IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Reliable Robotic Task Execution in the Face of Anomalies

Bharath Santhanam[†], Alex Mitrevski[‡], Santosh Thoduka[*], Sebastian Houben[§,*], and Teena Hassan[§]

***Abstract*—Learned robot policies have consistently been shown to be versatile, but they typically have no built-in mechanism for handling the complexity of open environments, making them prone to execution failures; this implies that deploying policies without the ability to recognise and react to failures may lead to unreliable and unsafe robot behaviour. In this letter, we present a framework that couples a learned policy with a method to *detect visual anomalies* during policy deployment and to *perform recovery behaviours* when necessary, thereby aiming to prevent failures. Specifically, we train an anomaly detection model using data collected during nominal executions of a trained policy. This model is then integrated into the online policy execution process, so that deviations from the nominal execution can trigger a three-level sequential recovery process that consists of (i) pausing the execution temporarily, (ii) performing a local perturbation of the robot's state, and (iii) resetting the robot to a safe state by sampling from a learned execution success model. We verify our proposed method in two different scenarios: (i) a door handle reaching task with a Kinova Gen3 arm using a policy trained in simulation and transferred to the real robot, and (ii) an object placing task with a UFactory xArm 6 using a general-purpose policy model. Our results show that integrating policy execution with anomaly detection and recovery increases the execution success rate in environments with various anomalies, such as trajectory deviations and adversarial human interventions.**

***Index Terms*—Cognitive control architectures, failure detection and recovery, learning from experience, visual learning**

I. INTRODUCTION

TO increase the flexibility of a robot's execution and reduce the requirements on explicitly modelling the execution process, robot execution policies are often acquired using learning methods, such as imitation learning or reinforcement learning (RL). The usefulness and versatility of learning-based robot skills, particularly for robot manipulation, has been demonstrated in many contexts [1], [2], [3]; this includes recent large robot foundation models [4], [5], [6], [7], which are potentially able to generalise experiences between skills and even between robot modalities.

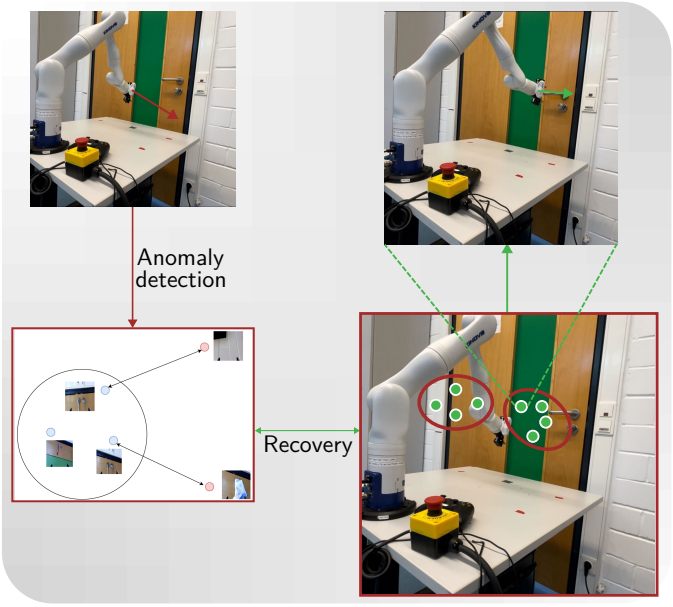


Fig. 1: An overview of our proposed framework for failure-aware policy execution. During execution, visual anomalies are detected using a self-supervised feature extraction method and different recovery actions are attempted (pausing, perturbation, and finally a reset using information from a learned success model).

The motivation for our work is that learning-based policies are usually defined without considering explicit strategies for handling anomalous cases during execution. This can reduce their applicability in practical applications, as a robot is then unable to recognise and recover from situations that may lead to execution failures. One strategy that can be used to ensure that a policy is likely able to avoid execution failures in various scenarios (though without explicit guarantees) is domain randomisation, namely training the policy on highly diverse data. This, however, poses significant requirements on the learning data quantity and variability, which is typically challenging to satisfy in practical robot applications. Another strategy to improve task robustness is to use model-based execution, as this would enable a robot to make predictions about the expected observations and react to discrepancies. Versatile models of execution are, however, generally challenging to learn [2], particularly over a long horizon and beyond specific modes of execution. This suggests that an alternative approach may be more flexible, where a robot is equipped with an ability to identify situations that may lead to execution failures and recover from them.

In this letter, we describe such a learning-based robot system that combines learned policies with anomaly detection

Manuscript received: July 01, 2025; Revised September 24, 2025; Accepted October 21, 2025.

This article was recommended for publication by Editor T. Ogata upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by the b-it foundation and in part by a starting research grant for Alex Mitrevski provided by Bonn-Rhein-Sieg University of Applied Sciences (project KEROL). (*Corresponding authors: Bharath Santhanam, Alex Mitrevski, and Santosh Thoduka*)

[†]Bharath Santhanam is with NEURA Robotics, 72555 Metzingen, Germany bharath.santhanam@neura-robotics.com

[‡]Alex Mitrevski is with the Division for Systems and Control, Chalmers University of Technology, 41258 Gothenburg, Sweden alemitr@chalmers.se

[*]Santosh Thoduka and Sebastian Houben are with the Fraunhofer Institute for Intelligent Analysis and Information Systems, 53757 Sankt Augustin, Germany <santosh.thoduka, sebastian.houben>@iais.fraunhofer.de

[§]Sebastian Houben and Teena Hassan are with the Institute for Artificial Intelligence and Autonomous Systems (A^{2}S), Bonn-Rhein-Sieg University of Applied Sciences, 53757 Sankt Augustin, Germany <sebastian.houben, teena.hassan>@h-brs.de

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

and execution recovery in a holistic framework. To detect anomalies during execution, we use self-supervised feature extraction from nominal visual data and nearest neighbour classification for identifying anomalous frames. In the case of anomalies, multiple recovery behaviours are attempted in succession (waiting, retrying, and then restarting the execution from a new initial state), where restarting is performed by sampling from an execution success model that is learned from the policy itself. The proposed framework, illustrated in Fig. 1, is agnostic of how the policy was trained.

We experimentally verify the individual elements of the framework and the system as a whole on two tasks with two robots and policy types: (i) a door handle reaching task, where the robot follows a policy trained in simulation that is transferred to the real world, and (ii) a task of placing an object inside a bowl, for which we use a pretrained Octo model [7] for execution. The results demonstrate that, without anomaly detection and recovery, the learned policies tend to experience failures due to data distribution shifts; the additional anomaly detection and recovery increase the robustness of the policies, with the main downside that the robot may sometimes act too conservatively and initiate recovery, although it could continue with its original execution.[1]

The letter's contributions include: (i) a policy-agnostic framework for monitoring execution policies and performing basic recovery, (ii) an evaluation of the framework's generalisation through different robots and policy types, and (iii) an open-source implementation of our method.[2]

II. RELATED WORK

Robot learning is commonly used for acquiring flexible execution policies, with work often focusing on training simulated policies and transferring them to the real-world [2], [3], [8]. The sim-to-real transfer can introduce domain shift and lead to suboptimal policy performance, but strategies such as domain randomisation [9] and domain adaptation [10] address this challenge. An alternative training strategy is imitation learning on real-world data; this is followed by general-purpose robot policies [5], [6], [7]. In this work, we use both simulation-based training, combining RL with domain randomisation for sim-to-real transfer, as well as general-purpose policies trained on real-world data.

While these methods have shown promise, the complexity of real-world environments often results in domain shifts that necessitate additional robustness measures. To enhance robustness, we implement visual anomaly detection, as visual data is a rich source of information about the environment and the execution itself. Learning approaches for anomaly detection primarily fall into two categories: reconstruction-based [11], [12] and memory bank-based methods [13], [14]. Reconstruction-based methods train models to reproduce nominal images, assuming anomalous images will be poorly reconstructed. In contrast, memory bank methods store features extracted from the nominal images, and classify test images as anomalous if their features are significantly different from the stored features. Our approach adopts the latter strategy, where we compare features of incoming images to stored image features from nominal robot task executions, due to its easier extensibility between tasks.

In addition to pure sensor-based anomaly detection, model-based trajectory evaluation methods, such as Lagrassa et al. [15], can be found in the literature as well. We do not learn a state transition model in order to simplify the model learning requirements, particularly as we aim for a policy-agnostic model that can also be easily integrated with large-scale, model-free policies.

Detecting anomalies during execution facilitates the implementation of recovery steps for safe execution. Hung et al. [16] integrate task execution with recovery using an imitation learning-based policy with concrete dropouts. The approach outputs a distribution of predicted actions, where the mean represents the chosen action, while high variance indicates uncertainty and triggers recovery actions. One limitation of [16] is that it tightly couples the task execution, monitoring, and recovery, requiring changes across all modules when modifications are made to one. Similarly, Lee et al. [17] learn a policy with uncertainty awareness so that anomalies can be detected. Concretely, imitation learning is used for acquiring a policy, and both epistemic and aleatoric uncertainty are estimated during execution; if the uncertainty reaches a high level, the execution is deferred to an expert policy. As [16], [17] tightly couples the task execution and monitoring, and also uses an expert policy for recovery, but this may not be available in general.

In the recovery context, Thananjeyan et al. [18] propose a framework that decouples the task execution and recovery policies. Here, the agent first executes a primary task policy and passes its output to a safety critic, which returns the probability of the action being unsafe; if this probability is high, a recovery policy is triggered to guide the agent to a safe state. Similar to [18], our approach maintains modularity by decoupling the task execution, anomaly detection, and recovery components. This property makes it applicable to various policy types (including general-purpose policies, which would otherwise have to be architecturally modified and retrained). Unlike [18], we do not use a learned safety critic, as training this requires data with various failure modes, which can be challenging to obtain in general. Reichlin et al. [19] propose a recovery method that learns a nominal state distribution density, and uses the gradient of this density to find recovery actions in out-of-distribution states. Similarly, Mitrano et al. [20] learn a model that evaluates whether action candidates will lead to successful recovery in a given state. Similar to [19], we learn using data of successful executions, but we learn an anomaly detector rather than a recovery model. The recovery model in [19], [20] could be used as a substitute of the first two levels of our recovery pipeline, though at a higher learning and overall computational cost.

Large language models (LLMs) and vision-language models (VLMs) have also been used for anomaly detection and recovery. Agia et al. [21] propose a method to monitor failures due to erratic behaviour or in task progress. For the former, the execution horizon by a diffusion policy is used for deriving an

[1] Accompanying video: https://youtu.be/oQT-xrRRvXk

[2] https://github.com/bharathsanthanam94/robust-robot-policy-execution

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

action consistency measure; for the latter, a VLM is queried with historical execution data and a description of the task objective. Liu et al. [22] propose a framework based on which multimodal robot data is encoded into scene graphs and used to extract relevant execution events, which are summarised in natural language and used as input to an LLM that identifies anomalies and proposes recovery plans. Conceptually, our anomaly detection method has similarities with the consistency estimation in [21], but we estimate the consistency implicitly by comparing with successful execution examples, and we do not assume a diffusion policy, as we aim for a policy-agnostic method. Our detection and recovery strategy could be integrated with a method such as [22], which would be used for recovering from plan failures, while our method would be used for short-term recovery of the ongoing skill.

III. FAILURE-AWARE ROBOT POLICY EXECUTION

In this letter, we present a methodology that aims to improve the robustness of a learning-based policy through anomaly detection and subsequent recovery. To achieve this, we roll out an execution policy in the target environment to collect nominal execution data, which we use to train a vision-based anomaly detection model. During deployment, we use this model to detect anomalies at each execution step, and trigger recovery actions when necessary. In this section, we describe the elements of our framework in more detail.

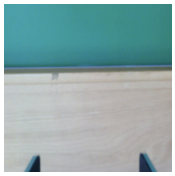
A. Execution Policy

As is common in the literature, we model the execution by a policy $\pi : (S, C) \rightarrow A$, which maps the robot's state $\mathbf{s} \in S$, and optional context $\mathbf{c} \in C$, to appropriate actions $\mathbf{a} \in A$. We assume a parameterised policy π_ϕ, modelled by a neural network with parameters ϕ, though our monitoring and recovery strategy is independent of the nature of the policy. For concreteness, we consider two different policy types in this letter: (i) a dedicated policy learned for a concrete task, and (ii) a general-purpose, language-conditioned policy.

a) Learning dedicated policies: To learn a dedicated execution policy, we use RL, so we model the execution by a Markov Decision Process (MDP); the robot thus interacts with its environment E and learns to take optimal actions by maximising a cumulative reward over time. Concretely, at each time step t, the robot selects an action $\mathbf{a}_t = \pi_\phi(\mathbf{s}_t, \mathbf{c}_t)$, such that the environment then provides a reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c}_t)$, which quantifies the immediate benefit of taking action $\mathbf{a}_t$ in state $\mathbf{s}_t$ under context $\mathbf{c}_t$. To ensure the applicability of the method to various robot skills and embodiments, we use a state space $S = (\mathcal{I}, Q, EE)$ that combines camera images $I \in \mathcal{I}$ as external observations, as well as proprioceptive measurements in the form of the robot's joint positions $\mathbf{q} \in Q$ and end effector positions $\mathbf{e} \in EE$.[3] While the images I can be from any type of camera, we use images from a wrist camera in this work. We use an action space A that consists of relative end effector motions $\mathbf{a} = (\Delta x, \Delta y, \Delta z, \Delta\alpha, \Delta\beta, \Delta\gamma)$, as task space control has been shown to speed up the learning process [23] and is also used in contemporary robot foundation models [5], [6], [7]. Since training an RL agent typically requires a large number of interactions with E, we train our policy in a simulated environment $E_{sim} \approx E$ and then transfer it to the real environment E. To ensure that the simulation-based policy is as invariant as possible to the visual domain shift that occurs due to the transfer, we use domain randomisation as a data augmentation technique.[4]

(a) Handle covered by a hand

(b) Handle covered by an object



(c) Deviated path

Fig. 2: Examples of anomalies in the case of a robot attempting to reach a door handle (wrist camera view)

b) General-purpose policies: Our anomaly detection and recovery method is agnostic to the nature of the policy training process, and is thus also applicable to general-purpose robot foundation models that have been trained on large imitation learning datasets, such as [6]. To demonstrate this aspect, we also consider a language-conditioned visuomotor policy $\pi_\phi^G(\mathbf{s}_t, \mathbf{c}_t)$, such that $S = \mathcal{I}$, and C are language commands. In this case, the action space A comprises relative end effector motions and a gripper command, namely $\mathbf{a} = (\Delta x, \Delta y, \Delta z, \Delta\alpha, \Delta\beta, \Delta\gamma, g)$, where $g \in [0, 1]$ indicates whether the gripper should be closed ($g \approx 0$) or opened ($g \approx 1$). Just as above, we use a wrist camera as input to π_ϕ^G. In this work, we use a pretrained Octo [7] model for execution. As Octo uses an underlying diffusion policy [24], it outputs a sequence of actions at each time step t. For simplicity and consistency with our dedicated policies, we only execute the first action in our evaluation.

B. Anomaly Detection

The performance of a learned policy depends on the quantity and quality of the training data: a larger dataset is likely to have more diverse state and action space coverage, thereby increasing the likelihood of successful execution. This, however, still does not guarantee that the execution will be unaffected by anomalies, which we define as deviations from the nominal policy execution. This is because the real world is complex and can introduce many novel scenarios that the trained policy may not be equipped to handle. For instance, as illustrated in Fig. 2, another agent may cover the robot's view, thereby preventing it from reaching a desired target.

In general, it is challenging to explicitly foresee all possible anomalous scenarios during training, so a more pragmatic strategy to handle anomalies is to monitor the execution and detect *any* deviations from the nominal execution as anomalies. In this letter, we propose a visual anomaly detection method

[3] In principle, $\mathbf{e}$ is not required in the input because it can be inferred from $\mathbf{q}$, but we use it explicitly in the state to provide richer context to the robot, thereby reducing the complexity of the learning process.

[4] In this work, we only use colour randomisation, as this was empirically sufficient for the task of interest.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

that can be integrated with the trained policy during online execution. This is based on the idea that execution anomalies can be detected using visual observations [25], which provide rich context that can be used to detect anomalies before failures occur. For instance, in Fig. 2, if a robot detects an obstructed target, it can adjust its execution to avoid collisions with the external agent. The general idea of our method is to compare online observations with observations collected during nominal policy execution, such that significant deviations are considered anomalous. We perform this comparison of observations at the level of image embeddings $\mathbf{z}$, which are extracted from a network trained in a self-supervised manner.

The training process of the anomaly detector is summarised in Alg. 1. Formally, we roll out the trained policy in the real world under nominal conditions. This results in a dataset $\mathcal{N} = \{I_{\pi_\phi^1}, ..., I_{\pi_\phi^n}\}$ of images collected during n nominal policy execution steps (line 2). Using $\mathcal{N}$, we fine-tune a visual feature extraction model f_I trained with the DINO training scheme [26], which uses self-distillation (line 3).[5,6] Using f_I, we extract features from $\mathcal{N}$, which results in a feature set $\mathcal{Z} = \{\mathbf{z}_k = f_I(I_{\pi_\phi^k}) \mid 1 \leq k \leq n\}$ (lines 4–6). To detect anomalies during execution, we extract features from the current observation I_t, namely $\mathbf{z}_t = f_I(I_t)$, and estimate the distance d to the nearest neighbour in $\mathcal{Z}$. We consider I_t to be anomalous if $\min_{\mathbf{z}_k \in \mathcal{Z}} d(\mathbf{z}_t, \mathbf{z}_k) > \tau^*$, where τ^* is an anomaly threshold. To determine τ^*, we collect an additional labelled dataset Y of m images with binary targets $\mathbf{y}$ (0 if fault-free, 1 if anomalous) containing both nominal and anomalous examples collected from π_ϕ (lines 7–8).[7] For each image in Y, we extract features using f_I and find the distance to the nearest neighbour in $\mathcal{Z}$ (lines 9–12). These distances D are used as candidate thresholds, such that the optimal threshold τ^* is the one that leads to the highest F-score on Y (line 13).[8]

Algorithm 1 Visual anomaly detector training using a self-supervised feature extractor

1: **function** TRAINANOMALYDETECTOR(E, π_ϕ, n, m)
2: $\quad\mathcal{N} \leftarrow$ rollOutPolicySampleImgs(π_ϕ, E, n)
3: $\quad f_I \leftarrow$ trainFeatureExtractor($\mathcal{N}$)
4: $\quad\mathcal{Z} \leftarrow \{\}$
5: $\quad$**for** $I_{\pi_\phi^k}$ **in** $\mathcal{N}$ **do**
6: $\quad\quad\mathcal{Z} \leftarrow \mathcal{Z} \cup f_I(I_{\pi_\phi^k})$
7: $\quad Y \leftarrow$ rollOutPolicySampleImgs(π_ϕ, E, m)
8: $\quad\mathbf{y} \leftarrow$ labelNominalAnomalous(Y)
9: $\quad D \leftarrow \{\}$
10: $\quad$**for** $\hat{I}^i_{\pi_\phi}$ **in** Y **do**
11: $\quad\quad\mathbf{z}_i \leftarrow f_I(\hat{I}_{\pi_\phi^i})$
12: $\quad\quad D \leftarrow D \cup \min_{\mathbf{z}_k \in \mathcal{Z}} d(\mathbf{z}_i, \mathbf{z}_k)$
13: $\quad\tau^* \leftarrow$ findOptimalThreshold($\mathbf{y}$, D)
14: $\quad$**return** ($\mathcal{Z}$, τ^*)

C. Execution Recovery

The detection of anomalies is only useful if a robot can use it to trigger a recovery process that would enable it to prevent or remedy failures. In principle, such recovery can involve modifying the manner in which the current skill is executed or even executing a different skill, especially for long-horizon tasks [27]. In this work, we only focus on recovering the currently executed skill (assuming an ergodic process [28]), with the objective of increasing the success rate of the policy executed under various anomalous situations.

Our proposed recovery strategy consists of executing three recovery stages in succession: (i) ER_1: pausing the execution for a predefined number of seconds, (ii) ER_2: locally perturbing the robot's state, where the locality is determined by a parameter σ_d, (iii) ER_3: resetting the robot's initial state and retrying the policy, where the initial state is sampled from a learned success distribution $\mathcal{F}$ that is represented as a Gaussian mixture model (GMM).

a) Pausing the execution: ER_1 is a useful recovery strategy when considering temporary anomalies; for example, in Fig. 2, pausing would enable continuing the execution if the person obstructing the target leaves the robot's view.

b) Perturbation: The assumption is that, from a state which might lead to a failure, ER_2 would move the robot locally to a randomised state from which the task execution can continue successfully. This may also overcome false positive anomaly detections: if the model outputs a false positive, pausing and retrying might give the same result, but perturbing results in a slightly different camera view, for which the anomaly detection model might give a different result. Providing a modified input to the model can be considered to be a variant of test-time augmentation [29].

c) Restarting the execution: ER_3 abandons the current trajectory and enables the robot to retry the execution, such that, by sampling from a distribution $\mathcal{F}$ of successful starting states, the robot can be reinitialised to another state that is likely to lead to a successful execution (as illustrated in Fig. 1). $\mathcal{F}$ is learned in an offline learning stage from a dataset P of initial states that have resulted in successful policy execution. This sampling strategy is inspired by the execution model proposed in [30], but we use it without a relational model and replace the Gaussian process that is used there by a mixture model, similar to [31], for simplicity.

A summary of the policy monitoring and recovery method is given in Alg. 2. The recovery stages are attempted in the above order, and the robot proceeds to the next stage if the anomaly persists; this imposes an order based on their execution costs. The recovery is executed until one of the behaviours leads to success or a maximum episode length $h_{\max}$ (from the start of the execution) is reached. This means that the recovery actions are counted as execution steps, which prevents the robot from getting stuck in an infinite recovery loop.

[5]We fine-tune f_I rather than use a pretrained model because the performance of anomaly detection was worse when using a pretrained model.

[6]Standard image augmentation, such as cropping, colour jitter, and blurring, is used for improved generalisation of the fine-tuned model.

[7]The threshold could, in fact, be determined based only on the statistics of the nominal data. The use of anomalous data only provides tighter constraints on the optimisation, but is in principle not required.

[8]In practice, the acceptable value of the F-score may depend on the task requirements. Ideally, a designer should verify that the score is high enough to avoid using the detector in cases where its performance is lower than desired.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Algorithm 2 Policy execution with anomaly detection and recovery. `recover` attempts the recovery behaviours described in the text in succession.

function EXECUTEPOLICY(E, π_ϕ, n, $\mathcal{Z}$, τ^*, $h_{\max}$)
  $done \leftarrow$ **false**, $h \leftarrow 0$
  while not $done$ **and** $h < h_{\max}$ **do**
    $h \leftarrow h + 1$
    $\mathbf{a}_t \leftarrow \pi_\phi(\mathbf{s}_t, \mathbf{c}_t)$
    $I_{t+1} \leftarrow$ `applyAction`$(\mathbf{a}_t)$
    $\mathbf{z}_{t+1} \leftarrow f_I(I_{t+1})$
    $anomaly \leftarrow \min_{\mathbf{z}_k \in \mathcal{Z}} d(\mathbf{z}_{t+1}, \mathbf{z}_k) > \tau^*$
    if $anomaly$ **then**
      `recover`$((ER_1, ER_2, ER_3), h)$
    $done \leftarrow$ `checkTermination`(π_ϕ)

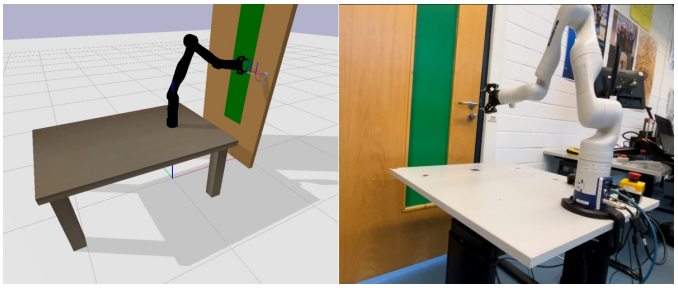


Fig. 3: Experiment setup of the door handle reaching task in a simulated environment (left) and a real environment (right)

IV. EVALUATION

We evaluate our integrated framework for policy execution, anomaly detection, and recovery on two different tasks and robots: (i) a task of reaching a door handle using a Kinova Gen3[9] manipulator with a Robotiq 2F-85 gripper, and (ii) a task of putting an object in a bowl with a robot equipped with a UFactory xArm 6[10] manipulator that has a parallel jaw gripper. In this section, we first analyse the individual elements of our framework on the door handle reaching task, and then demonstrate the generalisability of our method on the object placing task.

A. Execution Policy Learning

For reaching a door handle, we learn a policy in a PyBullet-based[11] simulated environment that is illustrated in Fig. 3, and perform sim-to-real transfer on a door in our lab. During the experiment, the manipulator is fixed on a table, as illustrated in Fig. 3. We use the built-in RealSense D410 camera on the robot's wrist to observe the environment; examples of views from this camera are shown in Fig. 2. For simplicity, the termination of the policy during the evaluation is performed manually by the experimenter.

The policy is a multilayer perceptron that takes a 266-dimensional input that is a concatenation of (i) a 256-dimensional image feature vector extracted from a NatureCNN [32] model, (ii) a 7-dimensional joint position vector, and (iii) a 3-dimensional end effector position. In this work, we use Proximal Policy Optimization (PPO) [33] for learning the policy, which is an on-policy algorithm that is commonly used in the literature. Training is performed in an end-to-end fashion, namely the feature extractor is trained along with the policy. PPO trains both a policy and a value network, such that both networks have the same design, and the feature extractor is shared between both networks.

[9]https://www.kinovarobotics.com/product/gen3-robots
[10]https://www.ufactory.cc/xarm-collaborative-robot/
[11]https://pybullet.org/

TABLE I: Success rate of the transferred policy (out of 25 executions) without introducing anomalies during the process

No domain randomisation	4%
With domain randomisation	64%

During learning, the initial state of the robot is randomly selected within $4cm$ of a predefined point from which we could ensure that the robot has at least a partial view of the door handle. We use the PPO implementation in Stable Baselines3 [34], where we interface our simulation through a Gymnasium [35] environment. For training, we use a shaped reward function that we define as

$$R = w_s \mathbb{1}_{\text{success}} + w_c \mathbb{I}_{\text{collision}} + R_{\text{step}} + R_{\text{distance}}$$

where $w_s = 15$ (rewards the agent if it reaches the door handle), $w_c = -5$ (punishes the agent if it collides with an obstacle), $R_{\text{step}} = -0.5$ for every step (to encourage the robot to move towards the handle quickly), and $R_{\text{distance}} = \alpha \frac{d_{t-1} - d_t}{d_0}$ rewards the robot for making larger steps towards the goal. Here, d_0 is the initial distance to the goal, d_{t-1} and d_t are the distances at the previous and current steps, respectively, and we use $\alpha = 10$ as a weight. To simplify the training process, we reduce the action space and only work with the end effector positions, but no orientations. Under this setup, we train the policy with a maximum allowed episode length of 100 steps, for a total of 80,000 steps, at which point the return has converged.

The results of the evaluation of the transferred policy over 25 trials are shown in Tab. I. These results demonstrate that the policy can be transferred to the real environment with reasonable success (provided that domain randomisation is used during training). However, the initial position has a significant effect on the success rate and sometimes leads to unwanted deviations of the trajectory, which indicates a need for a more robust execution process.

B. Anomaly Detection

To train our anomaly detector, we collected a set of 3,700 real-world image frames, which combine nominal policy executions and additional manual teleoperation examples of the manipulator for a larger variety. During the policy execution, the initial position of the robot was varied manually. For feature extraction, we fine-tune a ResNet-50 backbone [36] (pretrained on ImageNet), using the DINO training scheme[12], such that we have a 2048-dimensional feature vector.[13] We use the Euclidean distance for detecting anomalies.

[12]We use the Lightly library: https://github.com/lightly-ai/lightly
[13]We use the trained student from DINO for feature extraction.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

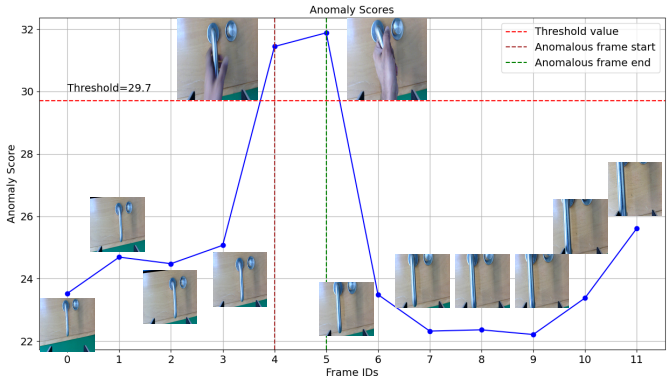


Fig. 4: Illustration of the results of the anomaly detection process during policy execution in the door handle reaching task. The anomaly here is a temporary hand obstruction as illustrated in Fig. 2 (successfully detected in this case).

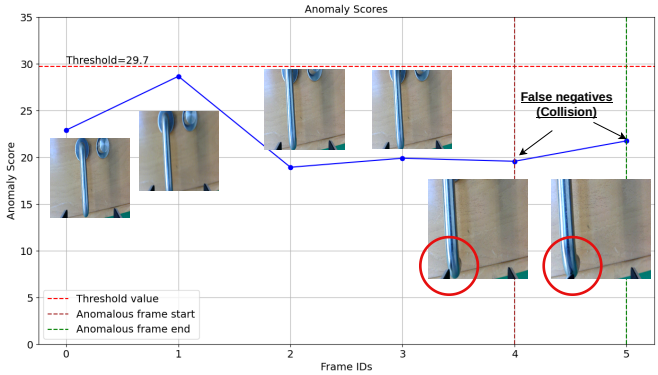


Fig. 5: An example of a false negative detection in the door handle reaching task, where a collision of the end effector with the handle is not flagged as an anomaly.

For selecting the anomaly detection threshold τ^*, we collected and labelled a dataset of 692 frames, out of which 20% were anomalous. The introduced anomalies were similar to those in Fig. 2, namely they included human hand obstructions, obstructions by another object, as well as trajectory deviations. As alluded to by the results in Tab. I, deviations sometimes occur during nominal deployment (due to observational noise), but they were also manually induced by starting the policy execution in a position where the handle is not visible.

In Fig. 4 and Fig. 5, we illustrate example detections of the anomaly detection model (using $\tau^* = 29.7$).[14] A successful detection of a human hand obstruction is illustrated in Fig. 4, while Fig. 5 illustrates a false negative case where a collision with the handle is not detected as an anomaly.

To evaluate the detector quantitatively, we collected and labelled an additional set of 467 frames during policy execution (96 anomalous and 371 nominal). For this data, the detector with the optimal threshold had a precision of 73.8% and recall of 82.3%, resulting in an F-score of 77.8%. The detector's runtime is dependent on the used hardware and the size of $\mathcal{Z}$. For our evaluation, the detector runs at a frequency of 15Hz on a laptop with an NVIDIA RTX 4060 GPU.

[14]We use the precision-recall implementation in scikit-learn to find τ^*: https://scikit-learn.org/stable/

TABLE II: Success rate of the policy in cases where anomalies where introduced (out of 20 trials)

Evaluation mode	Success rate
Policy without anomaly detection and recovery ($\tilde{\pi}_\phi$)	0%
Failure-aware policy (π_ϕ)	**75%**

TABLE III: Number of occurrences of the individual recovery strategies during the failure-aware policy evaluation in the door handle reaching task (over all 20 trials)

Strategy	Number of attempts	Strategy successes
Pausing (ER_1)	48	3
Perturbation (ER_2)	45	3
Sampling (ER_3)	42	9

C. Execution Recovery

For the recovery, with ER_1, the robot sleeps for 4–5s; we consider this to be a reasonable waiting time, particularly for external obstructions. With ER_2, we perturb the arm's position with a value of $\sigma_d = 2cm$; a small value was used to prevent the robot from straying far from its current trajectory and introducing other anomalies, such as collisions. With ER_3, we reset the robot's initial position; to train $\mathcal{F}$, we collected 45 starting positions from which the execution policy could be executed successfully.[15] Here, we fit multiple GMMs and choose the number of components that minimises the Bayesian Information Criterion (BIC) [37] (for the collected data, a single GMM component was found to be optimal).

To quantitatively evaluate our method, we compare two versions of our policy: (i) a baseline policy $\tilde{\pi}_\phi$ without the anomaly detection and recovery, and (ii) the failure-aware policy π_ϕ with the integrated anomaly detection and recovery. For the evaluation, we performed 20 trials of both policies from various starting positions, introducing various anomalies during the process, just as above. The results of the evaluation are shown in Tab. II. As can be seen here, $\tilde{\pi}_\phi$ was unable to succeed in any of the trials where anomalies were introduced, while π_ϕ was able to recover from a large number of anomalies and succeeded in most of the trials. A few of these trials failed as well, mostly due to false negative detections that led to collisions with the handle.

In the trials with π_ϕ, we also evaluate the effectiveness of the individual recovery strategies. The results of this evaluation are shown in Tab. III. In most cases that were recognised as anomalies, ER_1 and ER_2 were insufficient for recovery, and the robot executed ER_3, which was successful in significantly more cases. The recovery was sometimes triggered by false positive detections, which explains the large number of recovery attempts; we particularly observed false positives due to blurry frames that were caused by the robot's motion.

D. Generalisation to a Pretrained Policy Model

To show the policy-agnostic nature of our method, we apply it to a scenario in which a robot uses a pretrained general-purpose policy π_ϕ^G to put a spherical, apple-like object into a bowl on a table in front of the robot; at the start of each trial, the robot is already holding the object. We work with a bimanual mobile manipulator for this task, although only

[15]We use the implementation in scikit-learn for learning a GMM model.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Fig. 6: Experiment setup of the object placing task (left) and typical wrist camera view (right)

TABLE IV: Success rate of the pretrained policy on the task of putting an object in a bowl (out of 25 executions)

Nominal execution (without anomalies)	56%
Execution in anomalous cases w/o detection and recovery	48%
Execution in anomalous cases with detection and recovery	**60%**

one of the arms is used. The setup is illustrated in Fig. 6. For execution, the robot uses a pretrained *Octo-Base* model, with a wrist camera image as state input and the prompt "You are holding an apple in your hand. Put the apple in the bowl." as a goal prompt. To ensure the stability of the gripper command provided by the policy, we use a majority filter on the five latest commands, namely we open the gripper if at least three of those indicate that it should be opened.[16] We train the anomaly detection feature extractor using 2,440 nominal images, and determine $\tau^* = 50.0$ using a validation set of 470 images.

We evaluate the success rate under three execution conditions (25 trials each): (i) under nominal conditions, (ii) in the presence of anomalies, but without using our method, and (iii) in anomalous conditions using our method. In the latter two cases, we manually introduce a variety of anomalies, such as covering the camera or the bowl, removing the bowl while the robot is moving towards it, turning the bowl upside down, or shaking the bowl while the robot is moving towards it. The anomalies are introduced either directly at the start of the execution or during it, and are removed after $5 - 8s$. For $\mathcal{F}$, we fit a GMM with three components using the successful starting positions under nominal conditions.

The results of this evaluation are shown in Tab. IV. As can be seen, the nominal, zero-shot success rate of the policy is slightly higher than 50% (most of the failures in the nominal executions are due to the model releasing the object before reaching the bowl or due to the object bouncing off of the bowl's edge). When anomalies are introduced in the execution (and our method is not used), the success rate drops below 50%, although it was observed that the policy itself has some recovery ability, presumably due to emergent behaviour (typically manifested by the robot moving slowly while an anomaly was present). When using our method, the success rate in the anomalous cases increases to 60%, which is comparable with the execution success of the nominal policy (given that the anomalies are eventually removed, the remaining failures are due to the nominal policy). This confirms that our method can enable a robot to successfully handle unforeseen anomalies, even though eventually the robot may fail due to limitations of the policy itself. In particular, the recovery behaviours were invoked in almost all anomalous cases, except when the bowl was moved to introduce a motion blur; this was seemingly not enough of a disturbance to be detected as an anomaly.

[16]The filter parameters were determined empirically.

TABLE V: Number of occurrences of the individual recovery strategies during the failure-aware policy evaluation in the task of putting an object in a bowl (over all 25 trials)

Strategy	**Number of attempts**	**Strategy successes**
Pausing (ER_1)	37	3
Perturbation (ER_2)	31	2
Sampling (ER_3)	29	7

Tab. V shows the number of times the individual recovery strategies were invoked, and how often the execution was successful after their application. These results are consistent with those in the first task, namely the recovery behaviours are sometimes triggered due to false positive detections, and ER_3 is the most successful recovery behaviour.

V. DISCUSSION AND CONCLUSION

In this letter, we proposed a method that integrates a learned robot policy with visual anomaly detection and subsequent recovery. Here, data from nominal policy executions is used to fine-tune a self-supervised DINO-based image feature extractor. Anomalies are then detected based on the distance to the nearest neighbour in the dataset of nominal execution features, where the detection threshold is optimised using a dataset of nominal and anomalous images. Anomalies trigger a recovery process that involves the sequential execution of different recovery behaviours (pausing the execution, locally perturbing the robot's state, and resetting the robot's state and retrying the execution). For resetting, states are sampled from a success model (in the form of a GMM) that is learned from nominal executions.

We evaluated our method on the task of reaching a door handle using a dedicated policy (with sim-to-real transfer) and the task of placing an object in a bowl using a pretrained general-purpose language-conditioned policy. The results demonstrate that integrating anomaly detection and recovery is important for improving the robustness of a learned policy, particularly when external anomalies are introduced, but, not surprisingly, the nominal policy's robustness is just as important for overall success.

We have various ideas for future work. To simplify the evaluation procedure using the dedicated policy, we manually determined if the execution should terminate, but using a complete skill model with initiation and termination classifiers [38] is necessary for integration of the policy in long-horizon tasks [39]. For training this policy, we used a PyBullet-based simulation due to the simplicity of customisation, but using a photorealistic environment as in [40] can improve the sim-to-real policy transfer. In this work, we only focused on visual anomalies, as visual data is a rich source of information and visual feature extraction models can be considered to be quite mature. For detecting a larger class of anomalies that are not effectively represented by visual data, for instance collisions as in [41], it would be useful to integrate multimodal data, such as force measurements or audio, similar to [42], [43]. Related to

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

this, in applications with concrete requirements on the robustness of the anomaly detector, it may be desirable to replace our detector with one that complies to those requirements more precisely; this would not compromise the operation of our system. Finally, as our results show, resetting the execution seems to be the most successful recovery strategy, but, for recovery planning, it would be useful to study the strategies in isolation and identify cases in which each of them is particularly effective.

REFERENCES

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annu. Review of Control, Robotics, and Auton. Systems*, vol. 3, pp. 297–330, 2020.

[2] O. Kroemer, S. Niekum, and G. Konidaris, "A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms," *Journal Machine Learning Research*, vol. 22, pp. 1–82, 2021.

[3] C. Tang *et al.*, "Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes," in *Proc. of the AAAI Conf. on Artificial Intelligence*, vol. 39, no. 27, 2025, pp. 28 694–28 698.

[4] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 7327–7334, 2022.

[5] M. J. Kim *et al.*, "OpenVLA: An Open-Source Vision-Language-Action Model," in *8th Annu. Conf. on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=ZMnD6QZAE6

[6] A. O'Neill *et al.*, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2024, pp. 6892–6903.

[7] O. M. Team *et al.*, "Octo: An Open-Source Generalist Robot Policy," in *Proc. Robotics: Science and Systems (RSS)*, 2024.

[8] OpenAI *et al.*, "Learning Dexterous In-Hand Manipulation," *The Int. Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[9] J. Tobin *et al.*, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[10] K. Rao *et al.*, "RL-CycleGAN: Reinforcement Learning Aware Simulation-to-Real," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 154–11 163.

[11] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger, "Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders," in *Proc. 14th Int. Joint Conf. Comput. Vision, Imaging and Comput. Graphics Theory and Applications (VISIGRAPP)*, 2019, pp. 372–380.

[12] D. Gong *et al.*, "Memorizing Normality to Detect Anomaly: Memory-Augmented Deep Autoencoder for Unsupervised Anomaly Detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vision (ICCV)*, 2019, pp. 1705–1714.

[13] L. Bergman, N. Cohen, and Y. Hoshen, "Deep Nearest Neighbor Anomaly Detection," *CoRR*, vol. abs/2002.10445, 2020. [Online]. Available: https://arxiv.org/abs/2002.10445

[14] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, 2021, pp. 475–489.

[15] A. Lagrassa, S. Lee, and O. Kroemer, "Learning Skills to Patch Plans Based on Inaccurate Models," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2020, pp. 9441–9448.

[16] C. M. Hung, L. Sun, Y. Wu, I. Havoutis, and I. Posner, "Introspective Visuomotor Control: Exploiting Uncertainty in Deep Visuomotor Control for Failure Recovery," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2021, pp. 6293–6299.

[17] K. Lee, K. Saigol, and E. A. Theodorou, "Early Failure Detection of Deep End-to-End Control Policy by Reinforcement Learning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 8543–8549.

[18] B. Thananjeyan *et al.*, "Recovery RL: Safe Reinforcement Learning With Learned Recovery Zones," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 3, pp. 4915–4922, 2021.

[19] A. Reichlin, G. L. Marchetti, H. Yin, A. Ghadirzadeh, and D. Kragic, "Back to the Manifold: Recovering from Out-of-Distribution States," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2022, pp. 8660–8666.

[20] P. Mitrano, D. McConachie, and D. Berenson, "Learning where to trust unreliable models in an unstructured world for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, p. eabd8170, 2021.

[21] C. Agia *et al.*, "Unpacking Failure Modes of Generative Policies: Runtime Monitoring of Consistency and Progress," in *Proc. Conf. Robot Learning (CoRL)*, 2024.

[22] Z. Liu, A. Bahety, and S. Song, "REFLECT: Summarizing Robot Experiences for Failure Explanation and Correction," in *Proc. Conf. Robot Learning (CoRL)*, 2023.

[23] P. Varin, L. Grossman, and S. Kuindersma, "A Comparison of Action Spaces for Learning Manipulation Tasks," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2019, pp. 6015–6021.

[24] C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *Int. Journal Robotics Research*, 2024.

[25] S. Thoduka, J. Gall, and P. G. Plöger, "Using Visual Anomaly Detection for Task Execution Monitoring," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2021, pp. 4604–4610.

[26] M. Caron *et al.*, "Emerging Properties in Self-Supervised Vision Transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vision (ICCV)*, 2021, pp. 9630–9640.

[27] S. Mukherjee, C. Paxton, A. Mousavian, A. Fishman, M. Likhachev, and D. Fox, "Reactive Long Horizon Task Execution via Visual Skill and Precondition Models," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2021, pp. 5717–5724.

[28] T. M. Moldovan and P. Abbeel, "Safe Exploration in Markov Decision Processes," in *Proc. 29th Int. Conf. Machine Learning (ICML)*, 2012, pp. 1451–1458.

[29] M. Kimura, "Understanding Test-Time Augmentation," in *Neural Information Processing*, 2021, pp. 558–569.

[30] A. Mitrevski, P. G. Plöger, and G. Lakemeyer, "A Hybrid Skill Parameterisation Model Combining Symbolic and Subsymbolic Elements for Introspective Robots," *Robotics and Auton. Systems*, vol. 161, pp. 104 350:1–22, Mar. 2023.

[31] A. K. Bozcuoğlu, Y. Furuta, K. Okada, M. Beetz, and M. Inaba, "Continuous Modeling of Affordances in a Symbolic Knowledge Base," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2019, pp. 5452–5458.

[32] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[34] A. Raffin *et al.*, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[35] M. Towers *et al.*, "Gymnasium: A Standard Interface for Reinforcement Learning Environments," *CoRR*, vol. abs/2407.17032, 2024. [Online]. Available: https://arxiv.org/abs/2407.17032

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[37] G. Schwarz, "Estimating the Dimension of a Model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[38] G. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez, "From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning," *Journal Artificial Intelligence Research*, vol. 61, no. 1, pp. 215–289, Jan. 2018.

[39] D. Sliwowski and D. Lee, "ConditionNET: Learning Preconditions and Effects for Execution Monitoring," *IEEE Robotics and Automation Letters (RA-L)*, vol. 10, no. 2, pp. 1337–1344, 2025.

[40] S. Nasiriany *et al.*, "RoboCasa: Large-Scale Simulation of Household Tasks for Generalist Robots," in *Proc. Robotics: Science and Systems (RSS)*, 2024.

[41] E. Sharma, C. Henke, A. Mitrevski, and P. G. Plöger, "Adaptive Compliant Robot Control with Failure Recovery for Object Press-Fitting," in *Proc. European Conf. Mobile Robots (ECMR)*, 2023, pp. 1–7.

[42] A. Inceoglu, E. E. Aksoy, A. Cihan Ak, and S. Sariel, "FINO-Net: A Deep Multimodal Sensor Fusion Framework for Manipulation Failure Detection," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2021, pp. 6841–6847.

[43] S. Thoduka, N. Hochgeschwender, J. Gall, and P. G. Plöger, "A Multimodal Handover Failure Detection Dataset and Baselines," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2024, pp. 17 013–17 019.