Implicit Contact Representations with Neural Descriptor Fields for Learning Dynamic Recovery Policies

Fan Yang[1], Sergio Aguilera Marinovic[2], Soshi Iba[2], Rana Soltani Zarrin[2], Dmitry Berenson[1]
[1]University of Michigan, Ann Arbor
[2]Honda Research Institute USA

***Abstract*—Real-world dexterous manipulation often encounters unexpected errors and disturbances, which can lead to catastrophic failures, such as dropping the manipulated object. To address this challenge, we investigate the problem of catching a falling object while it remains within grasping range and, more importantly, resetting the system to a configuration favorable for resuming the primary manipulation task. We propose Contact-Aware Dynamic Recover (CADRE), a reinforcement learning framework that incorporates a Neural Descriptor Field (NDF)-inspired module to provide an implicit contact-centric representation. Compared to methods that rely solely on object pose or point cloud input, NDFs can directly reason about finger-object correspondence and naturally adapt to different object geometries. Our experiments show that incorporating contact features improves training efficiency, enhances convergence performance for RL training, and ultimately leads to more successful recoveries. Additionally, CADRE demonstrates zero-shot generalization ability to unseen objects with different geometries.**

I. Introduction

Robots performing real-world manipulation tasks can encounter unexpected disturbances and modeling errors, which can lead to catastrophic failure, such as dropping the manipulated object. This challenge is especially severe in dexterous manipulation with a multi-fingered hand, where precise control of high-dimensional hand systems is essential to prevent dropping the object. This issue is further exacerbated in robotic systems without tactile sensing, where the absence of direct contact feedback makes it challenging to detect whether the object is being grasped firmly.

In this work, we focus on dynamic recovery, specifically, catching falling objects before irrecoverable failure occurs. Rather than improving the inherent robustness of primary manipulation policies, we propose a complementary strategy that incorporates a fallback catching policy – The robot switches from the primary manipulation policy to the catching policy when object dropping is detected. This policy is responsible for catching falling objects during failure events, and, importantly, resetting the robot system to states from which the primary manipulation can resume. This is because merely catching falling objects is not enough to ensure successful recovery. Specifically, while power grasps can effectively catch a large variety of objects [18, 8], the primary manipulation task often requires specific grasp types, such as precision grasps [34, 42, 4]. Switching from power grasps to precision grasps presents significant challenges. Therefore, our recovery policy is designed to achieve grasp configurations that support the contact requirements of the primary manipulation task. Additionally, to ensure practical applicability, the recovery policy must be able to adapt to objects with different geometries.

To address the challenges mentioned above, we present Contact-Aware Dynamic Recovery (CADRE), a reinforcement learning approach that incorporates a contact-centric representation in its observation space. Our work is based on the importance of contact in dexterous manipulation [16, 6, 3, 9]. CADRE is built upon the insight that maintaining consistent contact behaviors across different object geometries is one of the fundamental factors for successful generalization. The contact information is derived from Neural Descriptor Fields(NDFs) [32, 33]. NDF captures the geometric correspondence between 3D coordinates and the object point clouds. CADRE leverages NDF features as implicit contact information for dexterous manipulation. This approach provides comprehensive contact modeling for both regions that should be in contact (e.g., fingertips) and regions where contact should be avoided (e.g., palm).

Our main contributions are summarized as follows: (1) We develop an NDF-based implicit contact representation for contact-rich dexterous manipulation that effectively captures the geometric correspondence between the hand and the manipulated object. (3) We propose the problem of recovery through catching, where a robot must not only catch the falling object but also achieve grasp configurations from which the robot can seamlessly resume the primary manipulation task. (3) We present a reinforcement learning framework for dynamic recovery that leverages the contact representations to achieve successful grasps and favorable states for subsequent manipulation tasks. (4) We demonstrate empirically that our contact representations enable effective generalization across different geometries in our dynamic recovery tasks.

Our experimental results demonstrate that our contact-aware approach significantly improves recovery performance on training objects while enabling zero-shot generalization to unseen objects of the same type but with different geometries (e.g. various sizes of screwdriver). Please see more details at https://cadrecatching.github.io/.

[1]This work was supported by Honda Research Institute USA.

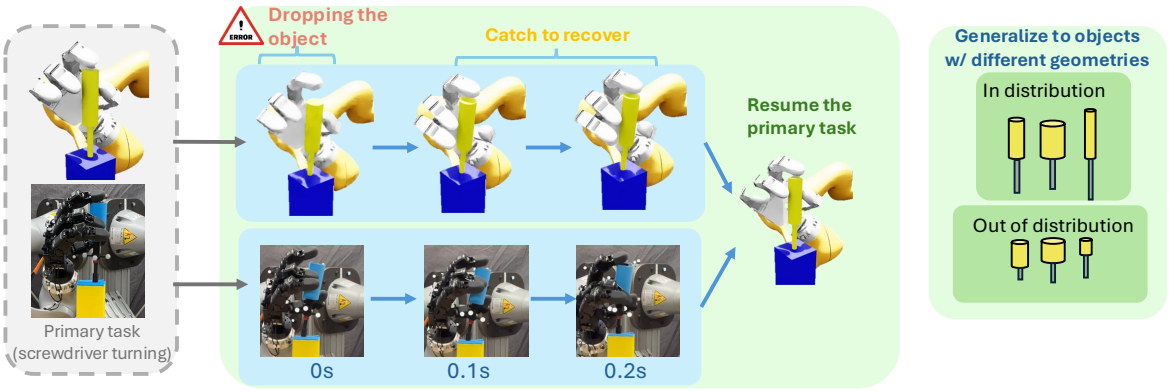


Fig. 1: CADRE recovers from object-dropping failures by catching the object and resetting to states favorable for resuming the primary manipulation task. Leveraging contact information from a pre-trained NDF model, it generalizes recovery behaviors to objects with different geometries.

II. Related Work

A. Dynamic Manipulation

Dynamic manipulation involving rapid robot and object motion has been a popular research area in robotics [15, 22, 10, 43, 40, 11]. Within this topic, catching fast-moving objects is a particularly relevant subdomain to our work. Prior work has explored both planning-based approaches [36, 25, 18, 30] and RL-based methods [44, 12, 20, 2] for object catching. However, these methods primarily focus on stable catching but without consideration of grasp configurations, which converges to using power grasps in most cases. Moreover, those methods do not consider the requirements of subsequent manipulation tasks. In contrast, our work addresses the more difficult challenge of recovery through catching, where the robot must not only catch falling objects but also achieve grasp configurations that enable seamless resumption of the primary manipulation task.

B. Representation Learning for Manipulation

The choice of perception representation (e.g., point cloud, image, contact information) to input into RL policies significantly impacts the manipulation performance. A majority of research directly uses point clouds as the representation for the 3D scene [21, 28, 13, 38, 1]. Recent work has also explored more sophisticated representations. For instance, Wu et al. [37] encode image observations into a learned latent space for model-based RL, while Driess et al. [7] utilize Neural Radiance Fields (NeRF) [24] to obtain latent scene embeddings for RL policy inputs. However, those methods do not consider contact-rich dexterous manipulation scenarios.

Neural Descriptor Fields (NDFs) and similar implicit geometric representations have also made progress in robot manipulation. Khargonkar et al. [17] propose an implicit grasp feature for cross-object and cross-robot generalization, while several other works [5, 14, 33] apply NDF-inspired representations to motion planning. However, the application of such implicit geometric representations to RL and dynamic dexterous manipulation remains unexplored. Our work addresses this gap by leveraging NDF-inspired contact representations to enable dynamic recovery.

III. Problem Statement

We focus on the task of catching a falling object for recovery. Specifically, the recovery problem inherently includes two objectives: (1)the robot must prevent dropping the object, and (2) the system should recover to a state from which the primary manipulation task can be resumed. The inclusion of the second objective distinguishes our work from prior literature on catching.

We formulate the dynamic recovery through catching as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and $\gamma \in [0, 1)$ is the discount factor. The objective is to optimize a policy π_θ to maximize the expected discounted return.

At each time step, the policy receives observations $\mathbf{o}_t := \{\mathbf{q}_t, \mathbf{x}_t, \mathbf{v}_t\}$, where $\mathbf{q}_t$ represents the robot's joint angles, $\mathbf{x}_t$ is the object pose in $\mathrm{SE}(3)$, and $\mathbf{v}_t$ denotes the object's twist (linear and angular velocities). We assume access to a low-level joint position controller; therefore, the robot action $\mathbf{a}_t$ is defined as the desired joint position at the next time step.

To make the problem more tractable, we assume knowledge of the object's full geometry, i.e., the full point cloud $\mathbf{P}$ of the object. The objective is to recover both the robot and the object to a desired configuration $\hat{\mathbf{q}}$ and $\hat{\mathbf{x}}$ respectively, which are manually defined and considered suitable for resuming the manipulation task.

Additionally, we aim to develop a method that generalizes across object geometries. During RL training, objects are randomly sampled from a predefined distribution. While during evaluation, objects are selected from both the in-distribution set and the out-of-distribution set. An ideal method should be able to catch the object in contact configurations similar to those from the training examples, even if the target object is out of distribution.

The method is evaluated based on (1) whether the robot successfully catches the object, (2) the difference between

the final state of catching and the desired state, (3) and the performance of the subsequent primary manipulation task. Compared to other dexterous manipulation setups, this task is more challenging because the robot needs to move fast to handle dynamic situations, but also precisely to reach the desired configurations.

IV. Methods

Our approach, Contact-Aware Dynamic Recovery (CADRE), leverages reinforcement learning to optimize recovery policies in dexterous manipulation tasks. Since the objective is to recover to desired grasps with appropriate contact patterns, the policy should naturally be aware of contacts. To enable contact awareness and generalization to unseen objects based on contacts, we incorporate contact features derived from NDFs into the observation space. The NDF model extracting contact features is pretrained and remains fixed during RL training. The contact features enable the RL policy to achieve consistent behaviors across objects with different geometries.

A. Preliminary: Neural Descriptor Fields

Neural Descriptor Fields (NDFs) [32] is a learned representation that captures geometric correspondence between a queried 3D coordinate and an object. Given an object point cloud $\mathbf{P}$, NDF extracts an n-dimensional geometric feature for any 3D coordinate $\mathbf{p}$:

$$f_{NDF}(\mathbf{p}|\mathbf{P}) : \mathbb{R}^3 \rightarrow \mathbb{R}^n. \tag{1}$$

During NDF training, an MLP-based network $\Phi(\mathbf{p}|\mathcal{E}(\mathbf{P}))$ is trained to predict occupancy or signed distance of the query point $\mathbf{p}$, conditioned on the object point cloud $\mathbf{P}$. $\mathcal{E}$ is the PointNet-based encoder, which extracts the latent features of the object point cloud. The NDF feature is defined as:

$$f_{NDF}(\mathbf{p}|\mathbf{P}) := \bigoplus_{i=1}^{L} \Phi_i(\mathbf{p}|\mathcal{E}(\mathbf{P})), \tag{2}$$

where $\oplus$ denotes concatenation of activations from each layer of Φ, and L is the number of hidden layers. This feature serves as an implicit occupancy representation and provides rich geometric information about the geometric relationship between the queried point and the object surface.

In our implementation, $\Phi(\mathbf{p}|\mathcal{E}(\mathbf{P}))$ is a double-headed model that predicts both occupancy and signed distance, similar to [14]. While occupancy prediction mainly captures features for points near contact regions, our method additionally requires features at non-contact points. In our method, the NDF model is pretrained and kept fixed during RL training.

B. Contact-Aware Dynamic Recovery

1) Contact-Aware Grasp Feature: In cases where the recovery policy must handle objects with various shapes, the robot needs to adjust its actions to catch (make contact) based on the object's geometry to ensure successful manipulation. Furthermore, the recovery should also result in favorable contact configurations for resuming the primary manipulation task. Therefore, incorporating contact information as a part of observation input is expected to improve training efficiency, convergence, and generalization performance.

Choosing an appropriate contact representation is the key question for designing the contact-aware recovery policy. In this work, we leverage NDFs to characterize contact features for dexterous manipulation. By querying points on the hand, we can interpret the corresponding NDF features as indicators of contact: contact points are expected to lie near the decision boundary of the occupancy function, where the NDF occupancy prediction output transitions between inside and outside predictions.

While NDF provides per-point contact features, it does not directly provide information characterizing the contact features of a grasp. To address this, we predefine K key points on the hand $\{\mathbf{p}_i^{k_i}\}_{i=1}^K$, where $\mathbf{p}_i^{k_i}$ denotes the position of the i-th key point in the k_i-th link frame. The grasp feature $g(\mathbf{q}, \mathbf{x}|\mathbf{P})$ is defined as the concatenation of the NDF features of all the contact points:

$$g(\mathbf{q}, \mathbf{x}|\mathbf{P}) := \bigoplus_{i=1}^{K} f_{NDF}(T(\mathbf{x}) f_{fk}(\mathbf{p}_i^{k_i}, \mathbf{q}, k_i)|\mathbf{P}), \tag{3}$$

where $\mathbf{q}$ is the robot joint angles, $\mathbf{x}$ is the object pose in $\mathrm{SE}(3)$. f_{fk} denotes the forward kinematics function returning key point locations in the world frame, and $T(\mathbf{x})$ is the transformation from the world frame to the object frame.

For key point selection, we sample points from each robot link rather than restricting them to the fingertips. Although fingertips are often primarily involved in contact, non-contact regions also play a critical role in characterizing the grasp feature. For example, one of the major distinctions between a power grasp and a precision grasp is whether the palm is in contact with the object. In practice, we assign one key point for each link for computational efficiency, although using more points could potentially lead to improved performance. (See details in Appendix C)

In summary, CADRE receives observations $\mathbf{o}_t$ and computes the contact-aware grasp feature $\mathbf{g} = g(\mathbf{q}, \mathbf{x}|\mathbf{P})$ at every time step. The input into the RL policy is the combination of observations and grasp features.

2) Reinforcement Learning for Dynamic Recovery: Due to the complexity of modeling the dynamic interactions between the hand and the object, we choose to use Proximal Policy Optimization(PPO) [31], a model-free RL method, to optimize the dynamic recovery policy.

Reward Function: The reward function is defined as follows:

$$r := r_{\dot{a}} + r_{torque} + r_{energy} + r_{drop} + r_{obj\_v} + r_{obj\_pose} + r_q + r_{contact}, \tag{4}$$

where we omit the weighting parameters and the time step subscript for simplicity. Specifically, $r_{\dot{\mathbf{a}}} := -||\mathbf{a}_t - \mathbf{a}_{t-1}||^2$ penalizes non-smooth actions, and $r_{toruqe} := -||\tau||^2$ penalizes torque. Although we assume only access to the joint position controller and joint torque τ is not part of the action space, τ is computed from the controller output. $r_{energy} := -||\tau^T \dot{q}||^2$

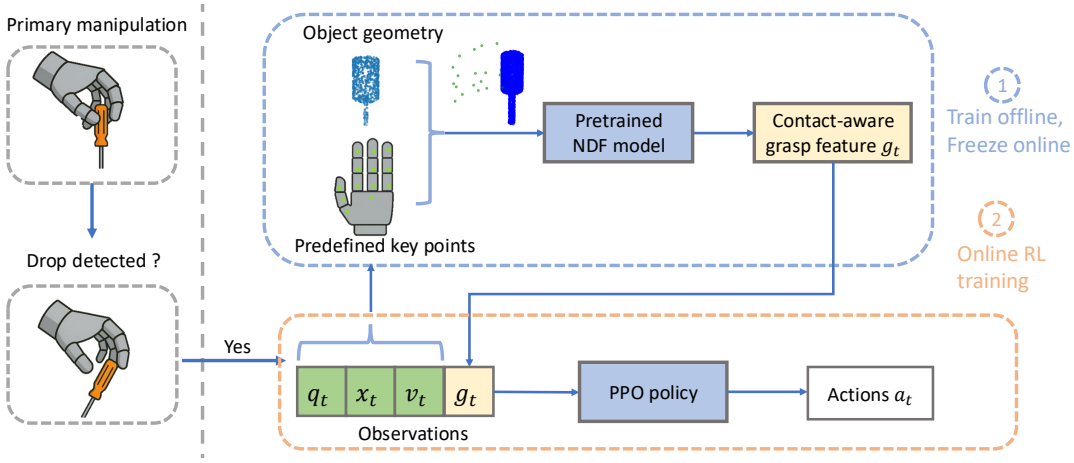


Fig. 2: Contact-Aware Dynamic Recovery (CADRE) aims to catch a manipulated object when it falls from the grasp and recover to states that support the resumption of the manipulation task. CADRE leverages a pretrained NDF model to extract implicit contact features from the grasp. The contact-aware grasp features are incorporated into the RL observations, enhancing the policy's awareness of contact and improving recovery performance.

regularizes the energy consumption for the robot. $r_{drop} := -\mathbb{1}_{drop}(\mathbf{x})$, where $\mathbb{1}_{drop}$ is an indicator function that returns 1 if the object's position exceeds a predefined workspace boundary. $r_{obj\_v} := \exp(-||\mathbf{v}||^2)+\exp(-||\boldsymbol{\omega}||^2)$ encourages low linear velocity $\mathbf{v}$ and angular velocity $\boldsymbol{\omega}$ of the object. We shape r_{obj_v} with an exponential function as the velocities can yield excessively large magnitudes during training (e.g., the object is falling fast). The exponential function helps bound the reward and leads to more stable training. $r_q := \exp(-||\mathbf{q}-\hat{\mathbf{q}}||^2)$ and $r_{obj\_pose} := \exp(-f_{pos}(\mathbf{x},\hat{\mathbf{x}}))+\exp(-f_{orn}(\mathbf{x},\hat{\mathbf{x}}))$ encourage the robot to recover to the desired robot configuration $\hat{\mathbf{q}}$ and object pose $\hat{\mathbf{x}}$. $\hat{\mathbf{q}}$ and $\hat{\mathbf{x}}$ are predefined and considered favorable for the primary manipulation task. Ideally, $\hat{\mathbf{q}}$ and $\hat{\mathbf{x}}$ would vary based on object geometries. In practice, we fix them across all objects because the optimal configurations remain similar within the range of variations considered during training. f_{pos} computes the difference between the current and desired object position, and f_{orn} computes the orientation difference. $r_{contact} := \mathbb{1}_{contact}(\mathbf{q},\mathbf{x})$ rewards the robot for achieving task-relevant contact with the object. The definition of $\mathbb{1}_{contact}$ depends on the task and is further explained in Sec. V.

V. Experiments

We evaluate our method on two tasks: screwdriver recovery and nut recovery. We aim to design our experiments to answer (1) whether incorporating implicit contact features can facilitate RL training and improve performance; and (2) whether CADRE generalizes to unseen objects sampled from a slightly different distribution.

A. Experiment Setup

We use IsaacSim to simulate the recovery task. Our robot consists of an Allegro Hand [29] mounted on a 7-DoF KUKA iiwa arm. We use PPO [31], implemented from RL games [23], to optimize the recovery policy. The RL implementation details are described in Appendix A.

Object generation: During RL training, we randomly sample 50 objects from a predefined shape distribution. Specifically, the objects are generated using the same geometric parameterization method, but with variations in their size parameters. (See Appendix B). The choice of objects are randomized across episodes during RL training to encourage generalization. For evaluation, we consider both in-distribution(ID) and out-of-distribution(OOD) objects. ID objects consist of 5 new unseen objects sampled from the same distribution used for training, while OOD objects are sampled from a different distribution as a more challenging test for generalization. Additionally, the NDF model f_{NDF} is trained with objects sampled from the same distribution for RL training. Thus, OOD objects for the RL policy will also be OOD for the NDF model.

Evaluation metric: (1) catch success rate: a catch is considered successful if the object remains within a predefined bounding box throughout the episode, (2) reward: cumulative reward over an episode, (3) number of desired contacts (4) number of undesired contacts, (5) position error: the Euclidean distance between the final and the desired object position, and (6) orientation error: defined as the angle between the final and desired z-axes of the object frame, since the objects in our experiments are rotationally symmetric about the z-axis. Metrics (3) and (4) are task-specific and detailed in Sec. V-A1 and V-A2. Metrics (2)-(6) are only calculated for successful trials.

As the most direct evaluation involves initializing the primary manipulation task with the final recovered states, we also introduce a screwdriver turning task to assess whether the post-recovery states are suitable for downstream manipulation tasks. See details in Sec. V-A1. Additional implementation details are described in Appendix A2.

1) Screwdriver Recovery: We set up the recovery task in a screwdriver turning scenario with a precision grasp. Screwdriver turning with a precision grasp has been widely

studied [26, 34, 41, 39, 19]. A precision grasp is preferred, as opposed to a power grasp, as the choice of robot-screwdriver contact significantly affects the turning performance, highlighting the importance of contact reasoning. Specifically, we follow the setup presented in [39](See Fig. 3a), where the index finger contacts the top of the screwdriver, while the thumb and middle finger form an antipodal grasp on the handle. To catch the falling screwdriver, the robot can attempt a power grasp. However, such grasps generally prevent resumption of the primary turning task. Thus, we define the desired recovered configuration $\hat{\mathbf{q}}$ and $\hat{\mathbf{x}}$ as the ones in which the robot is using a precision grasp. See the visualization in Fig. 3a. Based on the desired grasp configuration, we define the desired contact as the fingertips of the index, middle and thumb finger contacting the screwdriver. The undesired contact is defined as the contact between all other links of the hand with the screwdriver. The contact reward $r_{contact}$ returns the number of desired contact points.

We model the screwdriver as two connected cylinders: one for the handle and one for the shaft. OOD screwdrivers have approximately half the length of the ID screwdrivers.

To evaluate whether the recovered states are favorable for screwdriver turning, we follow the setup from [39], attempting to turn the screwdriver $60°$. We assume there exists a motion planning algorithm to mate the screwdriver with the screw and reorient it perfectly upright. In practice, we record the final state of recovery, keep the joint angles and the relative transformation between the hand and the screwdriver fixed, but we transform both the hand and the screwdriver so that the screwdriver is upright. We evaluate the turning performance via the success rate and the object orientation difference between the final turning state and the desired state, which indicates how far the robot turns the screwdriver towards the goal. A turning is successful if it does not drop the screwdriver, i.e., the Euler angles of the screwdriver remain below predefined thresholds.

2) Nut Recovery: We consider the recovery for nut mating with a bolt. During mating, the robot can easily drop the nut as the nut mating involves forceful contacts, especially if tactile sensors are not available.

As in the screwdriver task, successful recovery requires the robot to catch the nut with a specific grasp. In this case, the desired contact is defined as the fingertips contacting the side of the nut and avoiding the top and bottom. Contacts on the top or bottom will block the nut's central hole, making it impossible to mate with the bolt. Any other robot-nut contacts are defined as undesired contacts.

Additionally, there are cases where the robot appears to stabilize the nut by using the external support from the bolt. For instance, the robot might push the nut towards the bolt while the grasp itself is not stable without the support from the bolt. We penalize such behaviors in the contact reward: $r_{contact} := r_{robot\_nut} - r_{nut\_bolt}$, where r_{robot_nut} returns the number of desired contacts and r_{nut_bolt} is the indicator function for nut-bolt contact.

The nut is modeled as a hexagonal prism with a cylindrical hole, and the bolt is modeled as a cylinder. OOD nuts have about half the thickness of the ID nuts, making them harder to catch and requiring more precise finger control.

A recovery is considered successful if the nut is not dropped, and the nut does not contact the bolt, as the robot must stably grasp the nut without the bolt support.

3) Baselines: We consider three RL baselines. Both baseline methods use the same reward functions and RL training setup as CADRE, but have different inputs: (1) **PPO with object pose observations:** it directly takes the observation $\mathbf{o}_t$ defined in Sec. III as input. This method only has access to object pose but not object geometries. (2) **PPO with point cloud observations:** Reinforcement learning with point cloud input has been widely used in dexterous manipulation [13, 38, 1]. Similar to these methods, we add the object's full point cloud $\mathbf{P}_t$ into the aforementioned observation space: $\mathbf{o}'_t := \{\mathbf{q}_t, \mathbf{x}_t, \mathbf{v}_t, \mathbf{P}_t\}$. We use a full point cloud and do not include the robot point cloud to maintain the same information as our method. This method has access to the object's geometry but does not explicitly reason about contact features. (3) **Dexpoint:** [28]. Please see Sec. D for details about the baseline implementation.

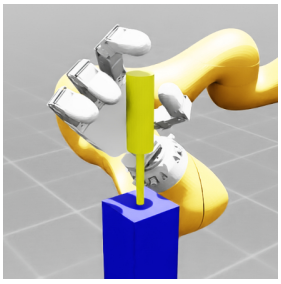
B. Experiment Results

The RL training curves are shown in Fig. 3. CADRE achieves higher rewards than baselines given the same number of environment interactions. In the screwdriver recovery task, although the baseline methods have eventually converged, they fail to match CADRE's performance.

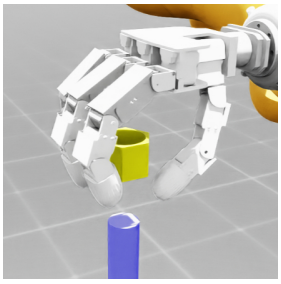
Each method is evaluated for 50 trials per object. The average performance is reported in Table Ia across all objects and all seeds. We also compute the standard deviation of the 50 trials on a given object, and then average the standard deviation from different objects and different seeds.

Screwdriver recovery: all methods achieve almost 100% success rates on ID objects. However, CADRE consistently performs better on all other metrics. This suggests that baselines often use power grasps rather than the desired precision grasps to catch the screwdriver. On OOD objects, CADRE still maintains similar performance, with only about 10% drop in the success rate. While the baseline exhibits a more significant performance drop. In terms of computation time, CADRE averages 5.14×10^{-3} seconds per step, while the pose baseline averages 8.04×10^{-4} seconds. and the point cloud baseline averages 2.09×10^{-3} seconds. Although contact feature computation introduces additional overhead, CADRE remains fast enough to support high-frequency control.

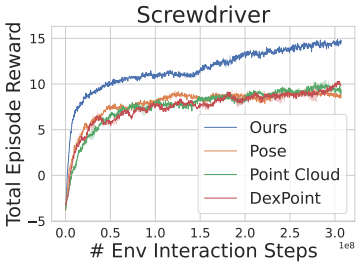
To further evaluate the quality of the recovered state, we set up the screwdriver turning (primary manipulation task) as described in Sec. V-A1 and use the turning algorithm proposed in [39]. Because the turning algorithm is computationally expensive, we only run it with the best-performing seed with the highest success rates for each method. We randomly sample 5 recovery trials per object for the turning evaluation. The goal is to turn the screwdriver 60 degrees. The turning results are shown in Table Ib. According to the results, CADRE has



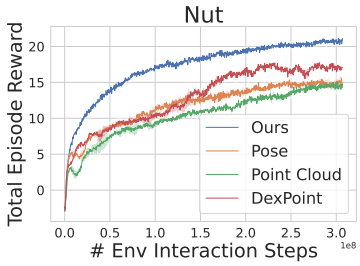
(a) Screwdriver



(b) Nut



(c) RL training curves for screwdriver recovery



(d) RL training curves for nut recovery

Fig. 3: Fig. (a) and (b) show the environment step. Three different seeds are used for RL training and the average results are shown in Fig. (c) and (d). The variance is relatively small as we use a very large batch size during training.

Obj	method	success↑	reward↑	#contact↑	#undesired contact↓	pos diff(cm)↓	orn diff↓
Screwdriver							
ID	CADRE	99.87%	**15.91** ±2.04	**2.91**	**0.05**	**1.97** ±1.72	**5.16°** ± 3.23°
	Pose	99.47%	10.50 ± 0.74	1.76	1.50	5.35 ± 1.82	8.90° ± 4.55°
	Point Cloud	99.87%	11.75 ± 0.89	1.92	1.48	3.67 ± 1.17	17.27° ± 6.60°
	Dexpoint	**100.00%**	11.41 ± 0.54	1.93	1.45	**1.09** ± 0.83	16.34° ± 5.84°
OOD	CADRE	**89.47%**	**15.17** ± 2.45	**2.79**	**0.09**	**3.04** ± 1.82	**8.25°** ± 6.14°
	Pose	78.40%	9.23 ± 1.29	1.43	1.32	5.03 ± 2.83	12.91° ± 12.92°
	Point Cloud	71.33%	9.98 ± 1.30	1.48	1.44	3.78 ± 1.64	20.79° ± 14.12°
	Dexpoint (high hand)	75.47%	10.48 ± 1.35	1.41	1.11	3.39 ± 1.90	23.22° ± 17.42°
	Dexpoint (low hand)	18.53%	8.80 ± 2.22	1.19	0.69	4.81 ± 1.84	30.61° ± 14.47°
Nut							
ID	CADRE	**98.27%**	**21.92** ± 2.69	**3.85**	**0.15**	**4.10** ± 2.39	**6.47°** ± 7.02°
	Pose	71.33%	16.16 ± 2.40	2.70	0.25	7.63 ± 1.93	21.58° ± 20.22°
	Point Cloud	78.93%	14.98 ± 2.53	1.59	0.41	6.00 ± 2.67	35.19° ± 21.35°
	Dexpoint	72.53%	19.03 ± 3.05	3.01	0.36	5.81 ± 0.79	13.97° ± 18.62°
OOD	CADRE	**92.93%**	17.99 ± 4.39	1.98	0.17	6.50 ± 3.94	18.23° ± 19.15°
	Pose	50.13%	16.44 ± 2.41	2.89	**0.01**	7.33 ± 1.66	19.74° ± 18.45°
	Point Cloud	48.67%	14.66 ± 3.21	1.55	0.16	6.22 ± 2.68	35.03° ± 28.81°
	Dexpoint	42.13%	**20.27** ± 2.77	**3.25**	0.08	**5.70** ± 1.00	**10.55°** ± 13.64°

(a) Evaluation results on unseen screwdriver recovery and nut recovery. All metrics except success rate are computed for successful trials only. We report two Dexpoint results in the OOD screwdriver recovery task. Please see Appendix. D for detailed reasoning.

	ID	
	success ↑	dist2goal ↓
CADRE	**80%**	**4.73°** ± 3.51°
Pose	12%	47.21° ± 10.22°
Point Cloud	60%	38.05° ± 12.92°
Dexpoint	64%	38.05° ± 12.92°
	OOD	
CADRE	**76%**	**4.11°** ± 3.26°
Pose	32%	50.81 ∘ ±18.15°
Point Cloud	28%	47.49° ± 7.13°
Dexpoint	65%	40.67° ± 20.18°

(b) Evaluation results for screwdriver turning initialized from recovered states. Dist2goal represents the screwdriver orientation difference between the final state of turning and the desired state of turning 60°.

demonstrated significantly better performance in both ID and OOD scenarios in terms of both metrics. The performance drop from ID to OOD of CADRE is much smaller than that of the baselines. The baseline with pose observation has a better performance on OOD objects than ID objects. We believe this is because of its strategy of attempting to place the index finger on top of the screwdriver. OOD objects are much shorter and such attempts are easier to achieve, and the index-top contact is helpful to maintain screwdriver stability.

We also performed hardware experiments to evaluate our method. Please refer to Appendix E2 for details.

Nut recovery: we have also observed similar results in the nut-catching task (See results in Table Ia), with CADRE outperforming baselines in ID and OOD scenarios. We do not perform the nut-mating evaluation, mainly due to the complexity of nut-mating with a dexterous hand, which still remains an open research problem. However, based on our screwdriver recovery experiments, we hypothesize a correlation between our evaluation metrics and the performance of the primary manipulation task.

VI. Discussion and Conclusion

In this work, we propose CADRE, an RL framework that utilizes an implicit contact representation derived from NDFs. We focus on the problem of recovering from catastrophic failure in dexterous manipulation–specifically, recovering from dropping objects and returning to states favorable for resuming the primary manipulation task.

We evaluate CADRE on screwdriver and nut recovery tasks. Our experiments demonstrate that simply providing point clouds as object geometry observations for RL is insufficient for learning effective dynamic recovery in contact-rich scenarios. In contrast, CADRE leverages implicit contact features to improve training efficiency, grasp quality, and generalization to unseen object geometries.

Beyond dynamic recovery, our results suggest the broader potential for RL of using implicit contact representations in contact-rich manipulation. Further extending this approach to more general settings presents a promising research direction for our future research.

References

[1] Chen Bao, Helin Xu, Yuzhe Qin, and Xiaolong Wang. Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21190–21200, 2023.

[2] Henry J Charlesworth and Giovanni Montana. Solving challenging dexterous manipulation tasks with trajectory optimisation and reinforcement learning. In *International Conference on Machine Learning*, pages 1496–1506. PMLR, 2021.

[3] Claire Chen, Preston Culbertson, Marion Lepert, Mac Schwager, and Jeannette Bohg. Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8262–8268. IEEE, 2021.

[4] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023.

[5] Shuo Cheng, Caelan Reed Garrett, Ajay Mandlekar, and Danfei Xu. Nod-tamp: Multi-step manipulation planning with neural object descriptors. In *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*, 2023.

[6] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T Mason. Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6520–6526. IEEE, 2021.

[7] Danny Driess, Ingmar Schubert, Pete Florence, Yunzhu Li, and Marc Toussaint. Reinforcement learning with neural radiance fields. *Advances in Neural Information Processing Systems*, 35:16931–16945, 2022.

[8] Thomas Feix, Javier Romero, Heinz-Bodo Schmiedmayer, Aaron M Dollar, and Danica Kragic. The grasp taxonomy of human grasp types. *IEEE Transactions on human-machine systems*, 46(1):66–77, 2015.

[9] Patrick Grady, Chengcheng Tang, Christopher D Twigg, Minh Vo, Samarth Brahmbhatt, and Charles C Kemp. Contactopt: Optimizing contact to improve grasps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1471–1481, 2021.

[10] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robot Learning*, pages 24–33. PMLR, 2022.

[11] Yifan Hou, Zhenzhong Jia, Aaron M Johnson, and Matthew T Mason. Robust planar dynamic pivoting by regulating inertial and grip forces. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pages 464–479. Springer, 2020.

[12] Binghao Huang, Yuanpei Chen, Tianyu Wang, Yuzhe Qin, Yaodong Yang, Nikolay Atanasov, and Xiaolong Wang. Dynamic handover: Throw and catch with bimanual hands. *arXiv preprint arXiv:2309.05655*, 2023.

[13] Wenlong Huang, Igor Mordatch, Pieter Abbeel, and Deepak Pathak. Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062*, 2021.

[14] Zixuan Huang, Yinong He, Yating Lin, and Dmitry Berenson. Implicit contact diffuser: Sequential contact reasoning with latent point cloud diffusion. *arXiv preprint arXiv:2410.16571*, 2024.

[15] Tatsuya Ishihara, Akio Namiki, Masatoshi Ishikawa, and Makoto Shimojo. Dynamic pen spinning using a high-speed multifingered hand with high-speed tactile sensor. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 258–263. IEEE, 2006.

[16] Wanxin Jin. Complementarity-free multi-contact modeling and optimization for dexterous manipulation. *arXiv preprint arXiv:2408.07855*, 2024.

[17] Ninad Khargonkar, Neil Song, Zesheng Xu, Balakrishnan Prabhakaran, and Yu Xiang. Neuralgrasps: Learning implicit representations for grasps of multiple robotic hands. In *Conference on Robot Learning*, pages 516–526. PMLR, 2023.

[18] Seungsu Kim, Ashwini Shukla, and Aude Billard. Catching objects in flight. *IEEE Transactions on Robotics*, 30 (5):1049–1065, 2014.

[19] Abhinav Kumar, Thomas Power, Fan Yang, Sergio Aguilera Marinovic, Soshi Iba, Rana Soltani Zarrin, and Dmitry Berenson. Diffusion-informed probabilistic contact search for multi-finger manipulation. *arXiv preprint arXiv:2410.00841*, 2024.

[20] Fengbo Lan, Shengjie Wang, Yunzhe Zhang, Haotian Xu, Oluwatosin Oseni, Ziye Zhang, Yang Gao, and Tao Zhang. Dexcatch: Learning to catch arbitrary objects with dexterous hands. *arXiv preprint arXiv:2310.08809*, 2023.

[21] Zhan Ling, Yunchao Yao, Xuanlin Li, and Hao Su. On the efficacy of 3d point cloud reinforcement learning. *arXiv preprint arXiv:2306.06799*, 2023.

[22] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[23] Denys Makoviichuk and Viktor Makoviychuk. rl-games: A high-performance framework for reinforcement learning. https://github.com/Denys88/rl_games, May 2021.

[24] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[25] Akio Namiki, Yoshiro Imai, Masatoshi Ishikawa, and Makoto Kaneko. Development of a high-speed multifingered hand system and its application to catching.

In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2666–2671. IEEE, 2003.

[26] Patrick Naughton, Jinda Cui, Karankumar Patel, and Soshi Iba. Respilot: Teleoperated finger gaiting via gaussian process residual learning. *arXiv preprint arXiv:2409.09140*, 2024.

[27] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.

[28] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *Conference on Robot Learning*, pages 594–605. PMLR, 2023.

[29] Wonik Robotics. Allegro hand. https://www.allegrohand.com/, 2024.

[30] Seyed Sina Mirrazavi Salehian, Mahdi Khoramshahi, and Aude Billard. A dynamical system approach for softly catching a flying object: Theory and experiment. *IEEE Transactions on Robotics*, 32(2):462–471, 2016.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[32] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE, 2022.

[33] Anthony Simeonov, Yilun Du, Yen-Chen Lin, Alberto Rodriguez Garcia, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Pulkit Agrawal. Se (3)-equivariant relational rearrangement with neural descriptor fields. In *Conference on Robot Learning*, pages 835–846. PMLR, 2023.

[34] Ling Tang, Yan-Bin Jia, and Yuechuan Xue. Robotic manipulation of hand tools: The case of screwdriving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13883–13890. IEEE, 2024.

[35] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[36] Gaotian Wang, Kejia Ren, Andrew S Morgan, and Kaiyu Hang. Caging in time: A framework for robust object manipulation under uncertainties and limited robot perception. *arXiv preprint arXiv:2410.16481*, 2024.

[37] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

[38] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance. In *Conference on Robot Learning*, pages 618–629. PMLR, 2023.

[39] Fan Yang, Thomas Power, Sergio Aguilera Marinovic, Soshi Iba, Rana Soltani Zarrin, and Dmitry Berenson. Multi-finger manipulation via trajectory optimization with differentiable rolling and geometric constraints. *arXiv preprint arXiv:2408.13229*, 2024.

[40] William Yang and Michael Posa. Dynamic on-palm manipulation via controlled sliding. *arXiv preprint arXiv:2405.08731*, 2024.

[41] Zhao-Heng Yin, Changhao Wang, Luis Pineda, Francois Hogan, Krishna Bodduluri, Akash Sharma, Patrick Lancaster, Ishita Prasad, Mrinal Kalakrishnan, Jitendra Malik, et al. Dexteritygen: Foundation controller for unprecedented dexterity. *arXiv preprint arXiv:2502.04307*, 2025.

[42] Rana Soltani Zarrin, Rianna Jitosho, and Katsu Yamane. Hybrid learning-and model-based planning and control of in-hand manipulation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8720–8726. IEEE, 2023.

[43] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[44] Yuanhang Zhang, Tianhai Liang, Zhenyang Chen, Yanjie Ze, and Huazhe Xu. Catch it! learning to catch in flight with mobile dexterous hands. *arXiv preprint arXiv:2409.10319*, 2024.

APPENDIX

A. RL Training Details

1) Domain Randomization: Since accurately perceiving fast-moving objects can be challenging in real-world scenarios, catching dynamic objects for recovery needs to handle significant uncertainties. To address this, we apply domain randomization [35] to improve robustness. Specifically, we introduce two types of uncertainty: external disturbances and perception noise. At each time step, a random external wrench $\mathbf{w}_{ext}$ is applied at the center of the screwdriver, and random perception noise $\boldsymbol{\epsilon}$ sampled from a uniform distribution is added to the object's pose and velocities. For the external wrench $\mathbf{w}_{ext} := \{\mathbf{f}_{ext}, \boldsymbol{\tau}_{ext}\}$, we randomly sample:

- the force $\mathbf{f}_{ext}$ from a uniform distribution of $\mathcal{U}(-1.5\text{N}, 1.5\text{N})$.
- the wrench $\boldsymbol{\tau}_{ext}$ from a uniform distribution of $\mathcal{U}(-0.01\text{Nm}, 0.01\text{Nm})$.

Perception noise $\boldsymbol{\epsilon}$ consists of position $\boldsymbol{\epsilon}_{pos}$ and orientation noise $\boldsymbol{\epsilon}_{orn}$. We sample the position noise from $\mathcal{U}(-1\text{cm}, 1\text{cm})$ for each dimension. For orientation error, as orientations are represented as quaternions, we randomly sample each noise dimension from $\mathcal{U}(-0.01, 0.01)$, add it to the quaternion and renormalize to ensure a valid quaternion.

2) Environment Initialization: **Screwdriver Recovery:** During training, the robot is initialized with a predefined configuration $\hat{\mathbf{q}}$, which is assumed to be favorable for the turning task. This initialization is reasonable given that an ideal turning policy normally only controls the robot toward such favorable configurations. The screwdriver's tip position is randomly sampled within a small range, since in practice the screwdriver is usually mated with the screw, and its orientation is sampled from a predefined distribution. If interpenetration between the robot and the object occurs at the initial state, we keep the object fixed and adjust the robot configuration accordingly. During evaluation, we change the object orientation initialization to make the task more challenging: we only initialize the object in directions falling away from the hand (rather than falling towards the hand). This is because in cases where the object falls towards the hand, the robot is able to catch it without any movement. We focus on the more interesting setup where the robot requires active movement for recovery.

When evaluating on OOD objects, we lower the initial hand height by 3 cm, except for the Dexpoint method, where we have experiments for two different heights (see Sec. D for details). This is because the OOD screwdrivers are shorter and the distance the robot needs to travel to make contact is longer. If initialized with the original height, it is very challenging for the robot to move fast enough to catch the screwdriver before the screwdriver drops. Moreover, the original hand height will create a large gap between the robot and the OOD screwdriver, which is an unlikely configuration during screwdriver turning, and one from which our recovery policy would not typically be triggered.

Fig. 4: Key Points visualization used for extracting contact-aware grasp feature.

Nut Recovery: The initialization for nut recovery is similar to the screwdriver recovery task. The only difference is that we keep the training and evaluation distribution the same as there exist no trivial initial setups that we shouldn't include in the evaluation.

B. Parameters for Generating Objects

The parameters are shown in Table II. For OOD object design, we specifically choose shorter screwdrivers and thinner nuts as those choices present significant challenges, where the desired contact regions are smaller and generally require more precise finger control.

C. Key Point Definition

To illustrate the key points used for extracting the contact-aware grasp feature, we manually annotate the key points, as shown in Fig. 4. Specifically, we directly select the root point of each link as the corresponding keypoint. It is worth noting that the keypoints are not required to lie within the actual contact regions. Since NDF provides an implicit contact representation, indicating how far each keypoint is from the object's surface, regardless of direct contact.

D. DexPoint Experiments

Similar to our method, Dexpoint [28] aims to leverage contact information and improve generalization across object geometries. The key components of Dexpoint include: (1) augmenting the observation with an additional imagined point cloud of the robot, and (2) incorporating contact-based reward functions during RL training.

Dexpoint assumes access to the robot's geometry, so we render a point cloud for each fingertip and the palm, and concatenate it with the object's point cloud. Unlike the original Dexpoint paper, we do not use the observed point cloud, i.e., the point cloud from a depth camera, as other methods in our experiments do not have access to such observations. We also retain our original reward function, which already includes contact-based reward terms similar to those in DexPoint.

In OOD screwdriver experiments in Table Ia, we present two sets of Dexpoint experiments using different initial hand heights: *low hand* corresponds to the same hand height used by other methods during OOD evaluation, while *high hand* refers to the original hand height used during RL training. Although lowering the hand height benefits other methods, it significantly reduces the success rate of Dexpoint. This

	Screwdriver				Nut		
	handle r	handle l	shaft r	shaft l	inner r	outer r	thickness
ID	$\mathcal{U}(1, 3)$	$\mathcal{U}(7, 14)$	$\mathcal{U}(0.4, 1.4)$	$\mathcal{U}(8, 10)$	$\mathcal{U}(1, 2.5)$	$\mathcal{U}(3, 4)$	$\mathcal{U}(3, 5)$
OOD	$\mathcal{U}(1.5, 4)$	$\mathcal{U}(4, 10)$	$\mathcal{U}(0.4, 0.8)$	$\mathcal{U}(4, 6)$	$\mathcal{U}(1, 2.5)$	$\mathcal{U}(3, 4)$	$\mathcal{U}(1, 3)$

TABLE II: Distribution used for generating object geometries. r: radius, l: length. $\mathcal{U}$ means uniform distribution. All units are in centimeters. OOD objects are generally shorter(screwdriver) or thinner(nut) than ID ones.

is mainly because Dexpoint learns a catching strategy that involves quickly moving the hand forward. When the hand is lowered, the palm is more likely to collide with the blue pillar holding the screwdriver, leading to failure. Therefore, we also report the results using the original (high) hand height, which better accommodates Dexpoint's learned strategy. For evaluating the subsequent screwdriver turning, we choose high-hand results.

Though CADRE generally outperforms Dexpoint in most metrics, in terms of the primary metric of success rate, Dexpoint performs comparably to other baselines. It also has a better performance in the downstream screwdriver turning. In the nut recovery task, it shows better reward and other metrics on contact quality than other baselines. Compared to the tasks evaluated in the Dexpoint paper, our tasks are more dynamic and demand greater precision in contact configurations, which may explain why Dexpoint does not consistently outperform other baselines in our setting.

E. Hardware Experiments

The main objective of the hardware experiment is to evaluate whether we can successfully transfer the highly dynamic recovery behavior learned in simulation to a real-world setup. We only focus on the screwdriver recovery task in our hardware experiments. The hardware experiment does not focus on generalization abilities across object geometries. Thus, we use an unseen screwdriver that falls within the training distribution for our hardware experiment.

1) Experiment Setup: We use the Vicon motion capture (mocap) system to estimate the state of the screwdriver. Since the hardware system has different parameters from our previous simulation environments, such as stiffness and damping of the PD controller, we retrain all methods to match the parameters of the hardware.

To enable deployment in the real world, we distill an open-loop policy from the closed-loop RL policy. Closed-loop execution of a highly dynamic manipulation policy is particularly challenging because of (1) Latency: Our system exhibits approximately 50 ms of latency between sending control commands and receiving the corresponding observation update. Such millisecond latency is usually negligible in many quasi-static tasks, but will lead to instability in our highly dynamic tasks; and (2) Occlusion: When the robot attempts to catch the screwdriver, its fingers will often block the mocap markers, causing loss of object state information.

Additionally, although mocap provides a high-frequency ttracking of the object, alternative perception methods, such as AprilTags [27], usually suffer from slower update rates and motion blur resulting from fast object movements. An open-loop policy is likely to mitigate these issues and increase compatibility with a broader range of perception systems.

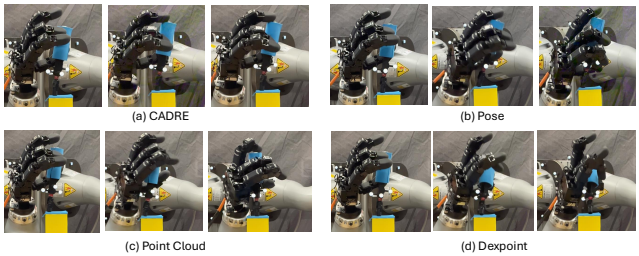


Fig. 5: Example sequences from the hardware experiment. Each frame is approximately 0.1 seconds apart.

	succ rate ↑	#desired contact (succ only)↑	#undesired contact (succ only)↓
CADRE	**100%**	**2.93**	**0.27**
Pose	93.33%	1.36	1.71
Point Cloud	86.67%	2.31	1.85
Dexpoint	86.67%	1.85	2.15

TABLE III: Distilled open-loop policies are evaluated for 15 hardware trials per method. The number of desired contacts and undesired contacts is computed only for successful trials.

To distill the open-loop policy, we configure the simulation environment to match the geometry of the screwdriver used in the hardware experiments. We collect 10,000 successful rollouts from the closed-loop policy, and train a supervised open-loop policy that takes the initial observation as input and outputs a sequence of actions for the entire trajectory. Though open-loop distillation will introduce additional action errors and potentially non-smooth trajectories, we demonstrate in our experiments that CADRE is still able to achieve a reasonable recovery behavior.

One of the main challenges in the hardware experiment is resetting the system to a state where the screwdriver is about to fall. As the error detection is not the focus of this work, we manually create such a falling scenario: we first manually position the screwdriver so that the robot can grasp it. To initiate the drop, we command the robot to open its hand for a short, fixed duration. The recovery policy is then triggered. To eliminate bias from manually setting the screwdriver's initial pose, we anonymize the methods during each trial: the method under evaluation is randomly selected and not revealed to the experimenter until the end of this trial.

2) Experiment Results: The quantitative results are shown in Table III, and the qualitative results are shown in Fig. 5.

Our method demonstrates a better performance in the hard-

ware experiments. However, there is still room for improvement, including reducing action jerk, achieving better contact configurations, and minimizing applied robot forces for safety.

F. Interpretation of the Metrics in Our Experiment

We present six evaluation metrics in our experiments. Here we clarify their significance.

Success rate is the primary metric–if the object is not caught, evaluating the quality of final catching states is not relevant. Reward and the number of desired contacts are most indicative of the recovery quality, as we have seen in our screwdriver experiments that they are closely related to the downstream task performance. Position difference and orientation difference are secondary, since the robot can use motion planning to reposition and reorient the object after catching the object.

G. Limitations

While CADRE demonstrates the ability to generalize across different geometries, the underlying NDF structure only focuses on geometry information for generalization. However, variations in geometries will also lead to different dynamics. For example, a grasp on an unseen object that shares similar grasp features with a force closure grasp on a known object might not result in force closure.

Additionally, dynamic manipulation with precise control introduces significant sim-to-real challenges. Specifically, Factors such as system latency, gravity compensation errors, and policy computation time, which are often negligible in quasi-static scenarios, can have a substantial impact on performance in highly dynamic recovery tasks. While CADRE can successfully catch a falling object in our hardware experiment, there remains room for improvement in achieving more precise desired contact configurations, e.g., by making the policy aware of the system latency and dynamics discrepancies. We will focus on addressing the limitations in our future work.