CCDP: Model-free Failure Recovery via Guided Diffusion Sampling

Amirreza Razmjoo[1,2], Sylvain Calinon[1,3], Michael Gienger[2], and Fan Zhang[2]

***Abstract*— Working in constrained environments means that failures are often inevitable, so robots must be able to recover from them. Typical recovery approaches require explicit models of the underlying task, either during learning or reproduction, to accommodate different possibilities and replan accordingly. However, such models are not always available, especially in imitation learning (IL), where one of the main advantages is precisely to avoid explicit environment/task modeling and rely instead on demonstration data. We present CCDP (Composition of Conditional Diffusion Policies), a method that considers failures during inference and guides the sampling steps of diffusion policies to avoid previously failed actions. Remarkably, CCDP relies solely on successful demonstrations: it infers recovery actions without additional exploratory behavior or a high-level controller. We validate our approach on several tasks, including door opening with unknown directions, object manipulation, and button searching, and show that it consistently outperforms standard baselines. Supplementary material is available at: https://hri-eu.github.io/ccdp/. This paper is a condensed summary of our main publication at *IROS 2025* [1].**

I. INTRODUCTION

Recent advances in imitation learning—including Implicit Behavior Cloning [2] and diffusion or flow-matching policies [3], [4]—have shown strong ability to capture rich, multimodal behaviors directly from demonstrations. However, these approaches still face a key limitation: when a sampled action does not succeed, the policy has no built-in mechanism for recovery. Since imitation learning typically provides only successful demonstrations, the challenge is to design policies that avoid blindly repeating failed actions and instead redirect sampling toward alternative strategies already present in the demonstrations, even if those strategies occur only rarely.

Many recovery strategies rely on additional resources such as simulators [5], reasoning via large models [6], [7], [8], or direct expert input [9]. Other works employ hierarchical schemes where a high-level planner selects primitives or sub-policies [5], [10]. While effective in some cases, these methods increase complexity and often scale poorly as the number of options grows. In contrast, our work focuses on a simpler setting: enabling the low-level policy itself to adapt after failures, without requiring explicit models or a separate reasoning layer. The idea is inspired by everyday experience—when searching for a light switch in the dark, failed attempts mainly tell us where *not* to look, and that information alone is enough to guide the next action.

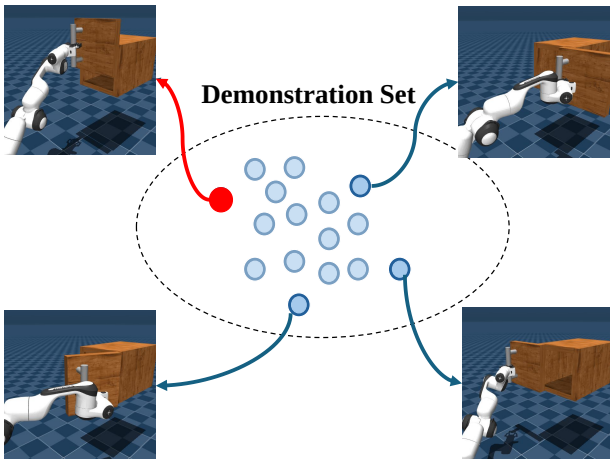


Fig. 1. A diverse demonstration set, featuring multiple task variations, is provided to the robot. In the event of a failure, the robot switches to alternative variations rather than repeatedly sampling the same actions.

We aim to train policies that adapt based on past observations and actions [11], [12]. Two challenges arise: (i) recovery must be learned without explicit failure data, since demonstrations usually only show successes, and (ii) the number of failures encountered at runtime can vary, making it difficult to design a fixed-size input. To address the first issue, we assume that while an attempted action may fail, suitable alternatives already exist within the demonstration set. By analyzing these demonstrations offline, we identify diverse actions and train a model that encourages the policy to choose options sufficiently different from those that just failed. To handle the second issue, we build on the idea of composing multiple models [13], learning recovery strategies for individual failures and combining them at inference time into a distribution that accounts for all observed failures.

We present the Composition of Conditional Diffusion Policies (CCDP), which extends the diffusion policy framework [3] by conditioning the sampling process on failure information. When a failure is detected, CCDP steers the policy toward actions that differ from the unsuccessful attempt, reducing repeated errors. To keep recovery manageable, we decompose the problem into smaller diffusion modules that can be combined at inference, avoiding the need for long histories or explicit data categorization. This modular view lowers the labeling burden while still capturing diverse strategies observed in demonstrations.

Contributions: The main contributions of this work are:

[1]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland amirreza.razmjoofard@epfl.ch
[2]Honda Research Institute Europe GmbH, Germany {michael.gienger, Fan.zhang}@honda-ri.de
[3]Idiap Research Institute, Switzerland Sylvain.calinon@idiap.ch

- A modular diffusion policy formulation that improves controllability and extends [3];
- A lightweight recovery strategy that requires only successful demonstrations, with no labels or simulators;
- Empirical validation across tasks such as door opening, button search, and object manipulation, showing advantages over existing baselines.

II. Preliminarily

A. Diffusion Policy

Let $\mathbf{a}_t \in \mathbb{R}^{d_u}$ denote the action at time t, $\mathbf{x}_t \in \mathbb{R}^{d_s}$ the state, and $\mathbf{h}_t^H = [\mathbf{a}_{t-H:t-1}^\top, \mathbf{x}_{t-H:t-1}^\top]^\top$ the history of the previous H actions and states. Diffusion Policy has been proposed in [3] to model the multimodal action distribution in robot imitation learning using Denoising Diffusion Probabilistic Models (DDPMs) [14] and Denoising Diffusion Implicit Models (DDIMs) [15]. Diffusion Policy regresses a noise prediction function $\boldsymbol{\varepsilon}_\theta(\mathbf{h}_t^H, \mathbf{a}_t + \boldsymbol{\varepsilon}^k, k) = \boldsymbol{\varepsilon}^k$ with a network $\boldsymbol{\varepsilon}_\theta$ parameterized by θ. During training, the current history and actions $(\mathbf{h}_t^H, \mathbf{a}_t)$ are sampled from the demonstrated dataset. Random noise $\boldsymbol{\varepsilon}^k$, conditioned on a randomly sampled denoising step k, is added to $\mathbf{a}_t$. Thus, the loss can be described as

$$\mathcal{L} = \left\| \boldsymbol{\varepsilon}_\theta(\mathbf{h}_t^H, \mathbf{a}_t + \boldsymbol{\varepsilon}^k, k) - \boldsymbol{\varepsilon}^k \right\|^2.$$

During inference, given the history $\mathbf{h}_t^H$, Diffusion Policy executes a sequence of K denoising steps starting from random samples actions $\mathbf{a}_t^k \sim \mathcal{N}(0, 1)$ to generate target robot actions $\mathbf{a}_t^0$. This inverse process can be defined as

$$\mathbf{a}^{k-1} = \alpha \left(\mathbf{a}_t^k - \gamma \boldsymbol{\varepsilon}_\theta(\mathbf{h}_t^H, \mathbf{a}_t^k, k) + \boldsymbol{\varepsilon} \right),$$

where α, γ are the parameters of the noise schedule, $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 I)$. The action $\mathbf{a}_t^0$ can be sampled from the demonstration data or expert policy $\pi : \mathbf{h}_t^H \mapsto \mathbf{a}_t$.

III. Methodology

A. Problem Definition

Given a dataset of M successful demonstrations $\mathcal{D} = \{(\mathbf{a}_t, \mathbf{x}_t, \mathbf{h}_t^H)_i\}_{i=1}^M$, our objective is to learn a diffusion policy that models the conditional distribution

$$\mathbf{a}_t \sim p_\pi(\mathbf{a}_t \mid \mathbf{x}_t, \mathbf{h}_t^H, \mathbf{z}_{1:N}^f), \tag{1}$$

where $\mathbf{z}_i^f = \mathbf{z}(\mathbf{a}_i^f, \mathbf{x}_i^f)$ extracts key features from the i-th failure's action $\mathbf{a}_i^f$ and state $\mathbf{x}_i^f$, and N denotes the total number of previous failures. we discuss different possibilities for $\mathbf{z}$ in Sec. III-B.1.

Inspired by [13], we propose decomposing (1) into several simpler sub-problems. Hence, (1) can be rewritten as

$$p_\pi(\mathbf{a}_t \mid \mathbf{x}_t, \mathbf{h}_t^H, \mathbf{z}_{1:N}^f) \propto p_a(\mathbf{a}_t) \frac{p_s(\mathbf{a}_t \mid \mathbf{x}_t)}{p_a(\mathbf{a}_t)} \frac{p_h(\mathbf{a}_t \mid \mathbf{h}_t^H)}{p_a(\mathbf{a}_t)} \prod_{i=1}^N \frac{p_z(\mathbf{a}_t \mid \mathbf{z}_i^f)}{p_a(\mathbf{a}_t)}. \tag{2}$$

Using the diffusion model framework, we initialize with a noisy sample and iteratively denoise through the reverse diffusion process

$$p(\mathbf{a}_t^{k-1} \mid \mathbf{a}_t^k) = \mathcal{N}\Big(\alpha\Big(\mathbf{a}_t^k - \gamma\, \hat{\boldsymbol{\varepsilon}}(\mathbf{a}_t^k, k)\Big), \sigma_t^2\, \mathbf{I}\Big). \tag{3}$$

According to (2), we decompose the denoising term $\hat{\boldsymbol{\varepsilon}}(\mathbf{a}_t^k, k)$ as follows:

$$\begin{aligned} \hat{\boldsymbol{\varepsilon}}(\mathbf{a}_t^k, k) = \boldsymbol{\varepsilon}_a(\mathbf{a}_t, k) &+ w_s\Big(\boldsymbol{\varepsilon}_s(\mathbf{a}_t, \mathbf{x}_t, k) - \boldsymbol{\varepsilon}_a(\mathbf{a}_t, k)\Big) \\ &+ w_h\Big(\boldsymbol{\varepsilon}_h(\mathbf{a}_t, \mathbf{h}_t^H, k) - \boldsymbol{\varepsilon}_a(\mathbf{a}_t, k)\Big) \\ &+ \sum_{i=1}^N w_z^i\Big(\boldsymbol{\varepsilon}_z(\mathbf{a}_t, \mathbf{z}_i^f, k) - \boldsymbol{\varepsilon}_a(\mathbf{a}_t, k)\Big). \end{aligned} \tag{4}$$

Here, w_s, w_h, and w_z^i are positive coefficients associated with the state, history, and failure key features, respectively. Namely:

- $\boldsymbol{\varepsilon}_a(\mathbf{a}_t, k)$ encourages sampling actions that are similar to those observed in the demonstrations.
- $w_s\Big(\boldsymbol{\varepsilon}_s(\mathbf{a}_t, \mathbf{x}_t, k) - \boldsymbol{\varepsilon}_a(\mathbf{a}_t, k)\Big)$ steers the actions toward those that match the current state of the robot and its environment.
- $w_h\Big(\boldsymbol{\varepsilon}_h(\mathbf{a}_t, \mathbf{h}_t^H, k) - \boldsymbol{\varepsilon}_a(\mathbf{a}_t, k)\Big)$ promotes temporal continuity by encouraging the system to follow the action history.
- $w_z^i\Big(\boldsymbol{\varepsilon}_z(\mathbf{a}_t, \mathbf{z}_i^f, k) - \boldsymbol{\varepsilon}_a(\mathbf{a}_t, k)\Big)$ allows the system to consider distinct failure cases and, based on the extracted failure features, steer away from regions that led to failures.

B. Recovery Model

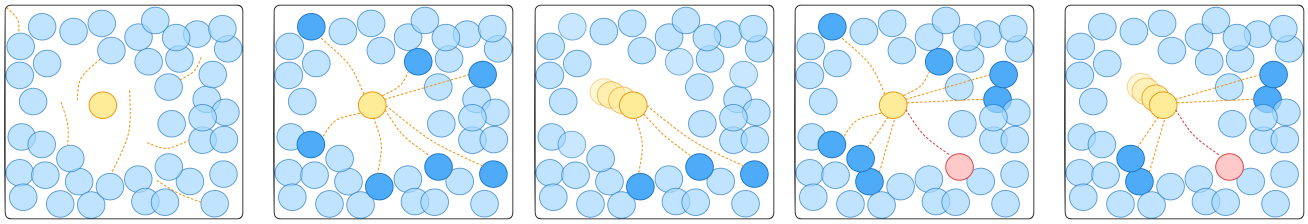
To synthesize an auxiliary dataset $\mathcal{R}$ to learn the recovery policies, we first relax the recovery definition slightly by considering all actions dissimilar to the failed one as potential recoveries. If we further assume that the cause of failure is static (i.e., repeating the same action in the same state will fail again), then we define the set of recovery actions as

$$\mathbf{a} \in \mathcal{R}(\mathbf{z}^f) \quad \text{if} \quad \begin{cases} \|\mathbf{z}(\mathbf{a}, \mathbf{x}) - \mathbf{z}(\mathbf{a}^f, \mathbf{x}^f)\|^2 > \delta_z, \\ \|\mathbf{x} - \mathbf{x}^f\|^2 < \delta_x, \end{cases} \tag{5}$$

where $\mathbf{a}^f$ is the action that led to failure, $\mathbf{x}^f$ is the corresponding state, and δ_z and δ_x are thresholds. Intuitively, if the state of the system has not changed significantly, we expect the same action to result in failure again.

Now we can construct the recovery dataset by traversing the available data and accepting every pair that satisfies (5). However, doing so directly on the main dataset can be problematic because the dataset is limited and actions are collected in various states, resulting in a sparse recovery dataset. To address this issue, we perform data synthesis: we randomly select states observed in the demonstrations and, for each state, sample multiple actions such that

$$\mathcal{D}_s(\mathbf{x}_s) = \{(\mathbf{a}, \mathbf{x}_s) \mid \mathbf{a} \sim \bar{p}^{\mathcal{D}}(\mathbf{a} \mid \mathbf{x}),\ \mathbf{x} \in \mathbf{x}_s + \boldsymbol{\xi}_x, \ \boldsymbol{\xi}_x \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})\}. \tag{6}$$



(a) $p(\mathbf{a}_t)$ (b) $p(\mathbf{a}_t|\mathbf{x}_t)$ (c) $p(\mathbf{a}_t|\mathbf{h}_t^H)$ (d) $p(\mathbf{a}_t|\mathbf{z}_i^f)$ (e) Combined

Fig. 2. An illustrative example displays samples generated from multiple distributions learned from the same demonstration sets. The yellow circle marks the current system state (which later appears more transparent as it becomes part of the history), light blue markers indicate potential samples, dark blue markers denote the generated samples, and the red marker highlights the failed sample.

Here, $\boldsymbol{\xi}_x$ denotes random noise added to the state. The corresponding noise estimator is defined as

$$\bar{\boldsymbol{\varepsilon}}(\mathbf{a}, \mathbf{x}, k) = \boldsymbol{\varepsilon}_a(\mathbf{a}, k) + w_s\Big(\boldsymbol{\varepsilon}_s(\mathbf{a}, \mathbf{x}, k) - \boldsymbol{\varepsilon}_a(\mathbf{a}, k)\Big), \quad (7)$$

where the key difference between (7) and (4) is the exclusion of history in (7), which is omitted to not bias the system toward past actions and reduce the dimensionality. Additionally, setting $w_s \leq 1$ encourages the system to explore a broader range of actions. With this recovery dataset in hand, we then apply the DDPM method to learn the corresponding distribution.

1) Failure Key Features: The function $\mathbf{z}(\cdot)$ captures the key features of failed actions. While learning these features autonomously via latent-space methods would be highly beneficial, we leave this for future work. Instead, we discuss three intuitive and practical choices for $\mathbf{z}(\cdot)$:

1) **Directly using the failed actions:** $\mathbf{z}(\mathbf{a}^f, \mathbf{x}^f) = \mathbf{a}^f$.
2) **Using the final state:** $\mathbf{z}(\mathbf{a}^f, \mathbf{x}^f) = \mathbf{x}_T^f$, where $\mathbf{x}_T^f$ is the state reached if $\mathbf{a}^f$ were executed at $\mathbf{x}^f$; this can be readily extracted from demonstrations.
3) **Action primitive:** $\mathbf{z}(\mathbf{a}^f, \mathbf{x}^f) = m$, where m is a discrete label indicating the action primitive applied to the system.

Fig. 3 summarizes the overall process in both the offline and online phases.

IV. Experiments

We evaluate CCDP against two baselines: a standard diffusion policy (DP) and a rejection-sampling variant (DP*). Demonstrations were generated in simulation using a motion planner with access to ground-truth task parameters, but these parameters were not available to the learning methods. Performance was measured by (i) overall task success and (ii) adherence to implicit preferences present in demonstrations (e.g., preferring closer baskets or lighter-hand manipulation). A failure detection signal was assumed available, but the specific detector was not part of our method.

Tasks. We considered five representative tasks: 1) *Door Opening* with unknown direction, 2) *Button Pressing* with uncertain location, 3) *Object Manipulation* with weight-dependent strategies, 4) *Object Packing* with basket capacity constraints, and 5) *Bartender* with sequential cup filling.

Findings. Across all tasks, CCDP consistently outperformed DP and matched or exceeded DP*.

- In button search and packing, CCDP avoided repeated failures more effectively.
- In manipulation tasks, CCDP balanced success rate with adherence to implicit preferences, whereas DP favored the dominant mode and DP* ignored subtle preferences.
- In sequential tasks (packing and bartender), CCDP naturally respected "nearest available" preferences without explicit rules.

V. Discussion

Our experiments show that CCDP substantially improves task success over standard diffusion policies while maintaining implicit preferences from demonstrations—something DP* often fails to preserve. The main distinction is that DP* relies on *positive forcing*, constraining sampling to predefined regions, whereas CCDP applies *negative forcing*, excluding only failed regions. This makes CCDP more flexible and better able to integrate contextual cues during execution.

A. Limitations and Future Work

The current implementation relies on manually defined failure features and fixed combination weights. More principled approaches—such as automatic feature extraction in latent space or adaptive weighting—could further improve robustness. In addition, incorporating offline exploratory mechanisms to generate richer recovery data is an important next step.

VI. Conclusion

CCDP provides a lightweight, modular way to integrate failure recovery directly into diffusion policies without requiring labels, simulators, or high-level planners. By leveraging only successful demonstrations, it achieves both higher success rates and better adherence to demonstration preferences across diverse tasks. We see this as a promising direction for building more adaptable and resilient imitation learning systems.

References

[1] A. Razmjoo, S. Calinon, M. Gienger, and F. Zhang, "Ccdp: Composition of conditional diffusion policies with guided sampling," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[2] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on robot learning*. PMLR, 2022, pp. 158–168.

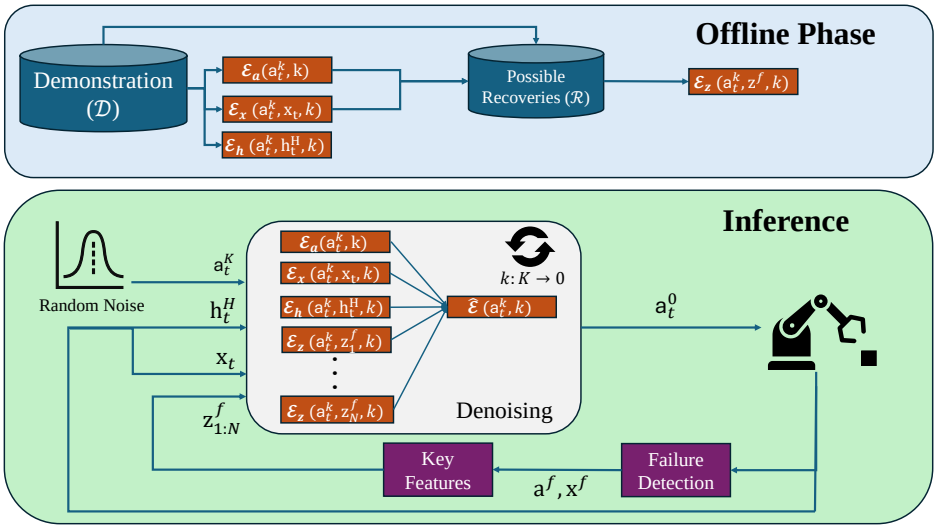
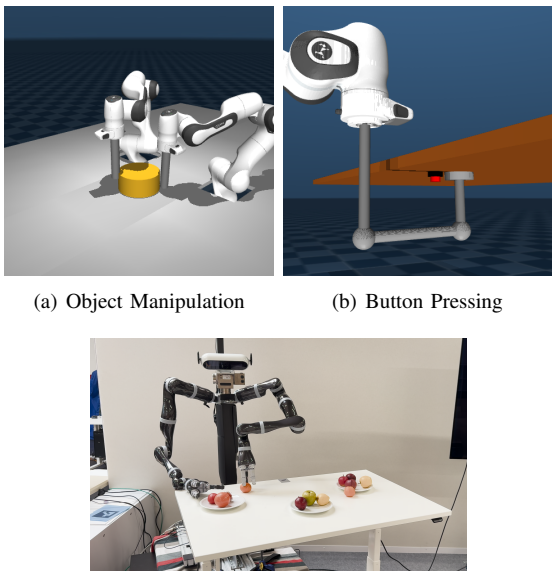


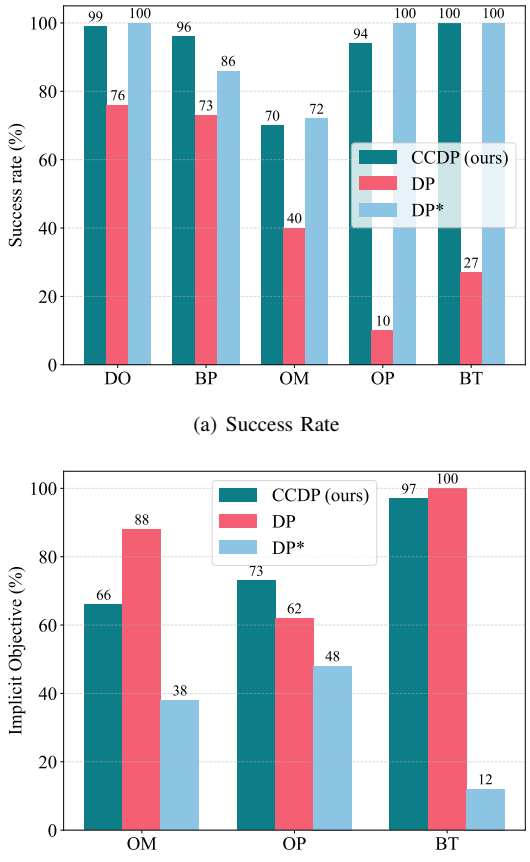
Fig. 3. Schematic overview of the proposed method illustrating the offline and inference phases.



(a) Object Manipulation (b) Button Pressing

(c) Object Packing

Fig. 4. Experimental setups: manipulation, button search, and packing. Additional tasks include door opening and bartender scenarios.



(a) Success Rate

(b) Implicit Objective

Fig. 5. Comparison with DP and DP*. CCDP improves success rates across tasks and better preserves implicit demonstration preferences.

[3] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[4] F. Zhang and M. Gienger, "Affordance-based robot manipulation with flow matching," *arXiv preprint arXiv:2409.01083*, 2024.

[5] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

[6] J. Duan, W. Pumacay, N. Kumar, Y. R. Wang, S. Tian, W. Yuan, R. Krishna, D. Fox, A. Mandlekar, and Y. Guo, "Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation," *arXiv preprint arXiv:2410.00371*, 2024.

[7] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[8] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," *arXiv preprint arXiv:2306.15724*, 2023.

[9] S. Niekum, S. Chitta, A. G. Barto, B. Marthi, and S. Osentoski, "Incremental semantically grounded learning from demonstration." in

Robotics: Science and Systems, vol. 9. Berlin, Germany, 2013, pp. 10–15 607.

[10] E. Triantafyllidis, F. Acero, Z. Liu *et al.*, "Hybrid hierarchical learning for solving complex sequential tasks using the robotic manipulation network roman," *Nature Machine Intelligence*, vol. 5, pp. 991–1005, 2023.

[11] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[12] W. Yu, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," in *Proceedings of Robotics: Science and Systems*, 2017.

[13] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum, "Compositional visual generation with composable diffusion models," in *European Conference on Computer Vision*. Springer, 2022, pp. 423–439.

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[15] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.