure situations. Therefore the robot uses these procedures to create a situation with a risk. Then the robot will be able to experience sequences of actions from risky situations to safety and learn an optimal policy that prevents failures.

D. Skill Selection

Once the skill library L is formed, the robot can execute the task safely with a skill selection policy π_{Ω} that arbitrates over all skills in L. π_{Ω} achieves this selection in a hierarchical fashion (See Figure 2). π_{Ω} selects a skill from L using risk estimations $\{\rho^0...\rho^n\}$, thus, it is expected to select an appropriate failure prevention skill to reduce a risk to prevent any catastrophic situations before they occur, and select the base skill when appropriate to complete the task.

We use a rule-based mapping from risk estimations to skills in L to form the skill selection policy π_{Ω} . With π_{Ω} , the robot executes the base skill π_b in a state without any risk. With the observation of a risk ρ^k , the corresponding failure prevention skill π_{p_k} is selected to reduce the risk ρ^k . Upon detection of multiple risks, predefined priorities between failure prevention skills determine which skill to be executed first. Priorities are determined by the impact of the failure on the manipulation's safety reliability. Note that in risks are sorted from most important to least important, therefore, the failure prevention skill corresponding to the risk with the greatest index is selected first.

III. Empirical Evaluation of Augmenting Stir Skill

In this section, we present our case study on a continuous *stirring* task. To accomplish this task, a Baxter humanoid robot uses a spoon to stir particles in a bowl. The goal is for the robot to move particles in the bowl for a given time. During the nominal execution of the *stirring* skill, one natural failure is observed: *spilling* the particles from the bowl. For different objects, environments, and/or control

conditions two additional failures might occur in our setup, *sliding* the bowl and *overturning* the bowl. Failure prevention skills are learned and used to augment the *stir* skill for the safety of the execution.



Fig. 4: Experimental setups. (a) Real (b) Simulated Environment.

In the real world environment, a Baxter humanoid robot with a spoon attached to its gripper is situated in front of a table. A bowl (r = 8cm, h = 8cm) is placed on the table, and several white-colored, sphere-shaped particles $(r = 1 \pm 0.5 \text{ cm})$ are placed in the bowl. In front of the table, a Kinect One RGBD sensor is placed to track the bowl and the particles. The real environment is shown in Figure 4a. The simulated environment is created using CoppeliaSim [22] and designed similar to the real environment. 40 sphereshaped particles are placed in the bowl. The number of the particles is experimentally determined, and the best number is selected according to the competence to observe the mentioned failures. During the simulated experiments, red and blue colors are used for the particles to observe the performance of the robot on the stirring task with the human eye. The simulated environment is shown in Figure 4b.

In the simulated environment, two setups are used to obtain optimal skill policies; the fixed bowl setup and the unrestricted setup. The fixed bowl setup restricts the effect of forces from the robot to apply on the bowl, resulting bowl to keep its position and orientation. Therefore, the robot focuses on the goal without considering failures related to the bowl pose. Then, the unrestricted setup is used for learning failure prevention skills for previously unconsidered failures. Forces acting on the bowl result in failures such as sliding and overturning.

First, we present our method in the simulated environment. Details of learning the stir base skill, forming risk estimation models, and learning failure prevention skills are given in Section III-A, Section III-B, and Section III-C respectively. The skill selection among these skills and the evaluation of the proposed method that uses these skills is given in Section III-D. Then, the learned skills are transferred into the real environment, and the proposed method is evaluated in the real world; whose results are presented in Section III-E. The additional materials and videos of both simulated and real robot are available¹.

A. Learning Stir Base Skill

We use the *stir* skill as the base skill in our study. With the *stir* base skill, the robot uses a spoon to continuously stir the particles inside a bowl without considering any potential failures. Therefore the fundamental goal of the stir base skill is to move particles as much as possible. By focusing on only one objective while omitting potential failures, we reduce the problem from multi-objective learning to single-objective learning, making the learning more straightforward.

In this work, we formulate the skill learning problem as an MDP with a tuple: $\langle S, A, T, R, \gamma \rangle$ where $s_t \in S$ is a continuous state, $a_t \in A$ is a continuous action, $T(s_{t+1}|s_t, a_t)$ is transition probability, $R(s_t, a_t, s_{t+1})$ is the reward and γ is the discount factor. In continuous state/action environments, Deep Deterministic Policy Gradients (DDPG) [23] is one of the most prominent approaches for skill learning. With DDPG, a skill is denoted by two policies; an actor policy π and a critic policy Q. The actor policy π maps states $s_t \in S$ to actions $a_t \in A$ in a continuous domain, and the critic policy Q estimates values of state-action $s_t - a_t$ pairs. For exploration, a noise ν from an Ornstein-Uhlenbeck [24] process is used. Acting according to a_t results in a transition to state s_{t+1} , thus, the robot obtains a reward $r_t \in R$ depending on the task to learn. Transitions (s_t, a_t, a_t) s_{t+1}, r_t) are collected into an experience replay memory, and they are used for the optimization of the target actor and the target critic policies. Actor and critic policies are updated with target policies periodically.

The *stir* skill policy π_b is learned as the base policy in the simulated environment. The policy has been optimized by formulating the execution as an MDP with the following state and action spaces:

$$S_{stir} = [x, \phi]$$

$$A = [\Delta x]$$
(3)

where S_{stir} is composed of the position of the spoon relative to the bowl (x) and the phase (ϕ) of the execution, and A is composed of the displacement of the position of the spoon (Δx) . Parameters in the state and the action spaces are selected as their relevance to the problem and for simplicity, positions are represented with 2D coordinates of the plane parallel to the table. x is the Cartesian positions between $[-\eta, \eta]$ where η is the safety perimeter for the robot to operate within the allowed range. ϕ is the phase value representing the relative time of the execution, making the system time-variant. Using ϕ adds a temporal aspect to the skill, and we experimentally found it useful for periodic movements such as *stir* skill. It is a value between $[0, \phi_{max}]$, and updated after each movement decision with ϕ_{step} using:

$$\phi = (\phi + \phi_{step}) \mod \phi_{max} \tag{4}$$

Stir base skill is expected to reflect the essence of stirring without safety concerns, reducing the complexity of the

¹https://air.cs.itu.edu.tr/projects/tubitak-119e436.html

problem. Therefore, an optimal *stir* base skill policy that focuses specifically on stirring is obtained with fixed bowl setup. Stirring can be considered as the continuous movement of the particles in the bowl hence the reward is formulated as the sum of the displacement of the particles in 50ms. The reward function is given as follows:

$$R^{b}(s_{t}, a_{t}) = \sum_{k=0}^{n} r^{b}(x_{t}^{k}, x_{t+1}^{k}) \quad ,$$

$$r^{b}(x_{t}^{k}, x_{t+1}^{k}) = \begin{cases} ||x_{t}^{k} - x_{t+1}^{k}||_{2}, & \text{if } in(x_{t+1}^{k}, bowl) \\ 0, & \text{else} \end{cases}$$
(5)

where r^b is the displacement of a particle between two consecutive states 50ms apart, if the particle is in the bowl and 0 otherwise. n is the number of particles in the bowl which is 40 in our setup. R^b is the reward of the base skill which is the sum of individual rewards for each particle.

Both actor and critic neural networks are designed with two linear feed-forward layers with 400 and 300 neurons respectively, and with ReLU activation layer in between. Networks are trained with Deep Deterministic Policy Gradients (DDPG) [23] for 1500 episodes with 500 steps where the batch size is 128, learning rates(α_a, α_c) are 0.0001 and discount factor (γ) is 0.99. For exploration, linearly decaying epsilon is used and the noise is modelled with Ornstein-Uhlenbeck [24] process with parameters $\mu_{\nu} = 0$, $\sigma_{\nu} =$ 1, $\theta_{\nu} = 0.15$.

As the result of the training, the best policy π_b is selected as the optimal policy for the *stir* base skill and added to L.



Fig. 5: The observed failures during testing the learned *stir* base skill. (a) Sliding the bowl (b) Overturning the bowl (c) Spilling contents from the bowl

B. Risk Estimation Models

Potential failures that may occur are observed during test executions of π_b in the simulated environment. When the *stir* base skill (π_b) is tested in the fixed bowl setup, *spill* failure is observed. *spill* failure happens when the particles in the bowl get out of the bowl by the forces applied by the spoon. However, when the *stir* base skill (π_b) is tested in the unrestricted setup, two additional failures are observed: *slide* and *overturn* failures. During the execution, the bowl is expected to stay close to the starting position. But forces applied on the spoon may slide the bowl away from the starting point resulting in *slide* failure. Especially, the increased amount of particles results in the bowl sliding

TABLE I: Risk Estimation Models

| | χ | κ_a | κ_d |
|----------|--------|------------|------------|
| slide | d | 0.05m | 0.02m |
| overturn | θ | 0.3rad | 0.1rad |
| spill | V | 0.66 | 0.33 |

away frequently. While the bowl slides away, the friction between the bowl and the table can act as a hinge, rotating the bowl and spilling most of the particles, resulting *overturn* failure. Depictions of these failures are shown in Figure 5. These failures are used as testbeds to learn failure prevention skills to augment the stir skill for safety.

For that purpose, relevant risk estimation models are designed as finite state machines(FSM) with two states, *safe* and *risky*. The distance between the current and the initial position of the bowl (*d*), the angle between the *z*-axis of the bowl and the normal vector of the table (θ), and the maximum excluded volume ratio of particles (*V*) are the parameters in the design of risk estimation models for *slide*, *overturn*, and *spill* failures, respectively. An observed parameter greater than risk activation κ_a triggers the risk. Risk disappears when the observed parameter is reduced to risk deactivation κ_d . Designed risk estimation models for *stir* base skill (π_b) and their empirically selected parameters are given in Table I.

The risk estimation model of *slide* failure uses the distance (d) between the initial location of the bowl (x_0) , and the current location (x_t) of the bowl, calculated as $d = \Delta(x_0, x_t)$. The risk estimation model of *overturn* failure uses the rotation angle (θ) between the initial pose of the bowl (θ^0) and the current pose (θ^t) of the bowl, calculated as $\theta = \theta^t - \theta^0$. The risk estimation model of *spill* failure uses the overflown volume of particles (Ve_n) from the volume of the bowl (Vb). The ratio of the overflow of each particle (V_n) is calculated as the overflown volume of the particle $(Ve_n - Vb)$ over the volume of the particle (Vc_n) as $Vn = (Ve_n - Vb)/Vc_n$. Then the overflown volume(V) is calculated as the maximum ratio of overflow $(max(V_n))$ among the particles. The pose of the bowl and particles are directly acquired from the simulation as its ground truth value, and particle volumes are predefined.

C. Learning Failure Prevention Skills

Failure prevention skills are acquired by learning failure prevention policies for corresponding risk estimation models. They are learned in the simulated environment with the same network design and optimization parameters as the learning of the base skill (Section III-A). Different from base skill learning, these skills have different state representations and reward functions. Additionally, they use initial procedures to increase the risk at the start of the episode letting the robot encounter the corresponding risk easily.

Failure prevention skills require additional parameters related to the failure they respond to on top of the base skill state representation. They use d, θ , and V parameters from the corresponding risk estimation model to learn the optimal robot movement that avoids the risk. The state of a failure prevention skill is obtained by concatenating S_{stir} with χ

TABLE II: Evaluation results of augmentation of the stir skill with failure prevention skills in the simulated environment

| Model | Stir Reward | | Spill | | Slide | | Overturn | |
|------------|-------------|-------|-------|------|-------|-------|----------|-----|
| | mean | std | mean | std | mean | std | mean | std |
| π_b -F | 329.00 | 17.57 | 4.55 | 1.50 | N/A | N/A | N/A | N/A |
| π_b -U | 251.29 | 12.90 | 2.2 | 1.32 | 0.11 | 0.01 | 0 | 0 |
| L_4 -U | 159.64 | 34.67 | 0.20 | 0.52 | 0.05 | 0.03 | 0 | 0 |
| L_2 -F | 87.20 | 53.98 | 0.10 | 0.30 | N/A | N/A | N/A | N/A |
| π_c -U | 126.76 | 10.70 | 0.15 | 0.36 | 0.022 | 0.007 | 0 | 0 |

from the corresponding risk estimation model; d for *slide* failure, θ for *overturn* failure and V for *spill* failure as given in Equation 6. The reward for each failure prevention skill is sparse and acquired with the corresponding risk estimation model as explained in Equation 2.

$$S_{slide} = [x_{spoon}, \phi, d]$$

$$S_{overturn} = [x_{spoon}, \phi, \theta]$$

$$S_{spill} = [x_{spoon}, \phi, V]$$
(6)

To learn a failure prevention skill effectively, risky states should be experienced during training which may not occur often enough. Initial procedures are designed to move the robot to a risky state, thus, letting the robot experience risky states. Each failure has different initial procedures. For prevention of *slide*, the initial position of the bowl is sampled randomly triggering the risk ρ^{slide} . For prevention of *overturn*, the robot moves the spoon toward a random direction until the risk $\rho^{overturn}$ is triggered. For prevention of *spill*, the robot moves the spoon toward a random location in the bowl until the risk ρ^{spill} is triggered. The episode starts once the corresponding risk is triggered using initial procedures, and the robot learns how to reduce the risk.

As the results of trainings, best policies of $\pi_{p_{slide}}$, $\pi_{p_{overturn}}$, $\pi_{p_{spill}}$ are selected as optimal failure prevention policies for sliding, overturning, spilling failures respectfully. Selected policies are added to the skill library forming L_4 .

D. Overall Results

A skill library is initialized with the learned base skill π_{stir} (Section III-A). Then, learned failure prevention skills (Section III-C) are added to the skill library (L) augmenting the base skill for safe robot manipulation. We define safe robot manipulation as skill selection in real time from the skill library consisting of a base skill to complete the task and failure prevention skills to reduce failure risks. We use a rule-based skill selection policy (Section II-D). The priorities are given in Equation 7 where $I(\pi_{p_k})$ is the importance of the failure k which is determined by its effect on the task. Overturning restrains completion of the task, therefore, it is the most prior failure. Spilling restrains obtaining the best outcome from a completed task, and it is set as the second most prior failure. Sliding results in soft failures such as reducing observability, and extending workspace which may lead to additional failures. In our setup, since it does not prevent the completion and the success of the task directly, it is the least prior failure.

$$I(\pi_{p_{overturn}}) > I(\pi_{p_{spill}}) > I(\pi_{p_{slide}})$$
(7)

Note that, for a different setup, the importance of failures can be different from what we present. For example, if the robot stirs a pan on a stove, keeping the pan on the stove would be more important than spilling the content.

1) Evaluation of the Augmented Stir Skill: The comparative results are presented in Table II where all methods are tested for 20 episodes with 1000 steps. The table reports mean and standard variation of performance measures (R, d, θ, V) . We first evaluate π_b in the fixed bowl setup for benchmarking $(\pi_b$ -F). Evaluation results indicate that π_b stirs effectively while spilling occasionally. Then, our method is evaluated in the unrestricted setup $(L_4$ -U). Comparing π_b -F and L_4 -U, we can claim that the proposed method is significant for failure prevention with a tradeoff of stir efficiency. The loss of stir efficiency is tolarable as the environment of our method is more challenging and vulnerable to failures than the former. Therefore, it uses its time effectively to prevent any of the failures and stir whenever it is safe.

For a fair comparison between our method and π_b , the latter is also tested in the unrestricted setup (π_b -U). Comparing π_b -U and L_4 -U, we see that our method is safer with a tradeoff of stir efficiency. Note that π_b -F does not perform better than π_b -U for safety even though the number of spill events decreased. In the unrestricted setup, forces affecting particles get diminished since a part of the force is transferred to the bowl, causing the number of spill events to decrease. Note that no overturn event is detected in the results because overturn failure occurs when the robot interacts with the bowl which only happens with $\pi_{p_{slide}}$. While this never happens for π_b -F and π_b -U, L_4 -U successfully prevent overturn failure with $\pi_{p_{overturn}}$.

2) Adaptability to Novel Failures: To show the adaptability of our method, we show how a robot working in the fixed bowl setup adapts its library to the unrestricted setup. In the fixed bowl setup, only spill failure can be observed since the bowl is fixed, and the skill library L_2 is formed with π_b and $\pi_{p_{spill}}$. When the bowl orientation restrictions are removed from the environment, novel failures; *sliding* and *overturning* are observed, and $\pi_{p_{slide}}$ and $\pi_{poverturning}$ skills are learned to prevent them, respectively. Now the skill library is extended (L_4) with these skills. When we compare L_2 -F and L_4 -U, it can be seen that with our method, it is easy to adapt to new conditions by discovering novel failures, and learning corresponding failure prevention skills.

3) Modularity vs Compound Skill: One of the main question that should be discussed is whether the modularity helps with the failure prevention problem or not. For this investigation, a compound base-failure prevention skill(π_c)

is learned that takes into account all three failures during learning to stir, and penalizes accordingly. π_c is trained in the unrestricted setup with a the same training setup(III-A), except the reward function. As the reward, the sum of all rewards for each objective is used as given in Equation 8.

$$R^{bp} = R^b + R^{slide} + R^{overturn} + R^{spill}$$

$$\tag{8}$$

Comparing L_4 -U and π_c -U from Table II, we can deduce that our method performs slightly better for the stir efficiency and the compound skill performed slightly better for failure prevention. However, when we compare the learned stir patterns of both methods (see a randomly selected particle's trajectories in Figure 6), we see that π_c -U does not perform a circular movement, it rather moves the spoon linearly in a narrow area resulting in only slight changes in particle locations and not an effective stir. This performance degradation also supports the decrease in the observed in the average stir reward. Due to this slow pattern of movement, the probability of failures are observed are small compared to that of our method. The differences in percentages are not significant. Our further analysis shows that modular methods reward is highly dependent on how fast the prevention policy reduces the risk.



Fig. 6: Trajectory that a particle travels using compound skill π_c -U(orange) and modular method L_4 -U(blue)

E. Transfer to the Real World

In this work, we directly transfer d and θ parameters from simulation to the real world. However, we use domain adaptation for the rest of the parameters(V) that can not be represented directly.

For the *stir* base skill, the position of the spoon x_spoon which is attached to the gripper is obtained by the kinematic chain of the robot. ϕ is initialized as 0 and ϕ_{max} is set to 50. For $\pi_{p_{slide}}$, $\pi_{p_{overturn}}$, $\pi_{p_{spill}}$, the pose of the bowl is observed using a particle filter-based tracking algorithm [25]. Then d and θ are estimated from the pose of the bowl. Detecting and tracking particles in the bowl is impractical in the real world. In order to obtain V, we used a point cloud filtering approach using Point Cloud Library (PCL)², sampling points from particles in the bowl. We estimated

max(z), where z is the position of a sample point on the z-axis, and max(z) is transferred to V using Equation 9 as the mapping function.

$$V \approx (max(z) - z_{bowl})/2r \tag{9}$$

 z_{bowl} is the tip position of the bowl on the z-axis. r is assumed to be a predefined fixed radius of a particle. The visualization of the observations from the real world is given in Figure 7. Action from a skill represents a desired movement vector for the gripper in the Cartesian domain. The robot is controlled using a set of position controllers in the joint state domain. We implemented a high-level controller that gets action from L_4 and calculates joint state goals with 100ms frequency. First, the Cartesian goal is calculated by shifting the position of the gripper with the desired movement vector. Then, the joint state goal is calculated from the Cartesian goal using MoveIt³. Then, the joint state goal is used to set desired positions of position controllers. Additionally, with 100ms frequency, ϕ is increased by one (Equation 4).



Fig. 7: Visualization on the point cloud: Green is for tracking the bowl, red is for the sampled points from the content of the bowl

By transferring L_4 model, we have shown our method's capability to prevent failures in the physical world. Since the exact positions of the particles in the bowl cannot be observed directly as in the simulation, the cumulative reward of the execution can not be determined in the real world. Therefore, the model's success can be stated qualitatively. Based on our observations, we can conclude that continuous *stirring* with preventative skills can be successfully applied. An example execution trace is given in Figure 8.

We conducted additional tests to analyze the performance of L_4 model's preventative ability. At first, L_4 model is tested for each failure individually by starting the scenario in a risky condition. In these tests, the robot easily avoided failure and continued with the base skill after the risk is reduced. Then L_4 model is tested for 2-minute long executions. In these tests, different amounts of particles are used. When the bowl is about 70% or less full, the robot runs as expected, prevents failures in risky states, and stirs continuously in safe states. When the bowl is more than 70% full, the robot can not reduce the max(z) easily and gets stuck in failure prevention. In some cases with high amount of particles, the actions of the robot can result in



Fig. 8: An example execution trace of L_4 model in the real world: (a) Robot uses $\pi_{P_{slide}}$ to correct the position of the bowl (b) While executing $\pi_{P_{slide}}$, bowl starts to overturn and the robot decides executing $\pi_{Poverturn}$ (c) After executing failure prevention skills, the robot continues to *stir* with π_b .

some particles popping out of the bowl. This failure is not observable by the available risk estimation models, yet may be estimated by using additional sensors such as force sensors which are not available in this work. Additionally, L_4 model is tested with interference by the operator. The operator interferes the execution by changing the location of the bowl, adding additional particles, and removing particles from the bowl. The model adapts robustly to new conditions without stopping and preventing possible failures.

IV. CONCLUSION AND FUTURE WORK

In this work, we propose a modular method where base skills and failure prevention skills are combined. Failure prevention skills are learned for preventing potential failures and used for augmentation of base skills to make them safer by using rule-based skill selection. Evaluation results indicate that learned failure prevention skills help base skills to complete their tasks safely. The method is also extendable upon novel failures. We also transferred learned skills to be used in the real world successfully. Additional materials, and videos of simulated and real results are available online. Even though potential failures are determined using base skills, failure prevention skills are learned independently and whether they can be used for augmenting different base skills is a question that we want to address in the near future.

REFERENCES

- M. Ersen, E. Oztop, and S. Sariel, "Cognition-enabled robot manipulation in human environments: Requirements, recent work, and open problems," *IEEE Robotics & Automation Magazine*, vol. 24, no. 3, pp. 108–122, 2017.
- [2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [3] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv*, no. CoRL, pp. 1–23, 2018.
- [4] A. Inceoglu, E. E. Aksoy, A. C. Ak, and S. Sariel, "Fino-net: A deep multimodal sensor fusion framework for manipulation failure detection," in *IEEE/RSJ International Conference on Intelligent Robots* and Systems (IROS), 2021.
- [5] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg, "Conservative Safety Critics for Exploration," *Arxiv*, pp. 1–25, 2020. [Online]. Available: http://arxiv.org/abs/2010.14497

- [6] S. H. Cheong, J. H. Lee, and C. H. Kim, "A New Concept of Safety Affordance Map for Robots Object Manipulation," *RO-MAN 2018* -27th IEEE International Symposium on Robot and Human Interactive Communication, pp. 565–570, 2018.
- [7] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," *34th International Conference on Machine Learning, ICML 2017*, vol. 6, pp. 4310–4319, 2017.
- [8] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine, "Leave no trace: Learning to reset for safe and autonomous reinforcement learning," *6th International Conference on Learning Representations, ICLR 2018* - Conference Track Proceedings, 2018.
- [9] A. C. Ak, A. Inceoglu, and S. Sariel, "When to stop for safe manipulation in unstructured environments?" in *Proceedings of the* 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 1767–1769.
- [10] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *Journal of Machine Learning Research*, vol. 22, 2021.
- [11] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 1503–1510, 2015.
- [12] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017. [Online]. Available: https://ojs.aaai.org/index.php/ AAAI/article/view/10916
- [13] A. Bagaria and G. Konidaris, "Option Discovery using Deep Skill Chaining," *International Conference on Learning Representations*, pp. 1–21, 2020.
- [14] A. Bagaria, J. Senthil, M. Slivinski, and G. Konidaris, "Robustly Learning Composable Options in Deep Reinforcement Learning," *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 2021.
- [15] R. Lowe and T. Ziemke, "Exploring the relationship of reward and punishment in reinforcement learning," *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL*, pp. 140–147, 2013.
- [16] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips 2017, pp. 5393–5403, 2017.
- [17] Z. Lin, L. Zhao, D. Yang, T. Qin, G. Yang, and T. Y. Liu, "Distributional reward decomposition for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019.
- [18] Z. Lin, D. Yang, L. Zhao, T. Qin, G. Yang, and T. Y. Liu, "RD2: Reward decomposition with representation disentanglement," in *Advances in Neural Information Processing Systems*, vol. 2020-Decem, no. NeurIPS, 2020.
- [19] S. Elfwing and B. Seymour, "Parallel reward and punishment control in humans and robots: Safe reinforcement learning using the MaxPain algorithm," *7th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics, ICDL-EpiRob 2017*, vol. 2018-Janua, pp. 140–147, 2018.
- [20] J. Wang, S. Elfwing, and E. Uchibe, "Deep reinforcement learning by parallelizing reward and punishment using the maxpain architecture," 2018 Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics, ICDL-EpiRob 2018, pp. 175–180, 2018.
- [21] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models formotor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.
- [22] "Robot simulator coppeliasim: Create, compose, simulate, any robot coppelia robotics." [Online]. Available: https://www.coppeliarobotics. com/
- [23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings, 2016.
- [24] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical Review*, vol. 36, no. 5, p. 823, 1930.
- [25] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal, "Probabilistic object tracking using a range camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Nov. 2013, pp. 3195–3202.