UNIVERSITY OF CALIFORNIA SAN DIEGO

# Learning Robotic Manipulation in 3D

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Litian Liang

2023

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

## LIST OF TABLES

ACKNOWLEDGEMENTS

I feel profoundly grateful for the guidance and inspiration provided by my advisor, Professor Hao Su. His audacious and lucid foresight into the future of embodied AI, coupled with his dedication to advancing current technological boundaries, has been the greatest source of empowerment to me. I would also like to thank Professor Henrik Christensen and Professor Michael Yip for their help with my thesis defense. Without their help, none of this would be possible.

Chapter 1, in full, is a reprint of the material as it appears in Reparameterized Policy Learning for Multimodal Trajectory Optimization. Huang, Zhiao; Liang, Litian; Ling, Zhan; Li, Xuanlin; Gan, Chuang; Su, Hao. International Conference on Machine Learning (ICML 2023). The dissertation author was the primary researcher and author of this paper.

Chapter 2, in full, has been submitted for publication of the material as it may appear in Robo360: A 3D Omnispective Multi-Material Robotic Manipulation Dataset. Liang, Litian; Bian, Liuyu; Xiao, Caiwei; Zhang, Jialin; Chen, Linghao; Liu, Isabella; Xiang, Fanbo; Huang, Zhiao; Su, Hao. IEEE / CVF Computer Vision and Patter Recognition (CVPR 2024). The dissertation author was the primary researcher and author of this paper.

## VITA

| | |
|---|---|
| 2022 | Bachelor of Science in Computer Science, University of California  Irvine |
| 2023 | Master of Science in Computer Science, University of California  San Diego |

ABSTRACT OF THE THESIS

Learning Robotic Manipulation in 3D

by

Litian Liang

Master of Science in Computer Science

University of California San Diego, 2023

Professor Hao Su, Chair

Building robots that can automate tedious, hazardous, and repetitive jobs has long been the driving force behind the advancements in machine learning, computer vision, and robotics community. Recent breakthroughs in deep learning, a data centric approach to building computer software, achieved tremendous success in visual recognition, natural language understanding, and video game playing. However, these approaches usually require large and diverse datasets to generalize to unseen situations.

Collecting diverse data is the core challenge in all existing learning-based methods for robotic manipulation. Many approaches learn robotic manipulation policy directly on real-world robot-object interaction data. However, collecting real data is order of magnitudes more costly than visual recognition and natural language tasks. In some cases, collecting real robot data of some tasks is even impossible with existing infrastructure. Other approaches first learn policies in simulation, then deploy in the real world. However, these methods encounter another set of challenges including hard to transfer due to physics gap between simulation and real world. When using RL to learn policies in simulation, challenges exist also at the algorithmic level. The engineered dense reward is hard to specify for the policy to autonomously collect data closer to the globally optimal solution, and sparse reward is hard for algorithms to optimize.

In this thesis, we will introduce two projects that lay the foundation of two promising directions of building real-world Embodied AI: 1. Large scale sparse reward policy learning in simulation, 2. Continuously improving simulation with real data. These projects serve as the foundation for building future RL algorithms and learning based simulations.

# INTRODUCTION

To automate tedious, hazardous, and repetitive jobs require a sophisticated system of multiple components working seamlessly together, including a visual perception module needs to understand the geometry, physical property, and spatial relationship of objects, and a robot planning and control system needs to produce a sequence of low-level control signal to influence the world.

One approach to learn such a robot control policy can be very similar to the established paradigm in visual recognition and natural language tasks. Given each observation, human labelers label the ground truth actions for the robot by teleoperating the robot in real world to complete the task. However, this method has many significant drawbacks. First, many contacts rich tasks that require haptic feedback for not only the robot but also the teleoperator to identify the state of objects that are being manipulated. Although there are many teleoperation systems that exist already, none provide touch sensing and force feedback to the teleoperator, which limits the types of tasks solvable with this method. Second, collecting real robot data requires extensive training of the teleoperator to be familiar with the system and reduce the number of accidents. When they are not used properly, these expensive and heavy-duty robot arms can easily damage themselves or other objects.

Another approach to learn robot control policy considers learning the policy in simulation, then deploy in the real world. In this case, since there is no cost in executing the policy to collect data, the data collection and policy update loop is often formulated under the paradigm of reinforcement learning (RL). Under this formulation, the policy is initialized to produce random actions and gradually explores the environment and finds solution through optimizing reward. However, this method is also problematic in multiple ways. 1. Due to the lack of ability of existing algorithms to explore the environment, the learning processes rely on an engineered dense reward to guide the optimization. 2. These dense rewards are individually designed for each task, which significantly limits scalability. 3. These dense rewards also have many local optima that can easily make policy optimization stuck. 4. The difficulty of transferring learned policies to the real world is much harder due to the problem nature of robotic manipulation, which often involves object visual, geometrical, and material variations.

Can we achieve large scale optimization of policies without engineering dense reward? Can we close the gap between simulation and the real world through a data-driven approach? To better answer these questions, we will introduce two papers. 1. An RL algorithm that still learns to solve the task when only sparse reward is used during optimization. 2. A real-world dataset and data collection system of omnispective-view robot-object interactions.

# Reparameterized Policy Learning for Multimodal Trajectory Optimization

Zhiao Huang [1] Litian Liang [1] Zhan Ling [1] Xuanlin Li [1] Chuang Gan [2 3] Hao Su [1]

## Abstract

We investigate the challenge of parametrizing policies for reinforcement learning (RL) in high-dimensional continuous action spaces. Our objective is to develop a multimodal policy that overcomes limitations inherent in the commonly-used Gaussian parameterization. To achieve this, we propose a principled framework that models the continuous RL policy as a generative model of optimal trajectories. By conditioning the policy on a latent variable, we derive a novel variational bound as the optimization objective, which promotes exploration of the environment. We then present a practical model-based RL method, called Reparameterized Policy Gradient (RPG), which leverages the multimodal policy parameterization and learned world model to achieve strong exploration capabilities and high data efficiency. Empirical results demonstrate that our method can help agents evade local optima in tasks with dense rewards and solve challenging sparse-reward environments by incorporating an object-centric intrinsic reward. Our method consistently outperforms previous approaches across a range of tasks. Code and supplementary materials are available on the project page `https://haosulab.github.io/RPG/`

## 1. Introduction

Reinforcement learning (RL) with *high-dimensional continuous action space* is notoriously hard despite its fundamental importance for many application problems such as robotic manipulation (OpenAI et al., 2019; Mu et al., 2021). In practice, popular frameworks (Silver et al., 2014; Haarnoja et al., 2018; Schulman et al., 2017) of deep RL formulate the continuous policy as a neural network that outputs a single-modal density function over the action space

[1]UC San Diego [2]MIT-IBM Watson AI Lab [3]UMass Amherst. Correspondence to: Zhiao Huang <z2huang@ucsd.edu>.

*Proceedings of the 40th International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

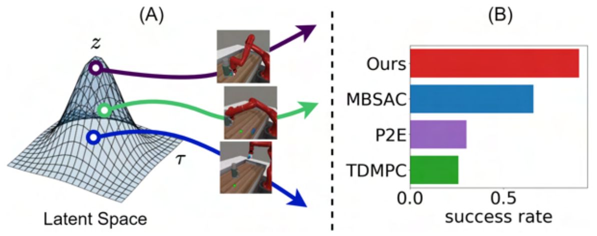


*Figure 1.* (A) Our method reparameterizes latent variables into multimodal policy to facilitate exploitation and exploration in continuous policy learning; (B) Average performance on 6 hard exploration tasks. Our method outperforms previous methods.

(e.g., a Gaussian distribution over actions). This formulation, however, breaks the promise of RL being a global optimizer of the return function because the single-modality policy parameterization introduces local minima that are hard to escape using gradients w.r.t. distribution parameters. Besides, a single-modality policy will significantly weaken the exploration ability of RL algorithms because the sampled actions are usually concentrated around the modality.

Although there are other candidates beyond the Gaussian distribution for policy parameterization, they often have limitations when used for continuous policy modeling. For example, Gaussian mixture models can only accommodate a limited number of modes; normalizing flow methods (Rezende & Mohamed, 2015) can compute density values, but they may not be as numerically robust due to their dependency on the determinant of the network Jacobian; furthermore, normalizing flows must apply continuous transformations onto a continuously connected distribution, making it difficult to model disconnected modes (Rasul et al., 2021). Option-critic (Bacon et al., 2017) represents policies with options and temporal structure, but it often requires specially designed option spaces for efficient learning, which motivates research on hierarchical imitation learning that uses demonstrations to avoid exploration problems (Peng et al., 2022; Fang et al., 2019). Skill discovery methods learn a population of skills without demonstrations or rewards by optimizing for diversity (Eysenbach et al., 2018). However, the separation of optimization and skill learning can be non-efficient as it expends effort on learning task-irrelevant skills and may ignore more important ones that would benefit a specific task.

This paper presents a principled framework for learning the continuous RL policy as a multimodal density function through multimodal action parameterization. We adopt a sequence modeling perspective (Chen et al., 2021) and view the policy as a density function over the entire trajectory space (instead of the action space)(Ziebart, 2010; Levine, 2018). This allows us to sample a population of trajectories that cover multiple modalities, enabling concurrent exploration of distant regions in the solution space. Additionally, we use a generative model to parameterize the multimodal policies, drawing inspiration from their success in modeling highly complex distributions such as natural images(Goodfellow et al., 2016; Zhu et al., 2017; Rombach et al., 2022; Ramesh et al., 2021). We condition the policy on a latent variable $z$ and use a powerful function approximator to "reparameterize" the random distribution $z$ into the multimodal trajectory distribution (Kingma & Welling, 2013), from which we can sample trajectories $\tau$. This policy parameterization leads us to adopt the variational method (Kingma & Welling, 2013; Haarnoja et al., 2018; Moon, 1996) to derive a novel framework for modeling the posterior of the optimal trajectory using variational inference, which enables us to model multimodal trajectories and maximize the reward with a single objective.

This framework allows us to build Reparameterized Policy Gradient (RPG), a model-based RL method for multimodal trajectory optimization. The framework has two notable features: First, RPG combines the multimodal policy parameterization with a learned world model, enjoying the sample efficiency of the learned model and gradient-based optimization while providing the additional ability to jump out of the local optima; Second, we equip RPG with a novel density estimator to help the multimodal policy explore in the environments by maximizing the state entropy (Hazan et al., 2019). We verify the effectiveness of our methods on several robot manipulation tasks. These environments only provide sparse rewards when the agent successfully fully finishes the task, which is challenging for single-modal policies even when they are guided by intrinsic motivations. In comparison, our method is able to explore different modalities, improve the exploration efficiency, and outperform single-modal policies, as shown in Fig. 1. Notably, our method is more robust than single-modal policies and consistently outperforms previous approaches across different tasks.

Our contributions are multifold: 1. We propose a variational policy learning framework that models the posterior of multimodal optimal trajectories for reward optimization. 2. We demonstrate that multimodal parameterization can help the policy escape local optima and accelerate exploration in continuous policy optimization. 3. When combined with a learned world model and a delicate density estimator, our method, RPG, is able to solve these challenging sparse-reward tasks more efficiently and reliably.

## 2. Related Work

**Policy as Sequential Generative Model.** Maximum entropy reinforcement learning (Todorov, 2006; 2008; Toussaint, 2009; Ziebart, 2010; Kappen et al., 2012) can be viewed as variational inference in probabilistic graphical models (Levine, 2018) with optimality as an observed variable and sampled trajectories as latent variables. When the demonstration or a fixed dataset is provided in the *offline* RL setting (Chen et al., 2021; Reed et al., 2022), policy learning is simplified as a sequence modeling task (Chen et al., 2021; Zheng et al.; Reed et al., 2022). They use autoregressive models to learn the distribution of the whole trajectory, including actions, states, and rewards, and use the action prediction as policy. In our work, we learn a sequential generative model of policy for *online* RL via the variational method.

**Variational Skill Discovery** Under additional assumptions of rewards, our method degenerates to skill discovery methods. However, previous skill discovery methods focus on unsupervised reinforcement learning (Eysenbach et al., 2018; Achiam et al., 2018; Campos et al., 2020) or diverse skill learning (Kumar et al., 2020; Osa et al., 2022). These methods build latent variable policy and encourage the policy to reach states that are consistent with the sampled latent variables through a mutual information term as a reward. These methods do not consider reward maximization or exploration when learning the skills, making them differ from our method vastly. For example, Eysenbach et al. (2018); Achiam et al. (2018) does not optimize the learned skill for the environment rewards; Osa et al. (2022) does not optimize the mutual information along trajectories; Kumar et al. (2020) needs to solve the optimization problem first before finding a diverse set of solutions. Moreover, these methods fix the latent distributions, limiting their ability to achieve optimality when rewards are given. Mazzaglia et al. (2022) also learns skills within a learned world model. However, it decouples the exploration and skill learning and needs offline data or data generated from other exploration policies to train the model. In contrast, we are motivated by the parameterization problems in *online* RL and jointly optimize the latent representation to model *optimal* trajectories. We show that learning a latent variable model benefits optimization and exploration and they can be considered together.

**Hierarchical Methods** The hierarchical methods, e.g., option-critic (Bacon et al., 2017), can be regarded as a special way of policy parameterization by conditioning the lower-level policy over a sequence of latent variables $z = (z_1, \cdots, z_T)$. Usually, most hierarchical RL methods need special designs for the latent space, e.g., state-based subgoals (Kulkarni et al., 2016; Nachum et al., 2018b;a) or predefined skills (Li et al., 2020) to avoid mode-collapse. Osa et al. (2019) regularized options to maximize the mutual information between the action and the options, which are very relevant to ours. However, it does not model temporal structures as ours to ensure consistency along the trajectories. Goal-conditioned RL (Andrychowicz et al., 2017; Mendonca et al., 2021; Nachum et al., 2018b) can also be considered a special hierarchical method that uses states or goals to help parameterize the policy and has been proven efficient in exploration, but designing the goal space, sampling and generating goals in high-dimensional space is non-trivial. The specific reward design of goal-reaching tasks also makes extending goal-conditioned policies to general reward functions not easy.

Hierarchical imitation learning (Gupta et al., 2019; Pertsch et al., 2021; Shankar & Gupta, 2020; Jiang et al., 2022; Lynch et al., 2020; Fang et al., 2020) extracts temporal abstractions from demonstrations using generative models. For example, InfoGAN (Li et al., 2017) and ASE (Peng et al., 2022) use adversarial training (Goodfellow et al., 2020; Ho & Ermon, 2016) to imitate demonstrations. These works all rely on demonstrations rather than rewards to learn abstractions. Co-Reyes et al. (2018) learns representation on the collected dataset with variational inference and then utilizes the trained model for planning or policy learning. The separation of the representation learning and reward maximization makes it differ from our methods: first, it requires a state reconstruction module to supervise the generative model, which is challenging for high-dimensional observations; second, it optimizes neither the latent distribution nor the actions for the reward directly, thus requires additional planning procedure during the execution to find suitable actions.

## 3. Preliminary

**Markov decision process** A Markov decision process (MDP) is a tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. $p(s'|s,a)$ is the transition probability that transits state $s$ to another state $s'$ after taking action $a$. The function $R(s,a,s')$ computes a reward per transition. A policy $\pi(a|s)$ outputs an action distribution according to the state $s$. Executing a policy $\pi$ starting from the initial state $s_1$ with density $p(s_1)$ will result in a *trajectory* $\tau$, which is a sequence of states and actions $\{s_1, a_1, s_2, \dots, s_t, a_t, \dots\}$ where $a_t \sim \pi(a|s = s_t), s_{t+1} \sim p(s|s = s_t, a = a_t)$. We also use the terminology *environment* to refer to an MDP in an RL problem. The discounted reward of a trajectory is $R_\gamma(\tau) = \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})$ where $0 < \gamma < 1$ is the discount factor to ensure the series converges. The goal of reinforcement learning (RL) is to find a parameterized policy $\pi_\theta$ that maximizes the expected reward $E_{s_1 \sim p(s_1)}[V^{\pi_\theta}(s_1)] = E_{\tau \sim \pi_\theta, s_1 \sim p(s_1)}[R_\gamma(\tau)]$, where $V^{\pi_\theta}$ is the value function. Many environments have an observation space $\mathcal{O}$ that is not the same to the state space. In this case the agent may need to identify the state $s_t$ from the observation $o_t$.

**RL as probabilistic inference** The RL as inference framework (Todorov, 2006; 2008; Toussaint, 2009; Ziebart, 2010; Kappen et al., 2012; Levine, 2018) defines optimality $p(O|\tau) \propto e^{R(\tau)/\mathcal{T}}$, where $\mathcal{T}$ is a temperature scalar and $R(\tau)$ is the total rewards of the trajectory $\tau$. It further defines a prior distribution of the trajectory $p(\tau) = p(s_1) \prod_{t=1}^{T} p(a_t|s_t) p(s_{t+1}|s_t, a_t)$, where $p(a_t|s_t)$ is a known prior action distribution, e.g., a Gaussian distribution. Thus, it can compute the density of optimality $p(O) = \int p(O|\tau)p(\tau)d\tau$. The goal of the framework is to approximate the posterior distribution of optimal trajectories $p(\tau|O) = \frac{p(O|\tau)p(\tau)}{\int p(O|\tau)p(\tau)d\tau}$. In the maximum entropy framework (Haarnoja et al., 2017), one can apply evidence lower bound (Kingma & Welling, 2013) $\log p(O) \geq \mathbb{E}_{\tau\sim\pi} [\log p(O|\tau) + \log p(\tau) - \log \pi(\tau)]$ to train the model.

## 4. Method

To overcome the limitations of single modality policies, we propose to use latent variables to parameterize multimodal policies in Sec. 4.1. We then propose a novel variational bound as the optimization objective to approximate the posterior of optimal trajectories in Sec. 4.2. The variational bound naturally combines maximum entropy RL and includes a term to encourage consistency (Zhu et al., 2017) between the latent distribution and the sampled trajectories, preventing the policy from mode collapse. To optimize this objective in hard continuous control problems, we propose to learn a world model and build the Reparameterized Policy Gradient, a model-based latent variable policy learning framework in Sec. 4.3.1. We design intrinsic rewards in Sec. 4.3.2 to facilitate exploration. Figure 3 illustrates the whole pipeline.

### 4.1. Reparameterize Latent Variables for Multimodal Policy Learning

**Policy parameterization matters.** In continuous RL, it is popular to model action distribution with a unimodal Gaussian distribution. However, theoretically, to make sure that the optimal policy will be captured by RL, the function class of continuous RL policies has to include density functions of arbitrary probabilistic distributions (Sutton & Barto,

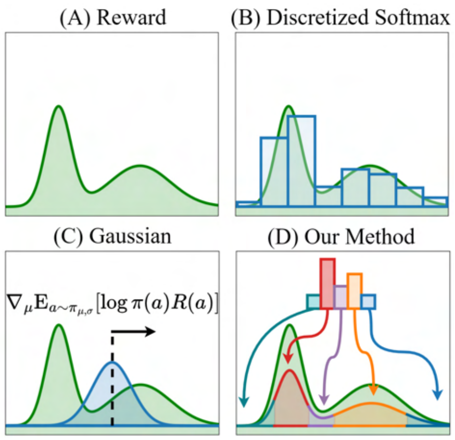


*Figure 2.* (A) rewards; (B); soft max policy over discrete action space; (C) single-modality Gaussian policy; (D) our methods reparameterize a random variable into multimodal distributions with neural networks.

2018). Consider maximizing a continuous reward function with two modalities as shown in Figure 2(A). When the action space is properly discretized, a SoftMax policy can model the multimodal distribution and find the global optimum after sampling over the entire action space as shown in Figure 2(B). However, discretization can lead to a loss of accuracy and efficiency. If we instead use a Gaussian policy $\mathcal{N}(\mu, \sigma^2)$ by the common practice in literature, we will have trouble – as shown in Figure 2(C), even if its standard deviation is so large to well cover both modalities, the policy gradient can push it towards the local optimum on the right side, causing it to fail to converge to the global optimum. To address the issue, a more flexible policy parameterization is needed for continuous RL problems, one that is simple to sample and optimize.

**Multimodal policy by reparameterizing latent variables** Motivated by recent developments in generative models that have shown superiority in modeling complex distributions (Kingma & Welling, 2013; Ho et al., 2020; Rombach et al., 2022; Ramesh et al., 2021), we propose to parameterize policies using latent variables, as illustrated in Figure 2(D). Instead of adding random noise to perturb network outputs to generate an action distribution, we build a generative model of policy distribution by taking random noise as input and relying on powerful neural networks to transform it into actions of various modalities.

Formally, let $z \in \mathcal{Z}$ be a random variable, which can be either continuous or categorical. We design our "policy" as a joint distribution $\pi_\theta(z, \tau)$ of the latent $z$ and the trajectory $\tau$. This paper considers a particular factorization of $\pi_\theta(z, \tau)$ that samples $z$ in the beginning of each episode and then sample trajectory $\tau$ conditioning on $z$:

$$\pi_\theta(z, \tau) = p(s_1)\pi_\theta(z|s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|z, s_t) \quad (1)$$

where $T$ is the length of the sampled trajectory.

One can use the policy gradient theorem (Sutton & Barto, 2018), i.e., $\nabla J(\pi) = \mathbb{E}_\tau[R(\tau)\nabla \log p(\tau)]$ to optimize the generative model policy. However, computing $p(\tau)$ needs to marginalize over $z$, i.e., computing $\int_z p(z, \tau)\, \mathrm{d}z$, which is often intractable when $z$ is continuous. Besides, optimizing the marginal distribution $\log\ p(\tau)$ by gradient descent suffers from local optimality issues (e.g., using gradient descent to optimize Gaussian mixture models which have latent variables is not effective, so EM is often used instead (Ng, 2000)).

## 4.2. Variational Inference for Optimal Trajectory Modeling

To overcome these obstacles, following Todorov (2006; 2008); Toussaint (2009); Ziebart (2010); Kappen et al. (2012); Levine (2018); Haarnoja et al. (2018), we adopt variational method (maximum entropy RL) to directly optimize the joint distribution of the optimal policy without hassles of integrating over $z$.

**The evidence lower bound** We learn $\pi_\theta(z, \tau)$ using variational inference (Kingma & Welling, 2013; Haarnoja et al., 2018; Moon, 1996). Like an EM algorithm, we define an auxiliary distribution $p_\phi(z|\tau)$ to approximate the posterior distribution of $z$ conditioning on $\tau$ using function approximators. This auxiliary distribution $p_\phi(z|\tau)$ helps to factorize the joint distribution of optimality $O$, latent $z$, and the trajectory $\tau$ as $p_\phi(O, z, \tau) = p(O|\tau)p_\phi(z|\tau)p(\tau)$. Treating $\pi_\theta(z, \tau)$ as the variational distribution, we can write the Evidence Lower Bound (ELBO) for the optimality $O$:

$$\begin{aligned}
&\log p(O)\\
&= \underbrace{E_{z,\tau\sim\pi_\theta}\left[\log p_\phi(O, z, \tau) - \log \pi_\theta(z, \tau)\right]}_{\text{ELBO}}\\
&+ D_{KL}(\pi_\theta(z, \tau)||p_\phi(z, \tau|O))\\
&\geq E_{z,\tau\sim\pi_\theta}\left[\log p_\phi(O, \tau, z) - \log \pi_\theta(z, \tau)\right]\\
&= E_{z,\tau\sim\pi_\theta}\left[\log p(O, \tau) + \log p_\phi(z|\tau) - \log \pi_\theta(z, \tau)\right]\\
&= E_{z,\tau}\left[\underbrace{\log p(O|\tau)}_{\text{reward}} + \underbrace{\log p(\tau)}_{\text{prior}} + \underbrace{\log p_\phi(z|\tau)}_{\text{cross entropy}} - \underbrace{\log \pi_\theta(z, \tau)}_{\text{entropy}}\right]
\end{aligned} \quad (2)$$

If we optimize $\pi_\theta(z, \tau)$ and $p_\phi(z|\tau)$ using the gradient of the variational bound, the variational distribution $\pi_\theta(z, \tau)$ learns to model the optimal trajectory distribution $p(\tau|O)$.

**How it works** ELBO contains four parts that can all be computed directly given the sampled $z$ and $\tau$ (the environment probability $p(s_{t+1}|s_t,a_t)$ is canceled as in (Levine, 2018)). The first two parts are the predefined reward $\log\, p(O|\tau) = R(\tau)/\mathcal{T} + c$, where $\mathcal{T}$ is the temperature scalar, and $c$ is the normalizing constant that can be ignored in optimization. The prior distribution $p(\tau)$ is assumed to be known. The third part is the log-likelihood of $z$, defined by our auxiliary distribution $p_\phi(z|\tau)$. It is easy to see that if we fix $\pi_\theta$, maximize $p_\phi$ alone will minimize the cross-entropy $E_{z,\tau\sim\pi_\theta}[-\log p_\phi(z|\tau)]$, similar to the supervised learning of predicting $z$ given $\tau$. This achieves optimality when $p_\phi(z|\tau) = p_\theta(z|\tau) = \frac{\pi_\theta(z,\tau)}{\int_z \pi_\theta(z,\tau)dz}$, modeling the posterior of $z$ for $\tau$ sampled from $\pi_\theta$. On the other hand, by fixing $\phi$, the policy $\pi_\theta$ is encouraged to generate trajectories that are easy to identify or classify; this helps to increase diversity and enforce consistency to avoid mode collapse, letting the network not ignore the latent variables. The fourth part is the policy entropy that enables maximum entropy exploration. Maximizing all terms together for the parameters $\theta$ and $\phi$ will minimize $D_{KL}(\pi_\theta(z,\tau)||p_\phi(z,\tau|O)) = D_{KL}(\pi_\theta(z,\tau)||p_\phi(z|\tau)p(\tau|O))$. The optimality can be achieved when $p_\phi(z|\tau)$ equals to $p(z|\tau)$, the true posterior of $z$. Then, $p_\theta(\tau) = p_\phi(z|\tau)p(\tau|O)/p(z|\tau) = p(\tau|O)$ where $p_\theta(\tau) = \int \pi_\theta(\tau,z)dz$ is the marginal distribution of $\tau$ sampled from $\pi_\theta$.

**Relationship with other methods** Our method is closely related to skill discovery methods (Eysenbach et al., 2018; Mazzaglia et al., 2022). A skill discovery method usually uses mutual information $I(\tau,z) = H(\tau) - H(\tau|z)$ or $H(z) - H(z|\tau) \geq E_{z,\tau}[\log p_\phi(z|\tau) - \log p(z)]$ to encourage diversity. For example, DIYAN (Eysenbach et al., 2018) directly optimizes mutual information to learn various skills without reward. Dropping out the reward term in Eq. 2 shows that the skill learning objective can be seamlessly embedded into the "RL as inference" framework with external reward, and there is no need to introduce the mutual information term manually. Furthermore, the framework suggests we can model the posterior of the optimal trajectories, which enables us to unify generative modeling and trajectory optimization in a single framework. As for the relationship of our method with other generative models, we refer readers to a more thorough discussion in Appendix F.

### 4.3. Reparameterized Policy Gradient for Model-based Exploration

We now describe Reparameterized Policy Gradient (RPG), a model-based RL method with intrinsic motivation for sample efficient exploration in continuous control environments. We first simplify the right side of Eq. 2 using the factorization in Eq. 1 and assuming $\log p_\phi(z|\tau) = \sum_{t>0}\log p(z|s_t,a_t)$. Thus, the ELBO becomes $-\log\pi_\theta(z|s_1) + \sum_{t=1}^{\infty} R(s_t,a_t)/\mathcal{T} - \log\pi_\theta(a_t|s_t,z) + \log p_\phi(z|s_t,a_t)$, which can be optimized with an RL algorithm by maximizing the reward

$$\underbrace{R(s_t,a_t)/\mathcal{T}}_{r_t}\underbrace{-\alpha\log\pi_\theta(a_t|s_t,z) + \beta\log p_\phi(z|s_t,a_t)}_{r'_t},$$

where scalars $\alpha,\beta$ control the exploration and consistency. We use neural networks to model $\log p_\phi(z|s_t,a_t)$ and $\pi_\theta(a_t|s_t,z)$.

#### 4.3.1. Model-based RL with Latent Variables

In our method Reparameterized Policy Gradient (RPG), we train a differentiable world model (Hafner et al., 2019; Schrittwieser et al., 2020; Ye et al., 2021; Hansen et al., 2022) to improve data efficiency. The world model contains the following components: observation encoder $s_t = f_\psi(o_t)$, reward predictor $r_t = R_\psi(s_t,a_t)$, Q value $Q_t = Q_\psi(s_t,a_t,z)$ and dynamics $s_{t+1} = h_\psi(s_t,a_t)$.

Given any $z$ and latent state $s_{t_0} = f_\psi(o_{t_0})$ at time step $t_0$, the learned dynamics network can generate an imaginary trajectory for any action sequence. If we sample actions from the policy $\pi_\theta(a_t|s_t,z)$ for $t\geq t_0$ and execute them in the latent model, it will produce a Monte-Carlo estimate for the value of $s_{t_0}$ for optimizing the policy $\pi_\theta$:

$$V_{\text{est}}(o_{t_0},z) \approx \gamma^K(Q_{t_0+K} + r'_{t_0+K}) + \sum_{t=t_0}^{t_0+K-1}\gamma^{t-t_0}(r_t + r'_t) \tag{3}$$

We self-supervise the dynamics network to ensure state consistency without reconstructing observations as in (Ye et al., 2021; Hansen et al., 2022). For any latent variable $z$ and trajectory segments of length $K+1$ $\tau_{t_0:t_0+K} = \{o_{t_0}, a^{gt}_{t_0}, r^{gt}_{t_0}, o_{t_0+1},\dots,o_{t_0+K}\}$ sampled from the replay buffer, we execute actions $\{a^{gt}_t\}$ in the world model and use the following loss function to train the world model, as well as the $Q$ function:

$$\begin{aligned}L_\psi(\tau) = &\sum_{t=t_0}^{t_0+K-1} L_1\|s_{t+1} - \mathbf{ng}(f_\psi(o_{t+1}))\|^2 + L_2(r_t - r^{gt}_t)^2\\ &+ L_3(Q_t - \mathbf{ng}(r^{gt}_t + \gamma V_{\text{est}}(o_{t+1},z)))^2\end{aligned} \tag{4}$$

where $\mathbf{ng}(x)$ means stopping gradient and $L_1 = 1000, L_2 = L_3 = 0.5$ are constants to balance the loss.

#### 4.3.2. Maximize State Entropy with Object-centric Randomized Network Distillation

For challenging continuous control tasks with sparse rewards, policies that maximize the action entropy of $\pi_\theta(a|s,z)$ usually have trouble obtaining a meaningful reward, making its exploration inefficient. We follow (Hazan et al., 2019) to let the policy additionally maximize

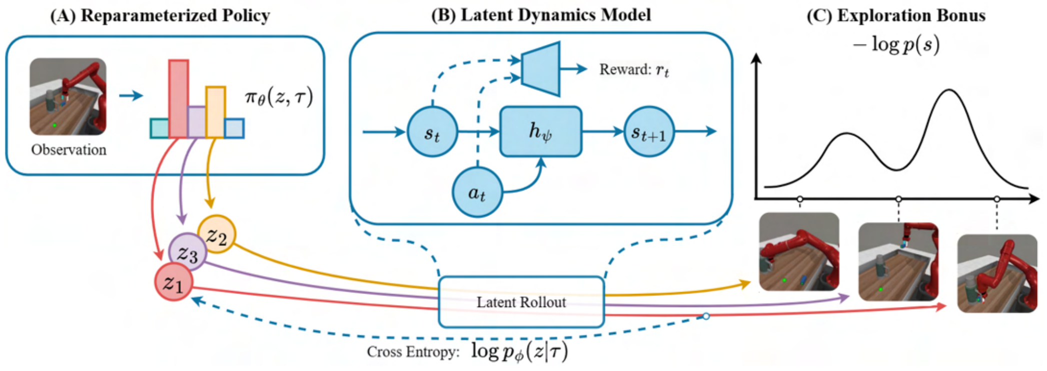


*Figure 3.* An overview of our model pipeline: A) a reparameterized policy from which we can sample latent variable $z$ and action $a$ given the latent state $s$; B) a latent dynamics model which can be used to forward simulate the dynamic process when a sequence of actions is known. C) an exploration bonus provided by a density estimator. Our Reparameterized Policy Gradient do multimodal exploration with the help of the latent world model and the exploration bonus.

the entropy of the discounted stationary state distribution $d_\pi(s) = (1-\gamma)\sum_{t=1}^{\infty}\gamma^t P(s_t = s|\pi)$.

We use the *object-centric* Randomized Network Distillation (RND) (Burda et al., 2018) as a simple and effective method to approximate the state density in continuous control tasks. RND uses a network $g_\theta(o_t)$ to distill the output of a random network $g'(o_t)$ by minimizing the difference $\|g_\theta(o_t) - g'(o_t)\|^2$ over states sampled by the current agent and treat the difference as the negative density of each observation $o_t$.

We make several modifications to the vallina RND to improve its performance for state vector observations in control problems. First, we inject object-prior to the RND estimator to make the policy sensitive to regions that include objects' position change. Specifically, before feeding objects' coordinates into the network, we apply positional encoding (Vaswani et al., 2017; Mildenhall et al., 2021) to turn all scalars $x$ to a vector of $\{\sin(2^i x), \cos(2^i x)\}_{i=1,2,...}$ for objects of interest (e.g., in robot manipulation, the end effector of the robot and the object). Second, we use a large replay buffer to store past states to avoid catastrophic forgetting (Zhang et al., 2021). We verified that it is necessary to normalize the RND's output to stabilize the training and make it an approximated density estimator. Lastly, to account for the latent world model, we relabel trajectories' rewards sampled from the replay buffer instead of estimating them directly in the latent model by reconstructing the observation.

An implicit benefit of a latent variable policy model is its ability to maximize the state entropy better, as will be shown in the experiments of Sec. 5.1. When combined with our RND method, RPG achieves much better state coverage while single modality policy cannot stabilize. The combination of multimodal policy learning and state entropy maximization accelerates the exploration of continuous control tasks with sparse rewards. We describe the whole algorithm in Alg. 1 and implementation details in Appendix A.

## 5. Experiments

In this section, we first illustrate the potential of RPG in optimization and exploration through two example tasks. We then show that our method can help solve hard continuous control problems, even with only sparse rewards. We ablate essential design choices and provide additional experiments in section 5.3.

### 5.1. Illustrative Experiments

**Can multimodal policies help escape local optima?** We study the effects of our method on a 1D bandit problem as shown in Fig. 4. It has a 1d action space and a non-convex reward landscape with an additional discontinuous point.

Fig. 4 compares the performance of our method with a single modality Gaussian policy optimized by REINFORCE. *Notice that we do not add the intrinsic reward for dense reward maximization tasks*. The Gaussian policy, initialized at 0 with a large standard deviation, can cover the whole solution space. However, the gradient w.r.t $\mu$ is positive, which means the action probability density will be pushed towards the right, as the expected return on the right side is larger than the left side, although the left side contains a higher extreme value. As a result, the policy will move right and get stuck at the local optimum with a low chance of jumping out. In contrast, under the entropy maximization formulation, our method maximizes the reward while seeking to increase diversity, providing more chances for the policy to explore the whole solution space. Furthermore, by turning the latent variables into action distribution, our method can build a multimodal policy distribution that fits the multimodal rewards, explore both modalities simultaneously, and

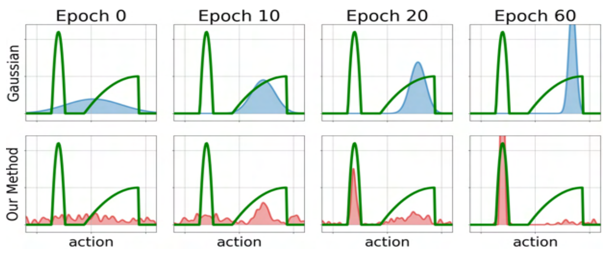


*Figure 4.* Illustrative experiment on continuous bandit

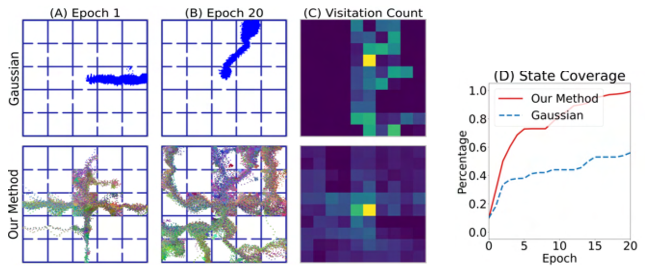


*Figure 5.* Illustrative experiment on 2D maze navigation problem

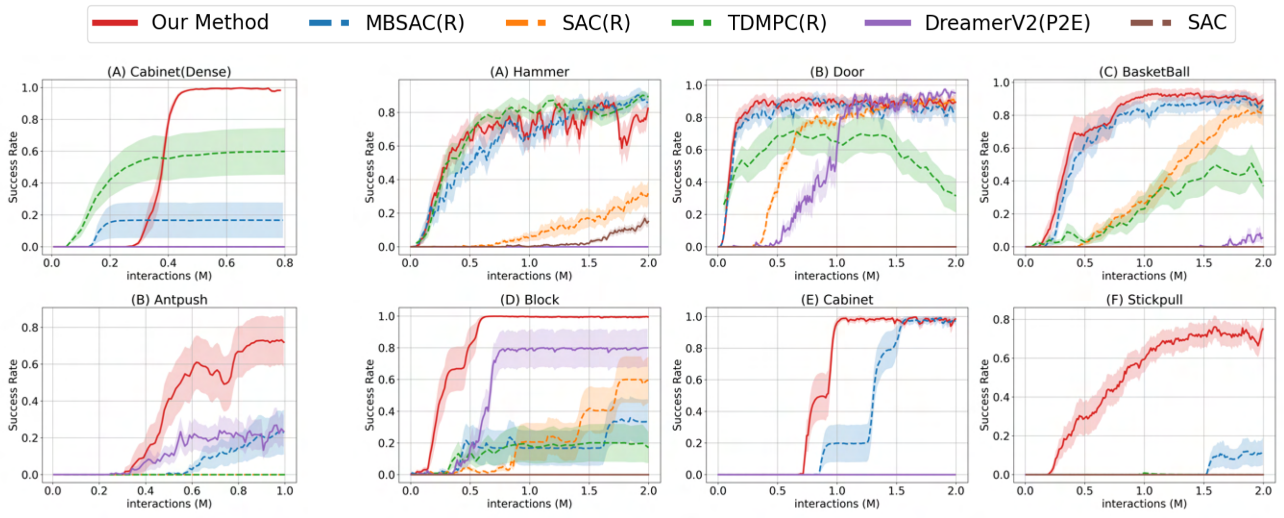


*Figure 6.* Results on dense reward tasks with local optima (exploration disabled)

*Figure 7.* Results on sparse reward tasks

eventually stabilize at the global optimum. This experiment suggests that a multimodal policy is necessary for reward maximization, and our method can help the policy better handle local optima.

**Can multimodal policies accelerate exploration?** We argue that maintaining a multimodal policy is beneficial even in the existence of an intrinsic reward to guide the exploration. We illustrate it in a 2D maze navigation task shown in Figure 5. The maze consists of $5 \times 5$ grids. Each of them is connected with neighbors with a narrow passage. The agent starts in the center grid and can move in four directions. The action space is its position change in two directions $(\Delta x, \Delta y)$.

We apply RPG and single-modality model-based SAC (Haarnoja et al., 2018) on this environment to maximize the intrinsic reward described in Sec. 4.3.2. We count the areas covered by the two policies during exploration with respect to the number of samples in Fig. 5(D). The curve suggests that our method explores the domain much faster, quickly reaching most grids, while the Gaussian agent only covers the right part of the maze within a limited sample budget.

To understand their differences, we visualize states sampled at different training steps of the two policies in Fig. 5 (A-B). Our policy below quickly finds four directions to move and gradually expands the state distribution until it fully occupies all grids. Fig. 5(C) shows the historic state visitation count. It is easy to see that our multimodal policy induces a more uniform distribution over the whole state space, generating a higher state distribution entropy. The optimization procedure of single-modality policy, as shown in the first row of Fig. 5, suffers from its policy parameterization. It can only explore one modality every time and has to switch modalities one by one, where modalities refer to different regions of the state space. It is hard to predict when it switches modality, making algorithms behave vastly differently in different environments with different random seeds. Sometimes it moves slowly from one direction to another because it has to wait for samples for density estimators to generate enough momentum. As a result, it never explores the left side in Fig. 5(C). While sometimes, it switches too fast due to the fast updates of the network and does not exploit some modalities enough, missing far-end grids of certain directions that it has explored once. This also causes issues when maximizing external rewards. Even if a single-modal policy finds the optimal solution, it may switch to another modality to continue exploration and it is hard to guarantee that it would come back in the end. In contrast, our method is more like Monte-Carlo sampling, which samples all candidates while converging to solutions of high rewards with high probability.

## 5.2. Continuous Control Problems

We now verify if our method can scale up and help solve challenging continuous control problems. We take 8 representative environments from standard RL benchmarks, including 2 table-top environments from MetaWorld (Yu et al., 2020), 2 dexterous hand manipulation tasks from Rajeswaran et al. (2017), 1 navigation problems from Nachum et al. (2018b), and 2 articulated object manipulation from ManiSkill (Mu et al., 2021). We show environment examples and provide a detailed environment description in Appendix C. Only **Cabinet (Dense)** and **AntPush** contain dense rewards that lead to local optima. *The remaining* 6 *environments all only provide sparse rewards*, which means the agents receive a reward 1 when it succeeds to finish the task and 0 otherwise. This change dramatically increases the difficulty of these environments and disastrously hurts the performance of classical RL methods like SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017).

We evaluate our methods against the following baselines: DreamerV2 + Plan2Explore (Sekar et al., 2020), abbreviated as **DreamerV2 (P2E)**, a model-based exploration method based on the disagreement of learned models' prediction. We also consider 3 baselines, **TDMPC**, **MBSAC**, and **SAC** using the same intrinsic rewards as ours. The suffix **(R)** means that when we apply these methods to a sparse-reward environment, we will add RND intrinsic rewards that are the same as in our method. For all results evaluated on dense-reward environments in Figure 6, the exploration method of the corresponding algorithm is disabled. The standard SAC without intrinsic rewards validates the difficulty of our tasks. Details of the baseline implementations are in Appendix D.

Fig. 6 and 7 plots the learning progress of each algorithm in all environments (x-axis: number of environment interaction steps in million, y-axis: task success rate). For all environments, we run each algorithm for at least five trials. The curve and the shaded region shows the average and the standard deviation of performance over trials. ***MBSAC** shares almost the same implementation as our method, except that it does not condition its policy on latent variables..*

We first observe that, for dense reward tasks, our method largely improves the success rate on tasks with local optima (Fig. 6). We can see that in both **AntPush** and **Cabinet (Dense)** tasks, our method outperforms all baselines. Our method consistently finds solutions, regardless of the local optima in the environments. For example, in the task of opening the cabinets' two doors and going to the two sides of the block, our method usually explores the two directions simultaneously and converges at the global optima. In contrast, other methods' performance highly depends on their initialization. If the algorithm starts by opening the wrong doors or pushing the block in the wrong direction, it will not escape from the local minimums; thus, its success rates are low.

Our methods successfully solve the 6 sparse reward tasks as shown in Fig. 7. Especially, it consistently outperforms the **MBSAC(R)** baseline, which is a method that only differs from ours by the existence of latent variables to parameterize the policy. Our method reliably discovers solutions in environments that are extremely challenging for other methods (e.g., the **StickPull** environment), clearly demonstrating the advantages of our method in exploration. Notably, we find that **MBSAC(R)**, which is equipped with our object-centric RND, is a strong baseline that can solve **AdroitHammer** and **AdroitDoor** faster than **DreamerV2(P2E)**, proving the effectiveness of our intrinsic reward design. **TDMPC(R)** has a comparable performance with **MBSAC(R)** on several environments. We validate that it has a faster exploration speed in Adroit Environments thanks to latent planning. We find that the **Dreamer(P2E)** does not perform well except for the **BlockPush** environment without the object prior and is unable to explore the state space well. We visualize modalities explored by our method in Appendix E.

## 5.3. Additional Experiments

**Ablation study** We analyze various factors influencing the performance of our method in the Maze navigation task in Section 5.1. More detailed discussion and experiment results are in Appendix B. Experimental comparisons between different latent spaces show that a Gaussian distribution of dimension 12 outperforms the categorical latent space, both surpassing a baseline that does not use latent variables. A moderate latent space size $\geq 6$ is found to be sufficient, with performance declining if the latent dimensions are too small. In terms of reward maximization, the weight of the cross-entropy term ($\beta$) is crucial, with results indicating an ideal range between 0.001 and 0.01 for the RND design. Furthermore, the performance from RND is tied to maintaining a large replay buffer and using positional embedding, with a lack of either resulting in degraded exploration. A comparative analysis of policy parameterization methods shows the superiority of the vanilla Gaussian policy over the Gaussian Mixture Models (GMM) and CEM-based policy. The latter two display several optimization issues; GMM struggles with log-likelihood maximization, and CEM, despite its proficiency at finding local optima, tends to sacrifice its explorative capabilities. Finally, normalizing flow showed initial promise but soon encountered numerical instabilities, highlighting the need for further investigation.

**Evaluation on locomotion environments** We modified the HalfCheetah-v3 environment in OpenAI Gym (Brockman et al., 2016) to study the performance of our methods in locomotion tasks, shown in Figure 11 in the appendix. The cheetah robot moves backward for a certain distance to receive a sparse reward of 1 to succeed. Our exploration method was able to effectively aid the exploration of the Cheetah robot and solve the task easily while removing the exploration term that led to the agent getting stuck. However, in this particular task, modeling multi-modal exploration did not increase the sampling efficiency, as there were only two modalities (moving forward and backward), and model-based SAC could exploit the two modes one by -one and solve the task. This made the advantage of our method negligible in this case. We also evaluated our method compared to SAC (Haarnoja et al., 2018) on the standard Mujoco environments. Results are shown in Fig. 12.

**Vision-based RL** As a proof of concept, we illustrate, in Fig. 13, the potential of our method for image observations in a single-block pushing environment: the observation consists of two consecutive 64x64 RGB images; the agent needs to control the red block to push the purple box into the target region. We use 4-layer convolutional networks as the encoder for both the policy network and RND estimator. We compare our method with model-based SAC (RND), which has an intrinsic reward to guide exploration but only models single modality policies, and model-based SAC without RND. The result validates our method's effectiveness.

## 6. Limitation and Future Work

Our approach capitalizes on the advantages offered by multiple components, effectively addressing complex exploration issues in continuous spaces. However, it also introduces certain hurdles and constraints. For instance, our intrinsic reward is predicated on assumptions regarding the recognition of objects and their spatial positioning. This approach may be unsuitable in environments with unidentified objects or where observations don't plainly reveal object-related information, akin to scenarios in vision-based RL; Learning the world model typically results in a slower pace of gradient updates; Incorporating a cross-entropy network adds an extra layer of complexity to the network design and training. Therefore, it is worth discussing potential future directions that might address these limitations.

**Object-centric learning for vision-based RL** While the Random Network Distillation (RND) is initially tailored for image observations, integrating object-centric design to accelerate exploration in vision-based RL will be an interesting direction. This suggests two typical strategies to apply our method to tasks with vision observations: (1) The first involves directly encoding observations without considering object information. It proves effective in scenarios with no occlusion and a static background, wherein objects emerge as the sole salient feature of the input. We provide a proof-of-concept experiment in Section 5.3. (2) The second approach harnesses computer vision techniques to identify objects for object-centric exploration. This includes applying recent large-scale vision foundation models, which possess zero-shot object detection capabilities as outlined in (Zhang et al., 2022) or leveraging slot-attention for object discovery as described in (Locatello et al., 2020).

**Combining with previous model-based control and planning methods** Instead of learning the world model from on-policy data, we can pre-train a physical world model (Li et al., 2019) or use analytical models (Posa et al., 2014; Huang et al., 2021) to gain generalizability and efficiency. Moreover, we drew inspiration from RRT-like motion planners (Karaman & Frazzoli, 2011) to derive our policy to sample over the configuration space and bias the exploration towards significant kinematics changes. Thus, an exciting direction is incorporating structures in model-based control into RL algorithms, including temporal structures like dynamics motion primitives (Stulp & Sigaud, 2013) and semantic information from TAMP (Garrett et al., 2021).

**Extending to other probabilistic models** Our method can be viewed as variational inference (Ranganath et al., 2014) over a particular stochastic computation graph (Weber et al., 2019). The computation graph contains hidden variables, and we use the Bellman equation and a learned model to estimate its gradient. This provides a new perspective that bridges online Reinforcement Learning (RL) with generative models and sequence modeling. In the future, we are interested in exploring how sequence-modeling techniques, such as transformers and hierarchical methods, can be used to model the policy in our framework.

## 7. Conclusion

We derive a framework that models the policy of continuous RL by a multimodal distribution in the variational inference framework. The method reparameterizes latent variables into trajectories like generative models. Under this framework, we learn a world model to help learn multimodal policy data efficiently. Incorporating an object-centric intrinsic reward, our method can solve challenging continuous control problems with little to no reward signal.

## Acknowledgement

This work is in part supported by Qualcomm AI and AI Institute for Learning-Enabled Optimization at Scale (TILOS).

## References

Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Campos, V., Trott, A., Xiong, C., Socher, R., Giró-i Nieto, X., and Torres, J. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pp. 1317–1327. PMLR, 2020.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Co-Reyes, J., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International conference on machine learning*, pp. 1009–1018. PMLR, 2018.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. doi: https://doi.org/10.1111/j.2517-6161.1977.tb01600.x. URL `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1977.tb01600.x`.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Fang, K., Zhu, Y., Garg, A., Savarese, S., and Fei-Fei, L. Dynamics learning with cascaded variational inference for multi-step manipulation. *arXiv preprint arXiv:1910.13395*, 2019.

Fang, K., Zhu, Y., Garg, A., Savarese, S., and Fei-Fei, L. Dynamics learning with cascaded variational inference for multi-step manipulation. In *Conference on Robot Learning*, pp. 42–52. PMLR, 2020.

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., and Lozano-Pérez, T. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT Press, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., Yuan, X., Xie, P., Huang, Z., Chen, R., and Su, H. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Huang, Z., Hu, Y., Du, T., Zhou, S., Su, H., Tenenbaum, J. B., and Gan, C. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021.

Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022.

Kappen, H. J., Gómez, V., and Opper, M. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

Karaman, S. and Frazzoli, E. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Kumar, S., Kumar, A., Levine, S., and Finn, C. One solution is not all you need: Few-shot extrapolation via structured maxent rl. *Advances in Neural Information Processing Systems*, 33:8198–8210, 2020.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Li, C., Xia, F., Martin-Martin, R., and Savarese, S. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, pp. 603–616. PMLR, 2020.

Li, Y., Song, J., and Ermon, S. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in Neural Information Processing Systems*, 30, 2017.

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*, 2019.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. In *Conference on robot learning*, pp. 1113–1132. PMLR, 2020.

Mazzaglia, P., Verbelen, T., Dhoedt, B., Lacoste, A., and Rajeswar, S. Choreographer: Learning and adapting skills in imagination. *arXiv preprint arXiv:2211.13350*, 2022.

Mendonca, R., Rybkin, O., Daniilidis, K., Hafner, D., and Pathak, D. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Moon, T. K. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

Mu, T., Ling, Z., Xiang, F., Yang, D., Li, X., Tao, S., Huang, Z., Jia, Z., and Su, H. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

Nachum, O., Gu, S., Lee, H., and Levine, S. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018a.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018b.

Ng, A. Cs229 lecture notes. *CS229 Lecture notes*, 1(1):1–3, 2000.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019. URL `http://arxiv.org/abs/1910.07113`.

Osa, T., Tangkaratt, V., and Sugiyama, M. Hierarchical reinforcement learning via advantage-weighted information maximization. *arXiv preprint arXiv:1901.01365*, 2019.

Osa, T., Tangkaratt, V., and Sugiyama, M. Discovering diverse solutions in deep reinforcement learning by maximizing state–action-based mutual information. *Neural Networks*, 152:90–104, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Peng, X. B., Guo, Y., Halper, L., Levine, S., and Fidler, S. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *arXiv preprint arXiv:2205.01906*, 2022.

Pertsch, K., Lee, Y., and Lim, J. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Posa, M., Cantu, C., and Tedrake, R. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1): 69–81, 2014.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial intelligence and statistics*, pp. 814–822. PMLR, 2014.

Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pp. 8857–8868. PMLR, 2021.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8583–8592. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sekar20a.html.

Shankar, T. and Gupta, A. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pp. 8624–8633. PMLR, 2020.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. PMLR, 2014.

Stulp, F. and Sigaud, O. Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn, Journal of Behavioral Robotics*, 4(1):49–61, 2013.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Todorov, E. Linearly-solvable markov decision problems. *Advances in neural information processing systems*, 19, 2006.

Todorov, E. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pp. 4286–4292. IEEE, 2008.

Toussaint, M. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 1049–1056, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553508. URL https://doi.org/10.1145/1553374.1553508.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Weber, T., Heess, N., Buesing, L., and Silver, D. Credit assignment techniques in stochastic computation graphs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2650–2660. PMLR, 2019.

Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11097–11107, 2020.

Ye, W., Liu, S., Kurutach, T., Abbeel, P., and Gao, Y. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Zhang, H., Zhang, P., Hu, X., Chen, Y.-C., Li, L., Dai, X., Wang, L., Yuan, L., Hwang, J.-N., and Gao, J. Glipv2: Unifying localization and vision-language understanding. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 36067–36080. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/ea370419760b421ce12e3082eb2ae1a8-Paper-Conference.pdf`.

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., and Tian, Y. Noveld: A simple yet effective exploration criterion. *Advances in Neural Information Processing Systems*, 34:25217–25230, 2021.

Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. *arXiv preprint arXiv:2202.05607*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

## A. Implementation Details

**Network architecture** We use the following two-layer MLP to model policy $\pi_\theta$, value $Q_\psi$, state encoder $f_\psi$, and the encoder $p_\phi(z|s)$. The network structures are shown in the pytorch's convention (Paszke et al., 2019).

```
Sequential(
  (0): Linear(in_features=inp_dim, out_features=256, bias=True)
  (1): ELU(alpha=1.0)
  (2): Linear(in_features=256, out_features=256, bias=True)
  (3): ELU(alpha=1.0)
  (4): Linear(in_features=256, out_features=out_dim, bias=True)
)
```

The dynamics network is a single-layer GRU with a hidden dimension 256. The RND network $g_\theta$ we use is a 3 layer MLP network with hidden dimension 512 and leaky ReLU as its activation function.

We maintain target networks like the standard double Q learning. The hyperparameters for training the network are listed in Table 1.

| Hyperparameter | Value |
|---|---|
| Discount factor ($\gamma$) | 0.99 |
| Seed step | 1000 |
| Replay buffer size | 800000 |
| Model rollout horizon ($H$) | 3 |
| Action distribution | Tanh Normal |
| Entropy target | $-\|\mathcal{A}\|$ |
| Initial entropy coefficient $\alpha$ | 0.01 |
| Cross-entropy coefficient $\beta$ | 0.005 |
| RND coefficient $\beta$ | 0.1 |
| Environment steps per gradient update | 5 |
| Temperature | $\mathcal{T}$ |
| Learning rate | $3 \times 10^{-4}$ |
| Batch size | 512 |
| Target network update ratio | 0.005 |
| Actor update freq | 2 |
| State embedding dimension | 100 |
| grad norm clip | 1.0 |
| Positional encoding dimension | 6 |
| Latent distribution $\mathcal{Z}$ | Normal |
| $\mathcal{Z}$ dimension | 12 |
| $p_\phi(z\|s,a)$ distribution | Normal distribution with std 0.38 |
| $\pi_\theta(z\|s_1)$ | $\mathcal{N}(0,1)$ for sparse reward tasks |

*Table 1.* RPG hyperparameters. We here list the hyper-parameters used in the experiments. The hyper-parameters keep the same for our **MBSAC** baseline except that **MBSAC** has no latent space. Notice that for dense reward tasks, the entropy of $\pi_\theta(z|s_1)$ is linearly decayed starting from $3 \times 10^5$ environment steps to $1M$ steps to ensure optimality.

## B. Ablation Study

We study and compare various factors in our methods in Fig. 8 on the Maze navigation task described in Sec. 5.1. Fig. 8(A) compares different latent spaces to use. The continuous latent space modeled by a Gaussian distribution of dimension 12 outperforms the categorical latent space, while both are better than the one without latent variables, i.e., the **MBSAC** baselines. Fig. 8(B) shows the effects of our method when using a Gaussian distribution as the latent space with different $\beta$ values. The $\beta$ controls the scale of the cross entropy term $\log p_\phi(z|s,a)$ in reward maximization, as mentioned in Sec. 4.3.1.

**Algorithm 1** Model-based Reparameterized Policy Gradient

```
Input: $p_\phi, \pi_\theta, h_\psi, R_\psi, f_\psi, Q_\psi$ and an optional density estimator $g_\theta$
Initialize $p_\phi, \pi_\theta$, construct the replay buffer $\mathcal{B}$.
while time remains do
    Sample start state $o_1$ and encode it as $s_1 = f_\psi(o_1)$. Select $z$ from $\pi_\theta(z|s_1)$.
    Execute the policy $\pi_\theta(a|s, z)$ and store transitions into the replay buffer $\mathcal{B}$.
    Sample a batch of trajectory segment of length $K$ $\{\tau^i_{t:t+K}, z\}$ from the buffer $\mathcal{B}$.
    Optional: update and estimate the density estimator $g_\theta$ and relabel transitions with the negative density as the intrinsic reward.
    Optimize $\psi$ using Equation 4.
    Optimize $\pi_\theta(a|s, z)$ with gradient descent to maximize the value estimate in Equation 3 for $s, z$ sampled from the buffer.
    Optimize $\pi_\theta(z|s_1)$ with policy gradient to maximize $V_{\text{estimate}}(s_1, z) - \alpha \log \pi_\theta(z|s_1)$ for $s_1$ sampled from the buffer.
    Optimize $\alpha, \beta$ if necessary .
end while
```

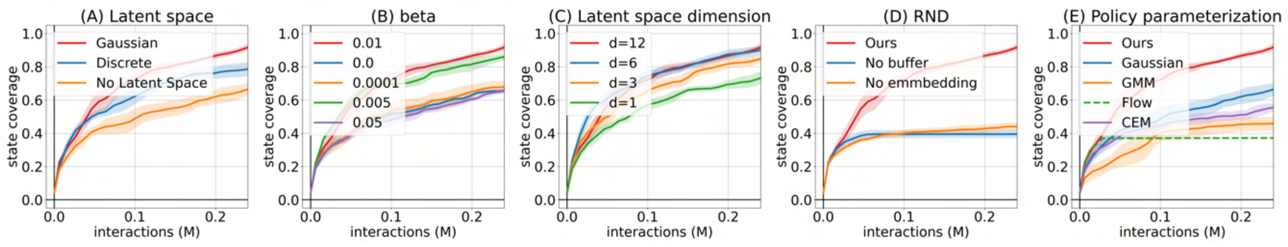
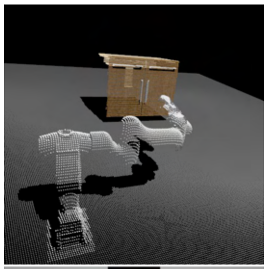


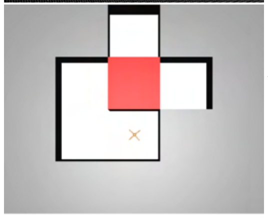
*Figure 8.* Comparing different factors in our methods.

The policy will ignore the latent variable if the $\beta$ is too small, e.g., $0.$, $1e-4$. But if the $\beta$ is too large, though the policy generates diverse solutions, it may explore too much without exploiting past experiences. This $\beta$ plays a similar role as $\beta$ in $\beta-$VAE (Higgins et al., 2017). In experiments, we find that $\beta$ from $0.001$ to $0.01$ works well in the case of our RND design. Fig. 8(C) shows the effects of the latent dimensions. For tasks like $2D$ maze, a moderate latent space size $d \geq 6$ is sufficient. But the performance will degrade when it is too small. Fig. 8(D) ablates our design for the RND. When the RND estimator does not maintain a large replay buffer or does not use the positional embedding, the exploration will suffer a lot. We further compare various policy parameterization methods in Fig. 8(E). We find that in our implementation, Gaussian mixture models (GMM) and CEM-based policy do not perform as well as the vanilla Gaussian policy. GMM may have trouble in log-likelihood maximization. We noticed several numerical issues in optimizing GMM and Flow when we applied them with RND in sparse reward tasks. Specifically, we have encountered some instability when optimizing the log prob for GMM due to its non-convex nature and the need for sampling to estimate entropy. Similarly, our experiments with Flow have revealed significant parameter divergence and instabilities, warranting further investigation to pinpoint the root cause. CEM has a stronger ability to find local optima and generates actions with less randomness, which may sacrifice its ability to do exploration. Besides, we find the policy parameterized by a normalizing flow distribution behaves well initially but soon meets numerical instabilities and fails to proceed with optimization, suggesting more investigations are needed in this direction.

## C. Environment Details



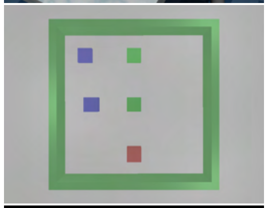
**Cabinet (Dense)** (Gu et al., 2023). The agent controls the movement of a 12 dof mobile robot arm and gripper robot to open both cabinet doors. The agent receives a dense reward for reaching its nearest door's handle. Besides, it receives a higher reward when it opens the right door than the left door. The agent succeeds when it fully opens the right door while the dense reward will typically drive the agent close to the handle of the left door. The episode length is 60.



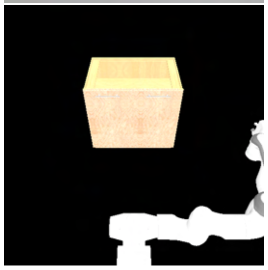
**AntPush** (Nachum et al., 2018b). The agent controls an ant robot with action dimension 8 to go to the upper room. The reward is the $l_2$ distance between the agent and a point in the upper room. The optimal path is to go to the left of the red block and push it to the right and go to the upper room. However, agents often get stuck at the local optima, which pushes the block forward or moves to go to the right side. The episode length is 400.

**Door** (Rajeswaran et al., 2017). The agent controls a dexterous hand with action dimension 26 to open a door. The agent only receives a reward of 1 when it successfully undoes the latch and opens the door. The episode length is 100 with an action repeat 2. Objects of interest include the hand's palm, the latch, and the door.

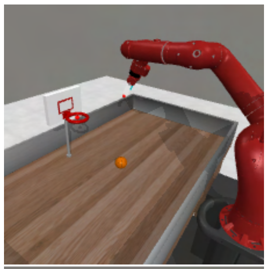
**Hammer** (Rajeswaran et al., 2017). The agent controls a dexterous hand with action dimension 26 to force drive a nail into the board. The agent only receives a reward of 1 when it has driven the nail all the way in. Action repeat is 2. The episode length is 125. We encode the position of the hand's palm, the hammer, and the nail.



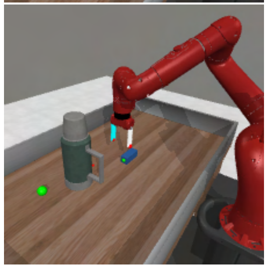
**BlockPush** (Xiang et al., 2020). The agent controls the movement of the red block with action dimension 2 to push the green block (middle) to the green destination (above) and the blue block (middle) to the blue destination (above). The agent only receives a reward of 1 when it has successfully pushed both blocks to the exact destination with a small tolerance. The objects of interest contain the location of the three blocks. The environment horizon is 60.



**Cabinet (Sparse)** (Gu et al., 2023). The agent controls the movement of a 9 dof robot arm and gripper robot to open both doors of the cabinet. The agent only receives a reward of 1 when both cabinet doors are fully opened. We encode the position of the robot's end effector and the location of the cabinet's door. Its episode length is 60.



**Meta-World BaseketBall** (Yu et al., 2020). The agent controls the movement of a gripper with a 4 dof controller to move the ball into the basket. The agent only receives a reward of 1 when the ball is sufficiently close to the basket. The locations of the ball and the location of robots' fingertips are what we are concerned about. The episode length is 100, including 2 action repeats.



**Meta-World StickPull** (Yu et al., 2020). The agent controls the movement of a gripper with a 4 dof controller to pull the container with a blue stick. The agent receives a reward of 1 only when the stick is inserted inside the handle, and the container is already pulled sufficiently close to the green dot. We encode the positions of the fingertips, the stick, and the handle of the cup for computing intrinsic rewards. The remaining setup is the same as **BasketBall**.

## D. Baseline

**TDMPC** (Hansen et al., 2022), we used the publically available official implementation and default hyperparameters provided by the authors at https://github.com/nicklashansen/tdmpc.

**SAC** (Haarnoja et al., 2018), we implemented according to the original paper and used the default hyperparameter provided by the authors.

We use the abbreviation **TDMPC(R)**, **SAC(R)** to represent that we add an intrinsic reward with scale 0.1 for exploration in environments with only sparse rewards.

**DreamerV2** (Hafner et al., 2020), we used the publically available official implementation and default hyperparameters provided by the authors at https://github.com/danijar/dreamerv2.

**Plan2Explore** (Sekar et al., 2020), we run DreamerV2 according to the instructions provided by https://github.com/ramanans1/plan2explore with hyperparameters provided by the authors of the paper.

For all baseline algorithms, we only change model update frequency to once every 5 environment steps.

## E. Visualization of the Multimodal Exploration

We plot the trajectory of the agent in *AntPush* environment, evaluated at different numbers of training stages in Fig. 9. The agent learned to move forward and explored all directions that would decrease the $l_2$ distance. It found the left side was easier for moving up in the beginning, but at episode 360, it learned to explore all directions. Ultimately, it explored the left path to the upper room and converged on it.

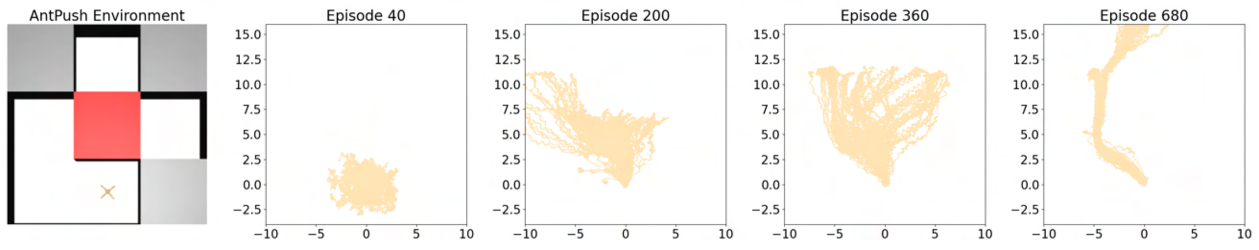


*Figure 9.* Exploration of AntPush, which has the dense reward to guide the agent to move forward.

We also plot the sampled states during exploration for Block, Cabinet, and Stickpull Envs in Fig. 10.

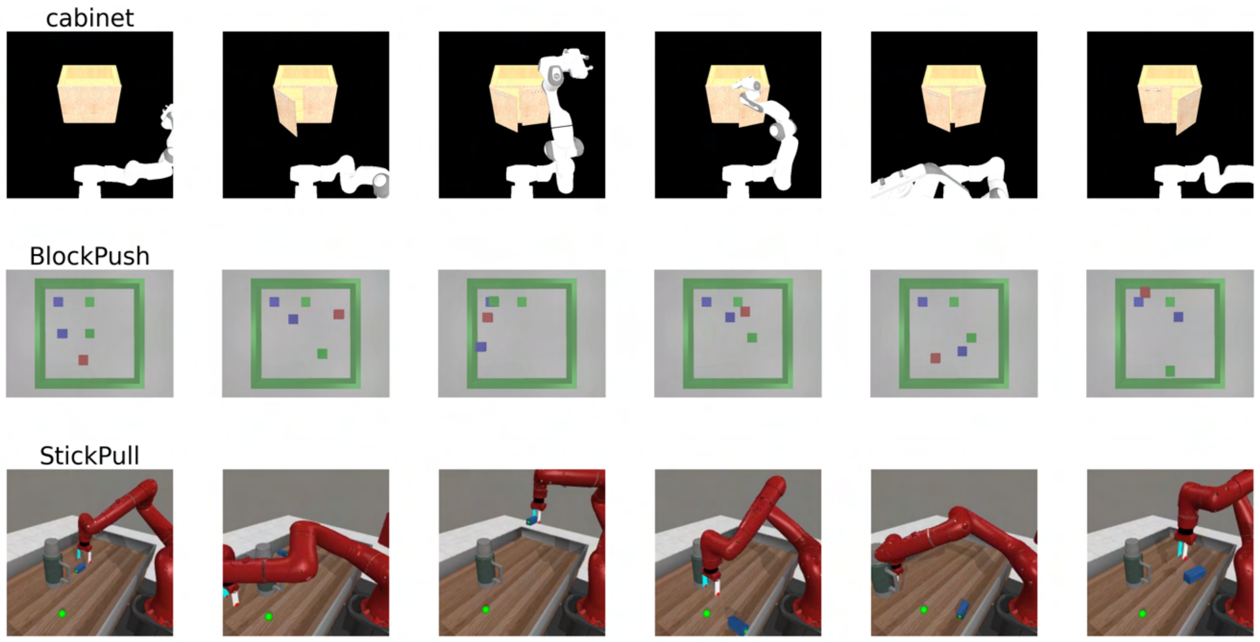


*Figure 10.* Exploration on several environments; The first column shows the initial state. The right 5 figures of the same row plot states sampled from a single agent.

## F. Connection with Other Generative Models

Our method is based on the same variational bound shared with many other generative models

$$\log p(x) = E_{z\sim q(z)}\left[\log p(x,z) - \log q(z)\right] + KL(q(z)||p(z|x)).$$

By different choices of latent space, posterior $q(z|x)$, joint distribution $p(x,z)$, we can obtain different generative models. For example, VAE models $p_\theta(x,z) = p_\theta(x|z)p(z)$ and $q(z) = q_\phi(z|x)$ using neural networks and then optimize $\theta, \phi$ jointly to maximize the ELBO bound. By doing so, $q_\phi(z|x)$ will align with the true posterior of $p_\theta(z|x)$. Thus

$$\log p(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)]$$

The Expectation–maximization algorithm (EM) (Dempster et al., 1977) for learning Gaussian mixture models assumes that we have $p_\theta(x,z) = p_\theta(x|z)p_\theta(z)$ where $z$ is a categorical representation. E-step: finding $q_\phi(z|x)$ by solving $\max_\phi \log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x))$ where $p_\theta(z|x) = p_\theta(x,z)/\int p_\theta(x,z)dz$. M-step: fixing $\phi$, find $\max_\theta E_{q_\phi}[\log p_\theta(x,z)] - E_{q_\phi}[\log q_\phi(z|x)]$ which is exactly maximizing the ELBO.

In Maximum Entropy RL (Levine, 2018), we have optimality $p(O,\tau) = p(O|\tau)p(\tau)$ defined by the reward, and we optimize $\pi_\theta(\tau|O)$ only. The ELBO bound becomes a maximum entropy term $\mathbb{E}_{\tau\sim\pi}\left[\log p(O|\tau) + \log p(\tau) - \log \pi(\tau)\right]$. Our method differs from it by introducing an additional variable $z$. Table 2 compares various generative models.

| | Latent | Encoder $q(z\|x)$ | Joint $p(x,z)$ | MLE objective |
|---|---|---|---|---|
| VAE | $z$ | $p_\phi(z\|x)$ | $p_\theta(x\|z)p(z)$ | $p(x)$ |
| EM | $z$ | $\max_\phi \log p_\theta(x) - D_{KL}(q_\phi(z\|x)\|\|p_\theta(z\|x))$ | $p_\theta(x\|z)p_\theta(z)$ | $p(x)$ |
| Diffusion | $\{x_t\}_{t\geq 1}$ | $\prod_{i=1}^{T}\mathcal{N}(x_t;\sqrt{1-\beta_t}x_{t-1},\beta_t\mathbf{I})$ | $p(x_T)\prod_{t\geq 1}p_\theta(x_{t-1}\|x_t)$ | $p(x_0)$ |
| MaxEntRL | $\tau$ | $\pi_\theta(\tau)$ | $p(O\|\tau)p(\tau)$ | $p(O)$ |
| RPG | $\tau, z$ | $\pi_\theta(z,\tau)$ | $p(O\|\tau)p_\phi(z\|\tau)p(\tau)$ | $p(O)$ |

*Table 2.* Comparison of different algorithms that optimize ELBO bounds for inference

## G. Environments and Results in Additional Experiments

### Cheetah Back

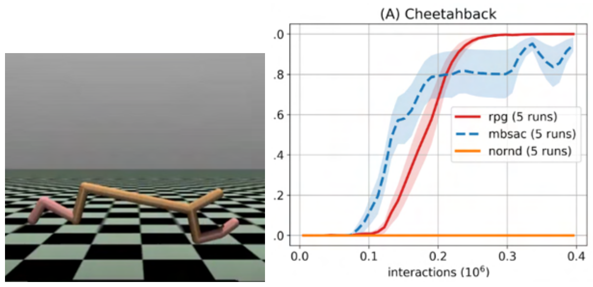


*Figure 11.* Cheetah Back Task (left), success rate (right)

### Standard Mujoco-v2 Environments

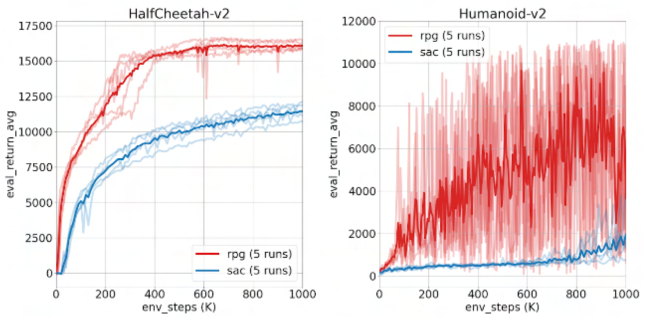


*Figure 12.* Results on Mujoco-v2 Environments

### Vision-based RL

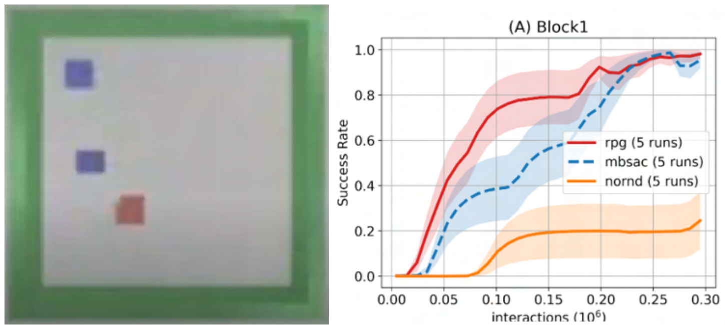


*Figure 13.* Visual Block Push Task (left), success rate (right)

# Robo360: A 3D Omnispective Multi-Material Robotic Manipulation Dataset

Litian Liang[1*] Liuyu Bian[2*] Caiwei Xiao[1] Jialin Zhang[2]
Linghao Chen[1] Isabella Liu[1] Fanbo Xiang[1] Zhiao Huang[1] Hao Su[1]

[1]UC San Diego [2]Tsinghua University
*equal contribution

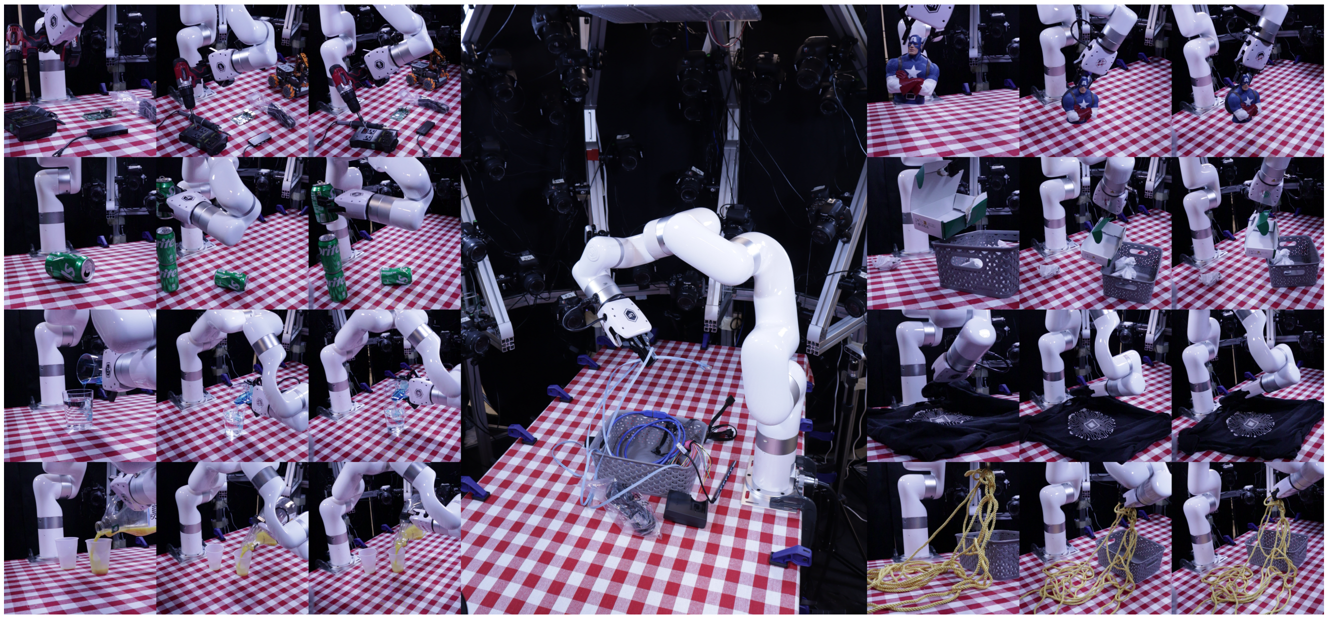


Figure 1. Example multi-view videos in Robo360 dataset.

## Abstract

*Building robots that can automate labor-intensive tasks has long been the core motivation behind the advancements in computer vision and the robotics community. Recent interest in leveraging 3D algorithms, particularly neural fields, has led to advancements in robot perception and physical understanding in manipulation scenarios. However, the real world's complexity poses significant challenges. To tackle these challenges, we present Robo360, a dataset that features robotic manipulation with a dense view coverage, which enables high-quality 3D neural representation learning, and a diverse set of objects with various physical and optical properties and facilitates research in various object manipulation and physical world modeling tasks. We confirm the effectiveness of our dataset using existing dynamic NeRF and evaluate its potential in learning multi-view policies. We hope that Robo360 can open new research directions yet to be explored at the intersection of understanding the physical world in 3D and robot control.*

## 1. Introduction

Mastering robotic manipulation in 3D is crucial for embodied agents. It involves manipulating objects within three-dimensional spaces to accomplish tasks. This field has seen growing interest recently. Advancements in 3D algorithms, particularly neural fields, have significantly enhanced robots' ability to perceive and act in 3D environments [24, 31, 38, 74]. Techniques like [16, 59, 75] use neural fields for neural policy training, facilitating complex decision-making in 3D scenarios. Moreover, [17, 32, 35, 54] employ these fields in world model developments [25, 33, 54].

Despite recent advancements, the real-world complexity poses substantial challenges to robotic systems. These challenges include the need for precise recognition of objects, materials, and deformations in 3D space, and the handling of complex scenarios involving specular and transparent objects. Additionally, real-world objects exhibit diverse physical properties, from rigid bodies to deformables and fluids. Consequently, creating comprehensive world models that support model-based planning and the development of generalizable policies is an intricate task.

To overcome these challenges, the development of *a dataset of dynamic scenes featuring real-world robot manipulation* is crucial. This dataset would propel the development of algorithms and hardware in 3D robotic manipulation, serving as an exhaustive test set for evaluating progress. The dataset should include a diverse range of objects with different visual and physical attributes to enable robust testing across various scenarios. Moreover, it should provide abundant, and ideally, redundant 3D information to facilitate the comparison of robot vision systems with varying algorithmic and hardware configurations. With a one-time investment in collecting such information-rich data, which might be via advanced devices, benefits may be brought to meet varied user requirements. For example, the dataset can help design systems with limited views that are cheap and scalable at deployment time. As another example, the detailed information on deformable objects will aid in the training and evaluation of neural simulators [17, 34]. Lastly, the dataset must incorporate action information so agents can understand how their actions influence the 3d environments.

Existing datasets often fail to meet these criteria. 3D datasets like [9] typically use limited viewpoint cameras and struggle with non-diffuse materials. Motion datasets like [40, 42] lack detailed action information. Large-scale robotic manipulation datasets [19, 67] still face viewpoint limitations, highlighting the need for a comprehensive approach combining robotics and 3D vision. This gap is also evident in fields like dynamic NeRF [21, 72], which often rely on limited real or simulated scenes for training.

In this paper, we introduce Robo360, showcased in Figure 1, as the first real-world omnispective robotic manipulation dataset specifically designed to foster research in 3D robotic manipulation. This dataset comprises over $2,000$ multi-view robotic manipulation trajectories generated through teleoperation [53]. To capture these trajectories, we established a multi-view system with 86 cameras, each precisely calibrated and temporally aligned across different viewpoints. The extensive visual information in Robo360 facilitates the creation of high-quality 3D neural fields, leveraging recent progress in 3D neural representation learning. The diversity of Robo360 is another highlight; the dataset includes $100$ objects with over 20 different materials, encompassing a broad spectrum of optical and physical properties. This diversity supports studies in a wide array of object manipulation and physical world modeling challenges. Comparative analysis of Robo360 with previous real-world robotic datasets is presented in Tables 1.

In preliminary experiments with Robo360, we explore 3D neural scene representation learning and multi-view policy learning to validate the dataset. We systematically evaluate dynamic neural radiance field methods on our dataset and confirm the dataset's effectiveness in supporting imitation learning for robots to develop and generalize multi-view manipulation skills. Robo360 is expected to advance research in 3D scene representation learning, visual policy learning, and material modeling, enhancing the understanding of the physical world for robot control.

Our contributions are summarized as follows:

- We curate a unique dataset featuring dynamic scenes of robot manipulation, encompassing dense view coverage, diverse objects with distinct mechanical and deformability properties, various optical materials, per-frame high-quality 3D neural representations, and precise robot action information.
- We detail the technical challenges encountered in building a multi-view robotic manipulation data collection system and outline a robust pipeline to address these challenges.
- We conduct evaluations of existing 3D dynamic NeRF algorithms in real-world robot manipulation scenarios.
- A policy learning algorithm that adapts to multi-view inputs, showcasing its versatility across various views.

## 2. Related Work

**3D Dataset Captured in Real World** Capturing real-world objects and scenes is vital for computer vision research. Numerous works [3, 9, 12, 60, 62] focus on collecting large-scale scenes for indoor 3D scene understanding, using RGB and depth cameras like Kinect and iPhone LiDAR scanners. Similarly, [15, 61] employ RGB-D camera rigs to capture individual objects, resulting in influential datasets like YCB [7]. However, existing depth sensors struggle to capture accurate depth for highly specular or transparent objects. Unlike depth sensors, RGB cameras directly capture light and are unaffected by these issues. Some works [28, 64] use gantries for precise camera control during capture, but they have limited speed and are unsuitable for dynamic scenes. To capture dynamic scenes, complex view-dependent effects, and visual parameters, large arrays of RGB cameras and light stages have been employed. For instance, [41] uses a 34-camera array to capture intricate visual effects like fog, fluids, and fur, pioneering neural radiance capture. Meanwhile, [14, 39] concentrate on capturing objects under controlled lighting conditions for inverse rendering and material decomposition.

Additionally, [29, 42, 51] capture dynamic scene videos to advance dynamic scene representation in the field.

**Dynamic NeRF** The advancements in neural radiance fields (NeRF) [48] have significantly improved the quality of novel view synthesis, enabling the generation of realistic and detailed images from 3D scenes. One notable extension of NeRF focuses on addressing dynamic scenes, where objects or the scene itself undergo temporal changes. This has led to the development of dynamic NeRF methods. One stream of the dynamic NeRF works directly extends the radiance field with the time dimension or a latent code [18, 22, 35, 70]; some works utilize volume factorization to convert 4D volume into multiple planes or tensors [8, 21, 57], which provides a compact and efficient representation for dynamic scenes. Another direction of works [20, 49, 50, 52, 65] focuses on constructing an additional deformation field that maps point coordinates from different time frames into a canonical space, where large motion and geometry changes can be captured and learned. Recently, with the emergence of 3D Gaussian Splatting [30], which adopts a novel approach based on point cloud rendering, a series of works started using 3D Gaussians to model the dynamic scenes [43, 69, 72].

**3D Robot Learning** There is growing interest in robotics policies based on 3D vision [16, 24, 38, 75], and NeRFs derived from multi-view images have proven effective [16, 17, 34, 68, 75]. These neural fields serve as 3D representations, trained concurrently with policy learning by reconstructing multi-view observations. Then, policies are trained to condition on past observations, using generative models to output action sequences [11]. For example, imitation learning [75] learns policies from demonstrations, while reinforcement learning directly optimizes policies [16]. Neural renderers also enhance world model learning [17, 32, 56], allowing robots to plan using world models and forecast expected rewards.

**Dataset for Robotic Manipulation and Physics** Recent advancements in imitation learning highlight the importance of large-scale robot datasets for policy learning. These datasets are gathered through various methods such as kinesthetic teaching [58], composing primitive actions [6, 13], or teleoperation [19, 67]. While some datasets include multiple views [13, 19, 67], their limited viewpoints hinder comprehensive 3D scene understanding. In contrast, obtaining multi-view and 3D data is more straightforward in simulated environments [24, 27, 31, 44–47, 73], which facilitate testing robot learning and advancing physical world modeling, intuitive physics, and reasoning research [2, 4, 33]. Despite some progress in soft body simulation [26, 37, 71], if they can match the simulator with real-world dynamics remains challenging. A dataset encompassing various physical dynamics that can serve as a benchmark for evaluating these simulators becomes essential.

## 3. Robo360 Dataset

Robo360 is the first real-world multi-view dataset that simultaneously supports high-quality dynamic 3D scene representation and learning-based robot control. Robo360 aims to support research in 3D robotic manipulation, especially focusing on understanding the low-level dynamics in the manipulation process. To achieve this, we include multi-modality data in the Robo360 dataset. Specifically, Robo360 contains over 2000 demonstration trajectories of more than 100 different objects with diverse material variations. The multi-view trajectory RGB video and multi-directional audio are captured by 86 DSLR cameras, along with data with different modalities, including robot proprioception and robot control signal. This section highlights several notable features of Robo360.

**RGB Video Sequences** The robot execution trajectory video is captured with 86 DSLR cameras distributed across a half-dome. We calibrated the intrinsic and extrinsic parameters of all cameras. The visualization of the dome and calibrated cameras are in Figure 2 (A). All videos are captured at 30 FPS with 1080p resolution.

**Multi-Material** Our dataset, Robo360, captures diverse interactions between robots and objects with a wide range of visual and physical characteristics, as depicted in Figure 3. It includes over 200 trajectories for each object of rigid material, including various plastic objects, wood, and metals. These objects typically undergo rigid transformations; however, some of them, such as biscuits, can experience fractures during interaction. The dataset also encompasses soft materials, including rubber, cloth, paper, and malleable metals like copper and iron, with over 200 trajectories for each. Many of the soft objects in our dataset are composites, such as cardboard boxes, napkins, electronic cables, plastic bags, and ropes, combining these materials in varied configurations. Additionally, certain soft objects, such as fruits, snacks, and miscellaneous items, combine a distinctive set of materials that distinguishes them from other objects. Regarding liquids, the dataset encompasses a variety of types characterized by different colors, levels of transparency, densities, viscosities, and surface tensions. This includes substances such as water, fruit juice, dish soap, soda, and coffee for more than 30 trajectories in total.

**Robot Proprioception and Control** Robo360 captures high-frequency robot proprioceptions (joint position, joint velocity, joint acceleration, and joint torque) at 30 Hz, and

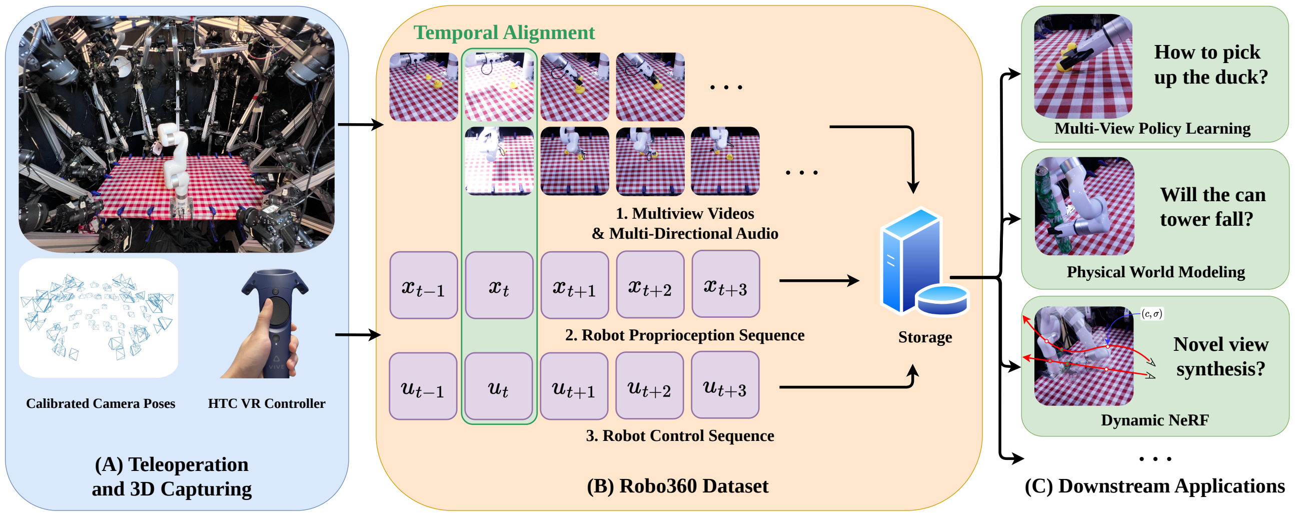

Temporal Alignment
1. Multiview Videos & Multi-Directional Audio
$x_{t-1}$ $x_t$ $x_{t+1}$ $x_{t+2}$ $x_{t+3}$
2. Robot Proprioception Sequence
$u_{t-1}$ $u_t$ $u_{t+1}$ $u_{t+2}$ $u_{t+3}$
3. Robot Control Sequence
Storage
Calibrated Camera Poses
HTC VR Controller
(A) Teleoperation and 3D Capturing
(B) Robo360 Dataset
How to pick up the duck?
Multi-View Policy Learning
Will the can tower fall?
Physical World Modeling
Novel view synthesis?
Dynamic NeRF
(C) Downstream Applications

Figure 2. An illustration of the comprehensive pipeline encompassing the stages of data collection, postprocessing, storage, and the subsequent downstream applications.

| Name | Robot | Control | # Views | Resolution | Camera Calib. | Audio | Depth | Cables | Cloth | Fluid |
|---|---|---|---|---|---|---|---|---|---|---|
| Deep3DMV [36] | ✗ | ✗ | 10 | **1080p** | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| DiVA360 [42] | ✗ | ✗ | 53 | 720p | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| RoboSet [5] | ✓ | 5 Hz | 4 | 480p | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| BridgeDataV2 [67] | ✓ | 5 Hz | 4 | 480p | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| RH20T [19] | ✓ | 10 Hz | 12 | 720p | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Robo360 **(Ours)** | ✓ | **30 Hz** | **86** | **1080p** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 1. Comparing our dataset with existing real-world multi-view video datasets.

teleoperation signals (target joint position, target gripper position) at 30 Hz. We also record the analytically solved end-effector pose and gripper tip position for the convenience of downstream applications.

**Multi-Modality** To support research in understanding the physical world, Robo360 consists of a rich set of different modalities, including 86-directional audio collected by DSLR cameras at 48000 Hz and depth map collected by 3 RealSense depth cameras.

## 4. RoboStage for Dataset Creation

We develop a multi-view system with 86 cameras to capture rich multi-modal data of the robot manipulation trajectories to support research in 3D robotic manipulation. Inspired by light stages in graphics [39, 41], we built our hardware and software system, which we call RoboStage. RoboStage can capture high-resolution multi-view videos and audio and synchronize them with the control signal of the robot arm. By leveraging a VR controller, we built a teleoperation system inside the RoboStage to collect robotic manipulation trajectories. Figure 2 shows the system and illustrates the dataset creation process, which enables collecting data that supports multiple downstream applications.

### 4.1. RoboStage Setup

RoboStage, as shown in Figure 2 (A), consists of 86 Canon DSLR 250D cameras (DSLR), and 3 RealSense depth cameras. The DSLR cameras are mounted on the upper half dome supported by a $2m \times 2m \times 2m$ metal frame, allowing RoboStage to capture videos from different directions. The system's specifics can be found in the supplementary.

### 4.2. Dynamic 3D Scene Capture

We carefully calibrate cameras to ensure precise parameters and align videos to ensure synchronized timesteps.

**Camera Calibration** Accurate camera parameters play a pivotal role in a multi-view dataset, particularly for tasks like 3D scene learning and novel-view synthesis. In our capturing system, where the positions and orientations of all cameras are fixed, we opt to perform calibration before the capturing process, mitigating the need for per-scene re-calibration. Specifically, we create a static scene with rich texture and low specularity. Subsequently, we employ COLMAP [55] to estimate camera parameters. To obtain

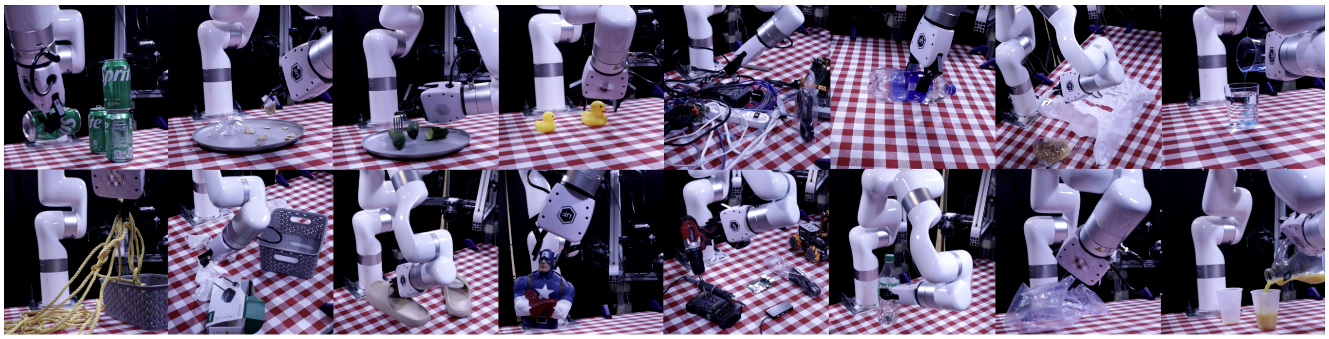


Figure 3. Robo360 captures real-world robot-object interactions with complex visual and material variations.

the metric scales and coordinate uniformity among camera poses, we utilize an ArUco board [23] to estimate the poses of a subset of the cameras and align the results from COLMAP and ArUco. We illustrated the calibrated camera poses in Figure 2 (A).

**Video Temporal Alignment** To support applications in the vision and robotics community, such as dynamic NeRF and imitation learning, we align the multi-view video, robot proprioception, and robot control sequences in the time dimension. To this end, we implemented a common event, i.e., a floodlight, that is easily detectable at the beginning of the video, serving as a keyframe to synchronize all videos. At the start of each trajectory, we activate a 30000-lumen floodlight and maintain its illumination for approximately 3 seconds. Subsequently, we calculate the average pixel value difference between each pair of consecutive frames. The frame exhibiting the maximum difference is identified as the keyframe. In addition to aligning the cameras, we align control signals, robot proprioception, and video frames using the RealSense cameras as a proxy. In detail, during each teleoperation step, we concurrently record the control signals and robot proprioception while capturing images using the RealSense cameras. This simultaneous recording aligns the control signals, robot proprioception, and RealSense videos automatically. Then, by applying the method described earlier to align the RealSense videos with the DSLR videos, we can achieve synchronization for the control signals and robot proprioception as well. We verified our alignment approach by comparing it with an iPad displaying a QR code video at 30 FPS in front of each camera and observing a maximum discrepancy of 1/30 seconds, while our method offers the advantage of greater automation as it is much cheaper and easier to activate.

## 4.3. Robot Arm

An xArm6 robot arm, equipped with 6 revolute joints and a parallel gripper, is mounted at the center of the capturing system. During data collection, the robot is teleoperated by a human operator, while during deployment, it can be controlled by a policy neural network. In this dataset, we use servo joint position as the low-level control signal for both teleoperation and policy deployment.

**Robot Teleoperation** The human teleoperator controls the robot using an HTC Vive VR gaming controller. During teleoperation, we use steam-openvr [1] to read the $3 \times 4$ target pose matrix and the trigger button status of the controller in real-time. Using inverse kinematics, we compute the closest target joint position to the current joint position that matches the end-effector pose to the target pose. We then compute the target gripper position by scaling the continuous-valued trigger position. Then, the robot arm and gripper are driven to the target joint position by making an API call to the xArm control box. The computed 6D target joint position and 1D target gripper position during teleoperation are recorded in the dataset, together with the real-time proprioception state information of the robot arm (joint position, joint velocity, joint acceleration, and joint torque).

**Robot Safety** We provide safety measures to prevent the robot from causing damage to other devices, such as cameras, desks, and single-board computers. To ensure the integrity of extended data-capturing and policy evaluation sessions, we have implemented a straightforward bounding box collision check during both teleoperation and policy deployment. Before directing the robot to a target joint position, we assess the positions of a predefined set of points on the surface of the robot links. These positions are then enclosed within a pre-defined bounding box. Subsequently, we calculate safe joint positions using inverse kinematics and execute these safe joint positions. This precautionary process mitigates the risk of collisions and safeguards against potential damage to surrounding devices.

**Policy Deployment** The trained visuomotor neural network policy predicts the delta joint position and target gripper position to control the robot, given real-time robot proprioception and image observation from a subset (4 in our experiments) of 86 DSLRs.

We find that our data collection system is quite efficient

| Method | D-NeRF | K-Planes | De. 3DGS | TensoRF | 3DGS |
|---|---|---|---|---|---|
| Per-frame | ✗ | ✗ | ✗ | ✓ | ✓ |
| **Bottle Flip** | 10.96 | 16.94 | **18.32** | 19.42 | **24.84** |
| **Bread Fall** | 10.75 | 17.32 | **19.05** | 20.36 | **24.90** |
| **Cucumber Skewer** | 11.02 | 17.21 | **18.87** | 20.05 | **25.36** |
| **Dish Soap** | 10.82 | 16.91 | **19.05** | 21.05 | **24.90** |
| **Tshirt Fold** | 11.27 | 16.89 | **18.52** | 20.36 | **24.07** |
| **Cookie Crushed** | 11.35 | 17.10 | **18.75** | 20.89 | **25.48** |
| **Rope Pick-and-Place** | 10.39 | 16.85 | **18.65** | 20.47 | **24.70** |
| **Air-bag Squeeze** | 10.46 | 17.31 | **17.94** | 20.55 | **24.06** |
| **power strip** | 10.40 | 16.74 | **18.09** | 20.08 | **23.90** |
| **Juice Pour** | 11.13 | 17.08 | **18.59** | 20.60 | **24.20** |
| **Bread Put-in-bag** | 9.94 | 17.07 | **17.78** | 19.79 | **23.94** |
| **Sand Pour** | 10.88 | 17.26 | **18.97** | 20.65 | **25.78** |
| **Sprite Pour** | 10.21 | 17.14 | **18.91** | 19.79 | **25.64** |
| **Sprite Stack** | 10.09 | 17.12 | **18.83** | 20.46 | **24.74** |
| **Tshirt Unfold** | 10.91 | 17.15 | **18.03** | 20.01 | **24.18** |

Table 2. Dynamic NeRF Evaluation: Average PSNR across a 5-second video.

and easy to learn. It took, on average, 2 minutes to collect a 30-seconds manipulation trajectory, including scene setup, temporal alignment, object manipulation, and data transfer. Moreover, our system enables a higher control frequency than other datasets in Table 1, supporting more complicated manipulation skills. We leave more details in the supplementary.

## 5. Baseline Experiments

In this section, we verify our dataset in two tasks: novel-view synthesis of dynamic scenes and policy imitation learning. We split our dataset into two categories for these two tasks. For novel-view synthesis of dynamic scenes, we use videos focusing on collecting a diverse set of objects with visual and material variation, while for policy learning, we use videos focusing on a small set of objects with each 50-200 trajectories. We found that from Robo360 we can obtain high-quality neural radiance fields, and the collected demonstrations can efficiently drive robot control policy learning, proving the effectiveness of our dataset. We systematically evaluated the existing 3D dynamic NeRF methods in our dataset and found that existing dynamic scene representation learning methods are insufficient to model objects with fast motion, resulting in a performance gap compared to static scenes.

### 5.1. Dynamic NeRF

We selected 15 different scenes to compare various dynamic NeRF methods. These scenes cover a wide range of scenarios involving the robot's interactions with objects that possess diverse materials, both in terms of their visual and physical properties, We crop the manipulation process into a 5-second 30FPS video, yielding a total of 150 frames per scene. For evaluating the models' ability in novel-view synthesis, we hold 5 of 86 views as the test set.

| Task | Test | BC(F) | BC(A) | DP(F) | DP(A) |
|---|---|---|---|---|---|
| **Towel** | Train | 0.60 | 0.87 | **1.00** | **1.00** |
| | 1-Test | 0.26 | **0.80** | 0.40 | **0.80** |
| | 2-Test | 0.00 | 0.73 | 0.00 | **0.80** |
| | 3-Test | 0.00 | 0.67 | 0.00 | **0.73** |
| **Slippers** | Train | 0.20 | 0.00 | **0.53** | **0.53** |
| | 1-Test | 0.07 | 0.00 | 0.00 | **0.60** |
| | 2-Test | 0.00 | 0.00 | 0.00 | **0.53** |
| | 3-Test | 0.00 | 0.00 | 0.00 | 0.00 |
| **Bottle** | Train | 0.27 | 0.80 | **1.00** | 0.87 |
| | 1-Test | 0.00 | 0.33 | 0.20 | **0.87** |
| | 2-Test | 0.00 | 0.33 | 0.00 | **0.80** |
| | 3-Test | 0.00 | 0.13 | 0.00 | **0.80** |
| **Cable** | Train | 0.47 | 0.60 | **0.87** | 0.67 |
| | 1-Test | 0.00 | **0.60** | 0.00 | **0.60** |
| | 2-Test | 0.00 | 0.27 | 0.00 | **0.47** |
| | 3-Test | 0.00 | 0.00 | 0.00 | 0.00 |
| **Rope** | Train | 0.67 | **1.00** | **1.00** | **1.00** |
| | 1-Test | 0.00 | 0.60 | 0.47 | **0.80** |
| | 2-Test | 0.00 | 0.00 | 0.00 | **0.67** |
| | 3-Test | 0.00 | 0.00 | 0.00 | 0.00 |

Table 3. Imitation learning for robot control baseline methods task success rate. We report the average success rate over 15 trials. BC and DP denote Behavior Cloning and Diffusion Policy, respectively. (F) denotes the method is trained on a fixed 4 views. (A) denotes the method is trained on randomly sampled 4 views. We test the methods in different 4-view combinations, where "Train" means the 4 training views are used as input during deployment. "N-Test" means we use N novel views and 4-N training views as input during deployment.

**Baselines** We validate five different dynamic NeRF methods: D-NeRF[52], K-Planes[21], Deformable 3DGS[72], per-frame TensoRF[10], and per-frame 3DGS[30]. We report the PSNR results of these methods on novel views in Table 2 and provide qualitative results in Figure 4.

D-NeRF employs a Multilayer Perceptron (MLP) to model the dynamic scene by directly extending the radiance field with the time dimension as an input. Thus, its performance is constrained by the network capacity, particularly in scenarios with a large number of frames. Despite being trained for 10 hours per scene on our dataset, D-NeRF can only capture a fuzzy representation of scene composition, lacking detailed information.

K-Planes utilizes six planes to decompose spatial-temporal volumes. In our dataset, K-Planes exhibits strong performance in areas visible from most views but struggles in regions visible from only a small subset of cameras. Since our cameras are predominantly oriented toward the center of the light stage, where most motion occurs, K-Planes accurately depicts the approximate motion of the object in these regions. However, the novel-view renderings lack temporal smoothness due to the limited representation ability.

Deformable 3DGS adds a deformation field to Gaussian

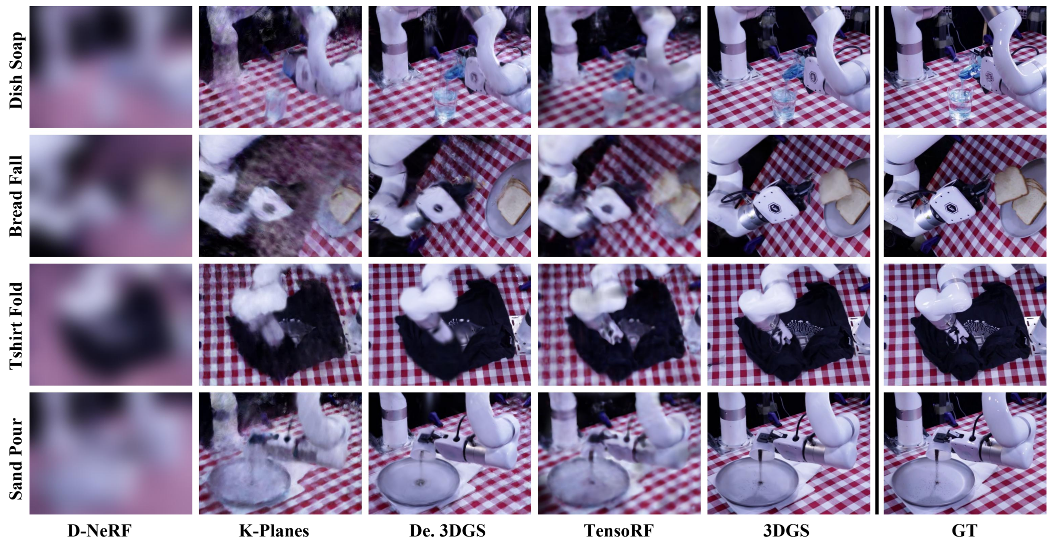


Figure 4. Qualitative results of dynamic NeRF methods. All the dynamic NeRF methods fail to accurately model objects with fast motion, such as falling bread and pouring sand, leading to a performance gap compared to static scenes.

Splatting to enable dynamic scene novel-view synthesis. While it represents the state-of-the-art method in dynamic NeRF, it still exhibits several limitations when applied to our dataset. First, the 5-second video contains too much information and is challenging for the capacity of Deformable 3DGS. Second, we also observe that the GS-based methods fail and produce a worse result on the specular surfaces.

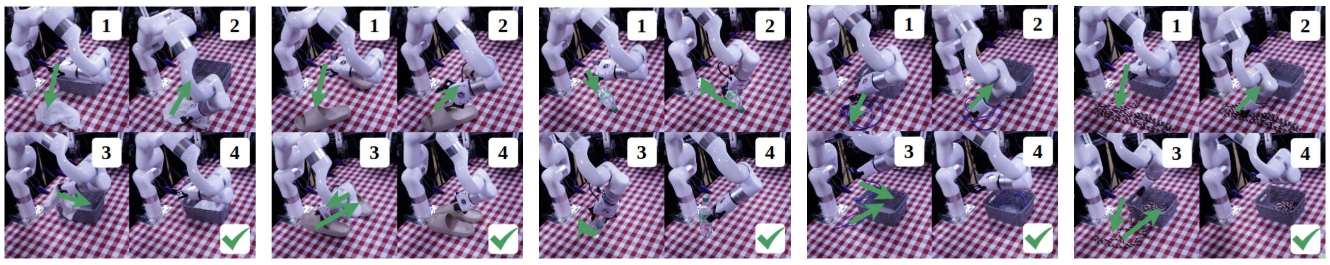
In addition to the dynamic NeRF methods, we also validate TensoRF and Gaussian Splatting for novel-view synthesis methods in static scenes by training and testing one model frame by frame. We restrict the model training on each frame for a maximum duration of 5 minutes to avoid exceeding the computation budgets. We notice TensoRF focuses on modeling the whole scene and achieves a relatively high PSNR but lacks details on the manipulated object. For Gaussian Splatting, we initialized the Gaussian using COLMAP and trained the model for 10000 iterations. We noticed an increase in the degree of spherical harmonic function boosted the performance on the specular region of the robot surface. Finally, per-frame Gaussian Splatting delivers the highest PSNR in our dataset in all scenes.

We also conducted an ablation study to compare their performances under different total lengths of frames. Please refer to the supplementary material for more results.

## 5.2. Learning-Based Robot Control

We now verify if our dataset can be used to learn robot control policies with imitation learning and if our dataset can help learned policies generalize to novel views. We selected five representative soft body manipulation tasks involving distinct materials: cloth, rubber, soft metal, plastic, and liquid. The specifics of these tasks are illustrated in Figure 5. Our analysis used a targeted subset of our dataset, comprising 30-200 demonstrations for each object. Specifically, we gathered 80 demos for Towel, 30 for Slippers, 80 for Bottle, 150 for Cable, and 155 for Rope to train the policy networks. We design the policy network to receive and process two sequential observations of the task, each from four different angles, presented in $128 \times 128$ RGB images. It outputs a delta joint position and a gripper position to control the robot.

**Baseline Methods and Evaluation Metric** We experimented with 2 imitation learning methods: Behavior Cloning (BC) [63] and Diffusion Policy (DP) [11]. BC learns the control policy by minimizing squared L2 loss between predicted action and ground truth action. Diffusion Policy learns a denoising network by predicting the noise from the action diffusion process. For each method, we train 2 different variants. The fixed-view network is denoted with suffix **(F)** and the arbitrary view network is denoted with suffix **(A)**. We verify the correctness of our dataset by training and evaluating the fixed-view network and then understand how our dataset helps with learning a policy that can generalize to novel view directions with the arbitrary view network. The fixed-view network computes the multi-view image feature with a convolutional neural network (CNN) with input channel 24 (4 RGB views, 2 consecutive frames stacked). The arbitrary view network computes the multi-view image feature with a CNN image tokenizer with input channel 6 (1 RGB image, 2 consecutive frames stacked) and a transformer encoder [66] that is used to process these tokens. For BC, the image features are processed by an MLP to output the action at the current timestep. For DP, the image features are processed by a transformer decoder to compute the denoising step of a sequence of actions.



(a) Towel (b) Slippers (c) Bottle (d) Cable (e) Rope

Figure 5. Visualization of 5 example demos used in training networks via imitation learning. For each demo, we show 4 keyframes, where green arrows show the desired end-effector motion and green checkmarks denote the task is completed at the current time. Zoom in for details.

**Experiment Results** We evaluate the performance of baseline methods in Table 3. We measure the success rate of each method-network pair in 4 view combinations (Train, 1-Test, 2-Test, and 3-Test) as mentioned in the table. We first observe that, given in-distribution training views, both versions of DP outperform BC in most cases. In this setting, the fixed-view version performs similarly or better than the arbitrary-view version. However, in novel view control setting, for both BC and DP, once even 1 camera position is perturbed to a novel viewing direction, we observe a severe performance drop for the fixed-view policies, whereas the policy trained on arbitrary view data only drops slightly or maintains similar performance. This is because when a view is perturbed, the input is out-of-distribution (OOD) for fixed-view policies, while the arbitrary view policy can interpolate the novel view input between the training view data that is close to the perturbed view direction to generalize to these unseen view directions and maintain performance. In the novel view control setting, we also observe that arbitrary view DP is more robust to view perturbation than BC. This is likely because the diffusion denoising process itself can help with combating OOD. However, this is not sufficient for a fixed-view DP to generalize to the novel view directions, since in this case, the novel view image observation that the diffusion process is conditioned on is itself OOD to the network. For arbitrary view policies, the performance continues to drop in most cases when more views are perturbed. We want to emphasize that building robust view independent policies is important for many applications and remains an open problem.

The amount of data available to train a network is crucial to the performance of a policy in all tasks. We observe all methods perform the worst in the task Slippers, comparing to their performance in other tasks. The performance of arbitrary view BC is impacted the most as it completely fails in all trials of all view combinations. This is very likely since we only provided 30 demonstrations for this task, compared to other tasks. This shows the importance of collecting more diverse data for learning-based robot controllers.

One important contributing factor to the performance of all trained policies is that the inference frequency needs to match the training frequency to achieve reasonable performance. We conjecture that it is because a significant delay would change the distribution of the observation, leading to a performance drop. We find that the inference time of BC is much shorter than DP, which does not require multi-step control sequence prediction to match the training data distribution. On the other hand, DP relies on multi-step prediction since the denoising step is relatively more compute-intensive. We find slightly reducing the number of transformer decoder layers of the denoising network benefits performance by providing a higher inference frequency.

## 6. Conclusion and Limitation

We introduce Robo360, a 3D omnispective multi-material robotic manipulation dataset. It comprises a diverse range of robot manipulation trajectories collected by teleoperation, encompassing various physical and optical properties, as well as manipulation skills. To facilitate data collection, we have constructed RoboStage, a multi-view system equipped with 86 cameras, enabling the collection of multi-modal observations with rich 3d information. However, certain limitations are inherent in our approach. Firstly, our heavy reliance on visual data limits our dataset's ability to discern the internal structure of objects, and it is susceptible to significant occlusion. Nonetheless, we believe this limitation aligns with the current state of vision-based robot control and is not necessarily a drawback. Additionally, it is worth noting that the setup of RoboStage entails non-negligible costs, which may impede its scalability. Although we anticipate expanding our data collection efforts in the future, an exciting avenue for further exploration lies in developing techniques aimed at learning 3D scenes more effectively using fewer views from our dataset.

## References

[1] Steam openvr, Welcome to Steam, 2020. 5

[2] Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre: A new benchmark for physical reasoning. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[3] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. 2021. 2

[4] Daniel M Bear, Elias Wang, Damian Mrowca, Felix J Binder, Hsiao-Yu Fish Tung, RT Pramod, Cameron Holdaway, Sirui Tao, Kevin Smith, Fan-Yun Sun, et al. Physion: Evaluating physical prediction from vision in humans and machines. *arXiv preprint arXiv:2106.08261*, 2021. 3

[5] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking, 2023. 4

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 3

[7] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *IEEE Robotics & Automation Magazine, 22 (2015) 36 - 52*, 22(3):36–52, 2015. 2

[8] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 3

[9] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. 2017. 2

[10] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 6

[11] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023. 3, 7

[12] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017. 2

[13] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In *CoRL 2019: Volume 100 Proceedings of Machine Learning Research*, 2019. 3

[14] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face, 2000. 2

[15] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. 2022. 2

[16] 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. 1, 3

[17] Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. Learning multi-object dynamics with compositional neural radiance fields. In *Conference on Robot Learning*, pages 1755–1768. PMLR, 2023. 1, 2, 3

[18] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14304–14314. IEEE Computer Society, 2021. 3

[19] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A robotic dataset for learning diverse skills in one-shot. *arXiv preprint arXiv:2307.00595*, 2023. 2, 3, 4

[20] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 3

[21] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 2, 3, 6

[22] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 3

[23] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. 5

[24] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023. 1, 3

[25] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. 1

[26] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021. 3

[27] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark &

learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020. 3

[28] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014. 2

[29] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015. 3

[30] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. 3, 6

[31] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023. 1, 3

[32] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512*, 2023. 1, 3

[33] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018. 1, 3

[34] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Proceedings of the 5th Conference on Robot Learning*, pages 112–123. PMLR, 2022. 2, 3

[35] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 1, 3

[36] Kai-En Lin, Lei Xiao, Feng Liu, Guowei Yang, and Ravi Ramamoorthi. Deep 3d mask volume for view synthesis of dynamic scenes. In *ICCV*, 2021. 4

[37] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 432–448. PMLR, 2021. 3

[38] Zhan Ling, Yunchao Yao, Xuanlin Li, and Hao Su. On the efficacy of 3d point cloud reinforcement learning. *arXiv preprint arXiv:2306.06799*, 2023. 1, 3

[39] Isabella Liu, Linghao Chen, Ziyang Fu, Liwen Wu, Haian Jin, Zhong Li, Chin Ming Ryan Wong, Yi Xu, Ravi Ramamoorthi, Zexiang Xu, and Hao Su. Openillumination: A multi-illumination dataset for inverse rendering evaluation on real objects. 2023. 2, 4

[40] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21013–21022, 2022. 2

[41] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (SIGGRAPH 2019) 38, 4, Article 65*, 38(4):1–14, 2019. 2, 4

[42] Cheng-You Lu, Peisen Zhou, Angela Xing, Chandradeep Pokhariya, Arnab Dey, Ishaan Shah, Rugved Mavidipalli, Dylan Hu, Andrew Comport, Kefan Chen, et al. Diva-360: The dynamic visuo-audio dataset for immersive neural fields. *arXiv preprint arXiv:2307.16897*, 2023. 2, 3, 4

[43] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 3

[44] Zhao Mandi, Fangchen Liu, Kimin Lee, and Pieter Abbeel. Towards more generalizable one-shot visual imitation learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2434–2444. IEEE, 2022. 3

[45] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[46] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[47] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3): 7327–7334, 2022. 3

[48] 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. 3

[49] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 3

[50] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 3

[51] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceed-*

*ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9054–9063, 2021. 3

[52] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 3, 6

[53] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dietor Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023. 2

[54] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020. 1

[55] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 501–518. Springer, 2016. 4

[56] Younggyo Seo, Junsu Kim, Stephen James, Kimin Lee, Jinwoo Shin, and Pieter Abbeel. Multi-view masked world models for visual robotic manipulation. *arXiv preprint arXiv:2302.02408*, 2023. 3

[57] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16632–16642, 2023. 3

[58] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018. 3

[59] Dongseok Shim, Seungjae Lee, and H Jin Kim. Snerl: Semantic-aware neural radiance fields for reinforcement learning. *arXiv preprint arXiv:2301.11520*, 2023. 1

[60] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images, 2012. 2

[61] Arjun Singh, James Sha, Karthik S. Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances, 2014. 2

[62] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite, 2015. 2

[63] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018. 7

[64] Marco Toschi, Riccardo De Matteo, Riccardo Spezialetti, Daniele De Gregorio, Luigi Di Stefano, and Samuele Salti. Relight my nerf: A dataset for novel view synthesis and relighting of real world objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20762–20772, 2023. 2

[65] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021. 3

[66] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 7

[67] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023. 2, 3, 4

[68] Yixuan Wang, Zhuoran Li, Mingtong Zhang, Katherine Driggs-Campbell, Jiajun Wu, Li Fei-Fei, and Yunzhu Li. D3fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation. *arXiv preprint arXiv:2309.16118*, 2023. 3

[69] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 3

[70] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021. 3

[71] Zhou Xian, Bo Zhu, Zhenjia Xu, Hsiao-Yu Tung, Antonio Torralba, Katerina Fragkiadaki, and Chuang Gan. Fluidlab: A differentiable environment for benchmarking complex fluid manipulation. *arXiv preprint arXiv:2303.02346*, 2023. 3

[72] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 2, 3, 6

[73] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020. 3

[74] Yanjie Ze, Nicklas Hansen, Yinbo Chen, Mohit Jain, and Xiaolong Wang. Visual reinforcement learning with self-supervised 3d representations. *IEEE Robotics and Automation Letters*, 8(5):2890–2897, 2023. 1

[75] Yanjie Ze, Ge Yan, Yueh-Hua Wu, Annabella Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. Gnfactor: Multi-task real robot learning with generalizable neural feature fields. *arXiv preprint arXiv:2308.16891*, 2023. 1, 3

# Robo360: A 3D Omnispective Multi-Material Robotic Manipulation Dataset

Supplementary Material

## 7. System Specifics

The 86 Canon 250D cameras record video at 30 FPS, 1080p resolution, 1/200 shutter speed, F8.0 aperture, ISO 3200, and Fluorescent white balance. This configuration is selected to balance image quality with motion blur.

The 3 RealSense D435 cameras record pixel-aligned color and depth video at 30 FPS at 480p, which is the recommended resolution of D435 for depth image quality.

The 86 Canon cameras are controlled by 23 Raspberry Pi 4B single-board computers for video recording and downloading. The Raspberry Pis and the main PC are connected via LAN using an internet switch. These devices with the main PC communicate within 192.168.0.x.

The main PC controls the xArm6 robot arm and gripper via direct ethernet connection on 192.168.1.x. The main PC reads the target end-effector pose and gripper position from the VR game controller via USB, computes the target joint position, and drives the robot arm by sending a packet to the xArm control box via ethernet using the xArm factory-provided API. This control loop runs at 30 Hz.

We use 4 Daylight Artist Studio Lamps for illumination, and 1 TYCOLIT Outdoor Led Flood Light for temporal alignment.

The desk is 152.4 cm by 76.2 cm.

## 8. Temporal Alignment Error Analysis

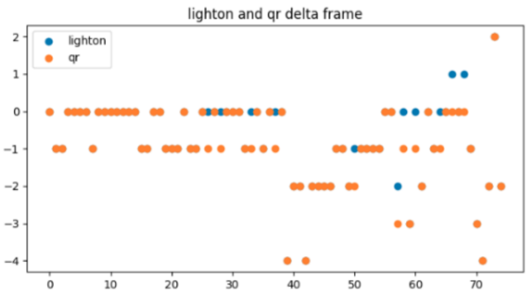


Figure 6. Temporal alignment error

The sudden increase in average pixel intensity is used to identify the keyframe to align the multi-view videos. This is shown in Figure 11. To verify the alignment quality of this method, we record a multi-view video of 1. light on and off, 2. a phone playing a 30-second 30 fps QR code video of 0-899 held around the stage to ensure all videos contain a section where the QR code is visible, shown in Figure 12. The starting frame is found by subtracting the QR code number from the frame index of the image. To analyze the error between the 2 methods, we recorded a video with both alignment events and found the starting frame of the videos using these 2 methods. Then, we assumed the two methods agreed with each other on the video recorded from the 0th camera. After that, we plotted the temporal shift (unit: frames) between 0th and every other camera using the 2 methods, shown in Figure 6, where the x-axis is the camera ID and the y-axis is the relative temporal shift (unit: frames) of this video compared to the 0th camera, estimated using flood light (blue) and QR code (orange). We found out two methods have a maximum alignment discrepancy of 1 frame in 11 out of 86 videos. We believe this error is almost negligible for any downstream applications.

The reason QR code video is not used as the alignment event is that it is inefficient in data collection and post-processing. During data collection, we empirically found that showing the QR code video to all cameras takes 30 seconds, whereas light on and off only takes 4-5 seconds max. During post-processing, the opencv QR code reading algorithm is much slower than computing the average pixel intensity of a frame, which is simply a sum and multiply operation on an integer array.

## 9. Control Frequency Requirements

The control frequency of teleoperation is crucial to enable the teleoperator to collect more diverse tasks. In our dataset, many involve the robot arm manipulating an open container full of liquid and pouring it into another container. In a task, we tasked the teleoperator to pour a cup with water filled to 3/4 to another cup. We empirically find out the well-trained teleoperator starts to spill the liquid uncontrollably when the control frequency drops below 20. When the control frequency drops below 10, the teleoperator loses control of the cup and spills almost all the liquid in the cup. There are many tasks that require high-frequency feedback control, such as liquid container manipulation, precise insertion of objects, balancing objects in hand, etc.

## 10. Dynamic NeRF experiments

### 10.1. More visualization results

Here we show more qualitative results of the visualization of dynamic NeRF baselines in Figure 7.

### 10.2. Video length

To evaluate the robustness and scalability of these baseline methods, we conducted an ablation study to investigate the influence of different video lengths on the performance met-

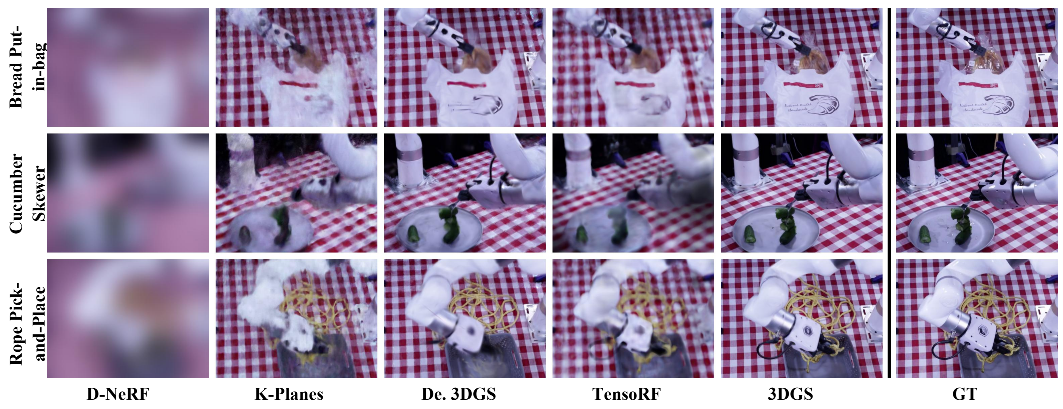


Figure 7. Visualization of more dynamic NeRF results using 5 different baselines.

| Method | 2 seconds | 5 seconds | 15 seconds |
|---|---|---|---|
| D-NeRF | 10.52 | 10.58 | 10.34 |
| K-Planes | 16.86 | 16.99 | 15.84 |
| De. 3DGS | 18.42 | 18.57 | 17.81 |
| TensoRF | 20.21 | 20.14 | 20.13 |
| 3DGS | 24.82 | 24.76 | 24.71 |

Table 4. Ablation study of different video lengths: Average PSNR over all scenes.

rics. Specifically, we varied the length of the input video while keeping other parameters constant. The tested video lengths included short clips (2 seconds), medium-length sequences (5 seconds), and longer videos (15 seconds). Table 4 presents the quantitative results of our ablation study for different video lengths using PSNR as the metric. The table shows that all three dynamic NeRF methods achieve the best performance on 5s-length videos, so we chose this length to evaluate the results of various methods.

### 10.3. Training time

We also conducted ablation studies on different training time to understand the performance of different training baselines under a limited training budget. The default training time of all 3 different dynamic NeRF methods is about 10 hours. Thus, we conducted experiments with three different training durations: 10 minutes, 1 hour, and the default 10 hours. The other parameters are kept the same, such as 5s-length video, 81 views for training and 5 views for testing, etc. The result is shown in Table 5.

| Method | 10 minutes | 1 hour | 10 hours |
|---|---|---|---|
| D-NeRF | 8.04 | 8.83 | 10.58 |
| K-Planes | 12.35 | 15.77 | 16.99 |
| De. 3DGS | 11.61 | 16.39 | 18.57 |

Table 5. Ablation study of different training times: Average PSNR over all scenes.

## 11. Imitation Learning

### 11.1. Task details

**Towel** The trained policy controls xArm6 to grasp the towel and put into the basket. The dataset contains 80 demos with each 300 steps at 30 Hz. Execution is considered a success when the towel is fully inside the basket.

**Slippers** The trained policy controls xArm6 to move the slippers together to organize them. The dataset contains 30 demos with each 450 steps at 30 Hz. Execution is considered a success when both slippers are closer to the center than before.

**Bottle** The trained policy controls xArm6 to flip the bottle up right. The dataset contains 200 demos with each 450 steps at 30 Hz. Execution is considered a success when the bottle is in an up right position without the robot holding it.

**Cable** The trained policy controls xArm6 to put a blue USB extension cable into the basket. The dataset contains 150 demos with each 450 steps at 30 Hz. Execution is considered a success when the USB cable is fully inside the basket.

**Rope** The trained policy controls xArm6 to put a red and white rope into the basket. The dataset contains 155 demos with each 750 steps at 30 Hz. Execution is considered a success when the rope is fully inside the basket.

### 11.2. More Visualizations

Please refer to the supplementary material video for more policy deployment visualizations.

## 12. Open Source

All data and code will be available at https://robo360dataset.github.io/.

## 13. More Example Data

86 views of video data of task orange juice (Figure 8), fold t-shirt (Figure 9) and sprite can tower collapse (Figure 10) are shown in the following pages. The full video can be found in the supplementary material video.

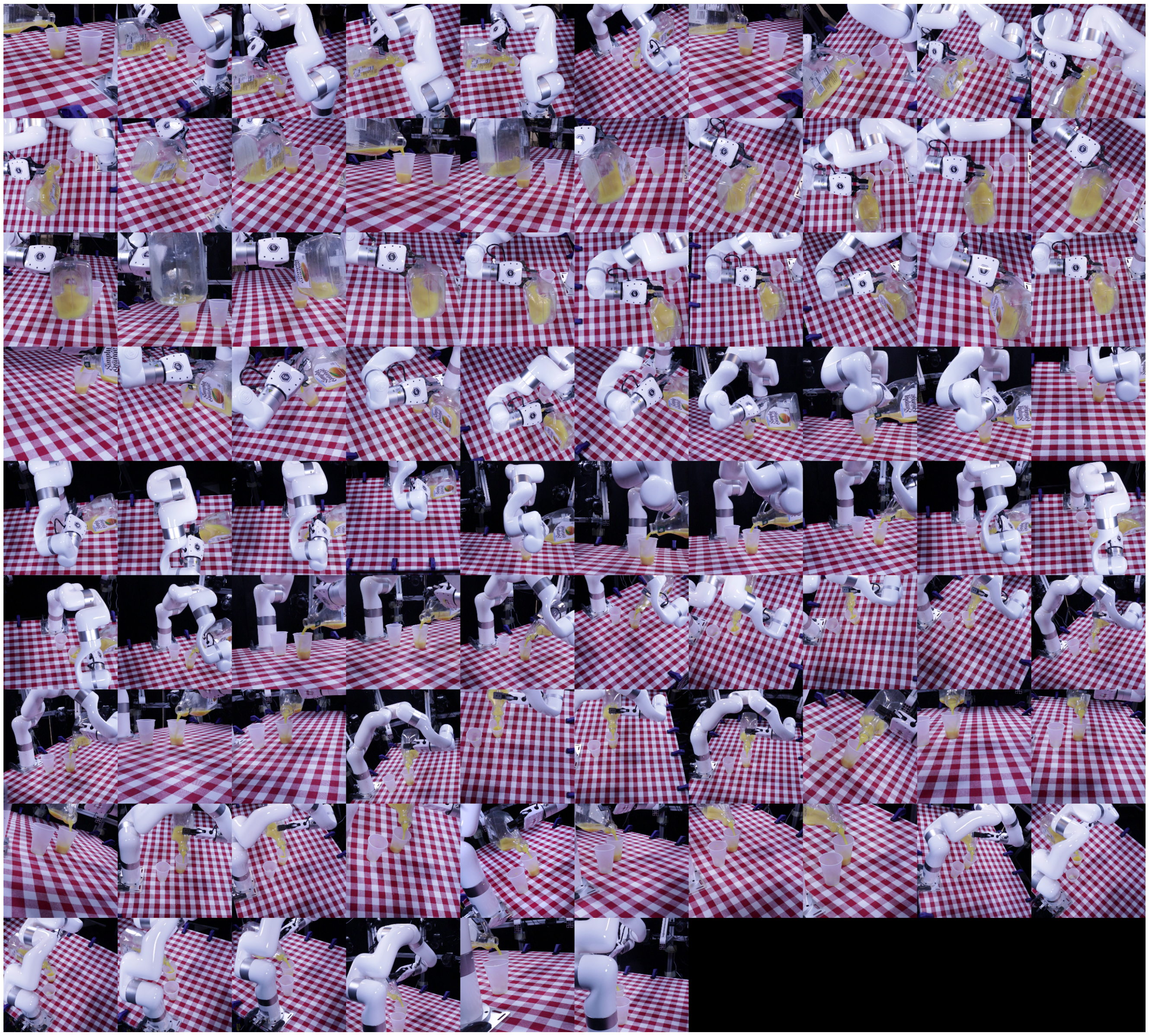


Figure 8. Video of xArm6 robot pouring orange juice into a plastic cup from all 86 views.

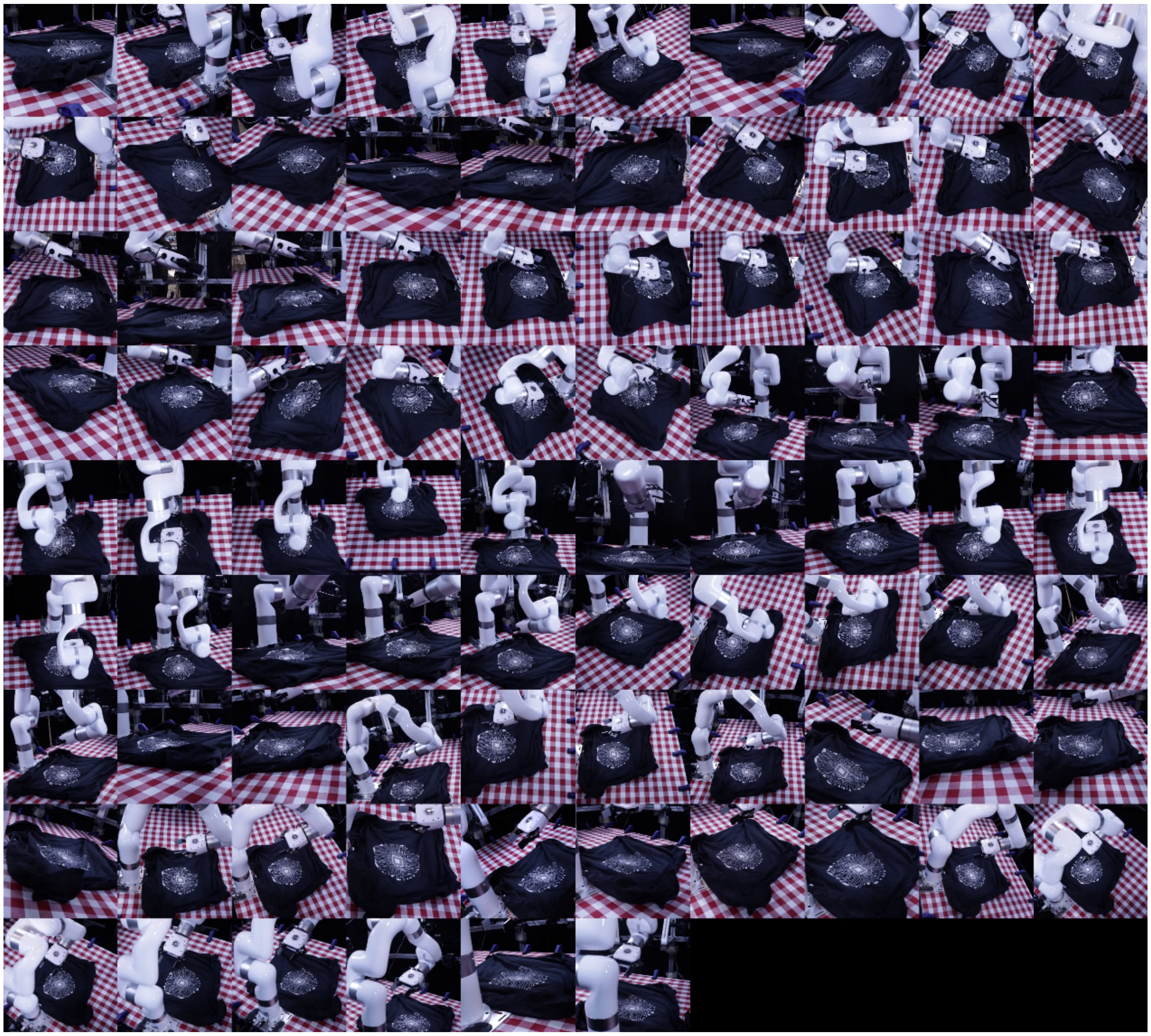


Figure 9. Video of xArm6 robot folding a t-shirt with an electric brain logo from all 86 views.

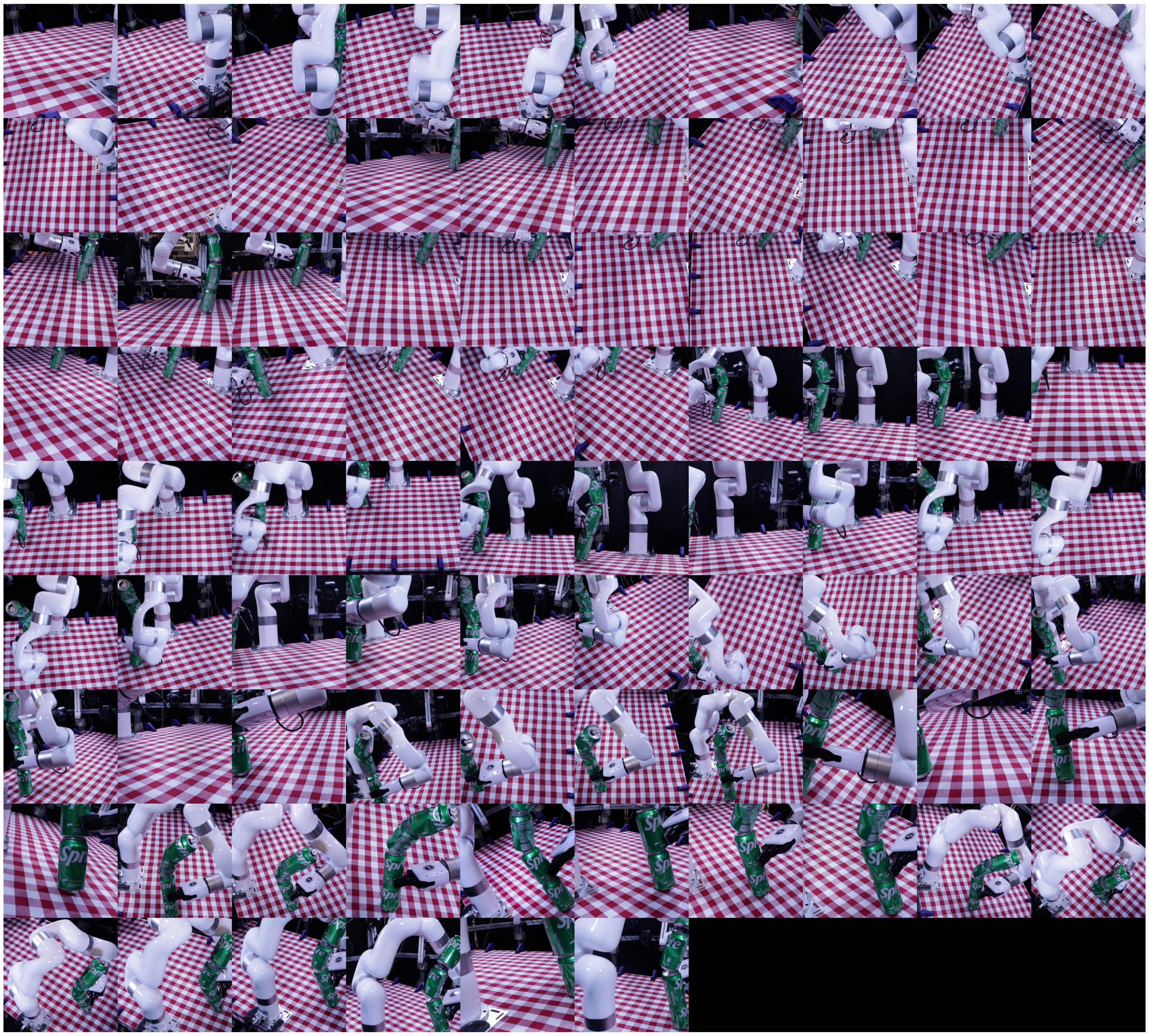


Figure 10. Video of xArm6 hitting a sprite can tower from all 86 views.

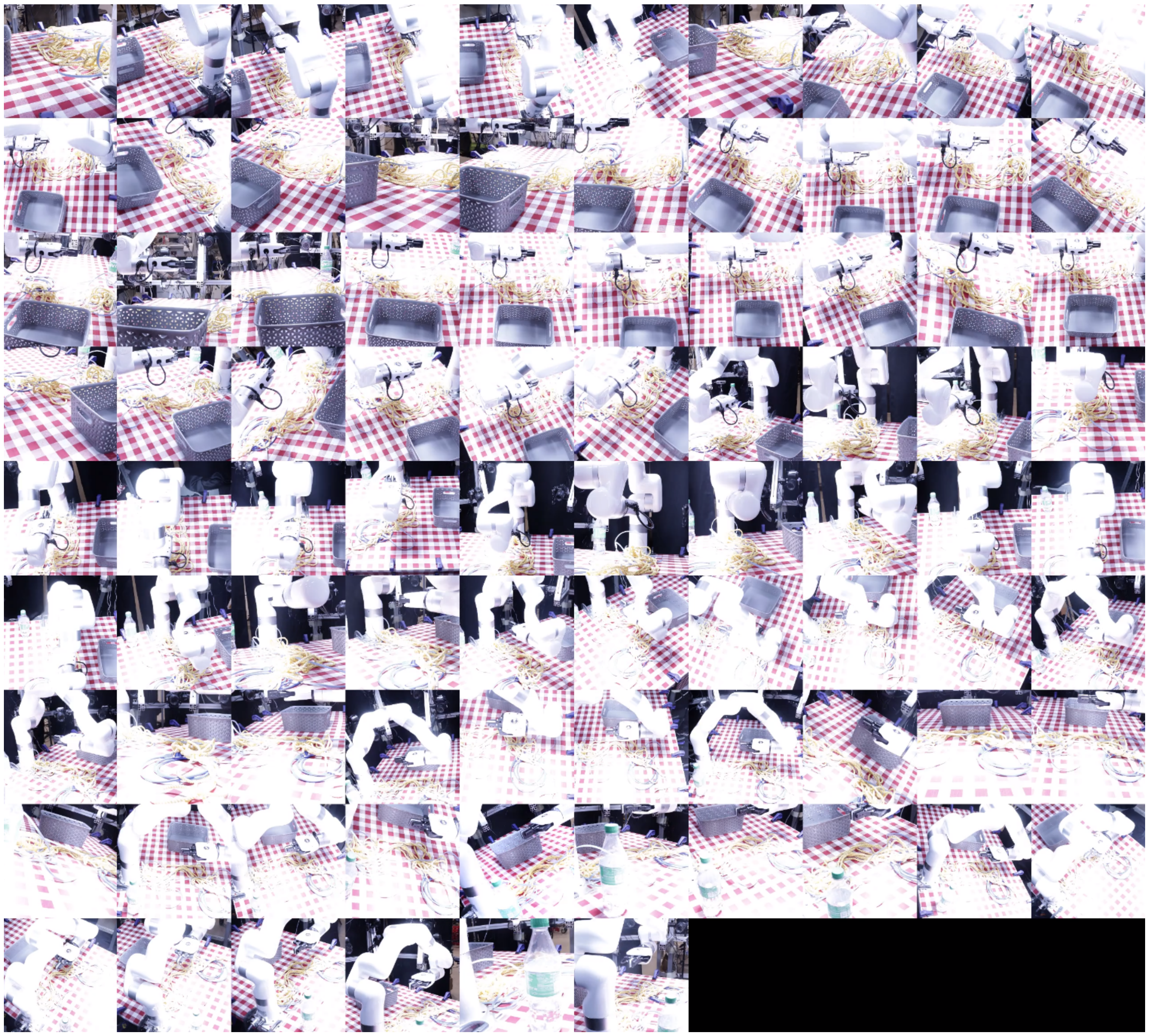


Figure 11. Example light on alignment event captured by all 86 cameras

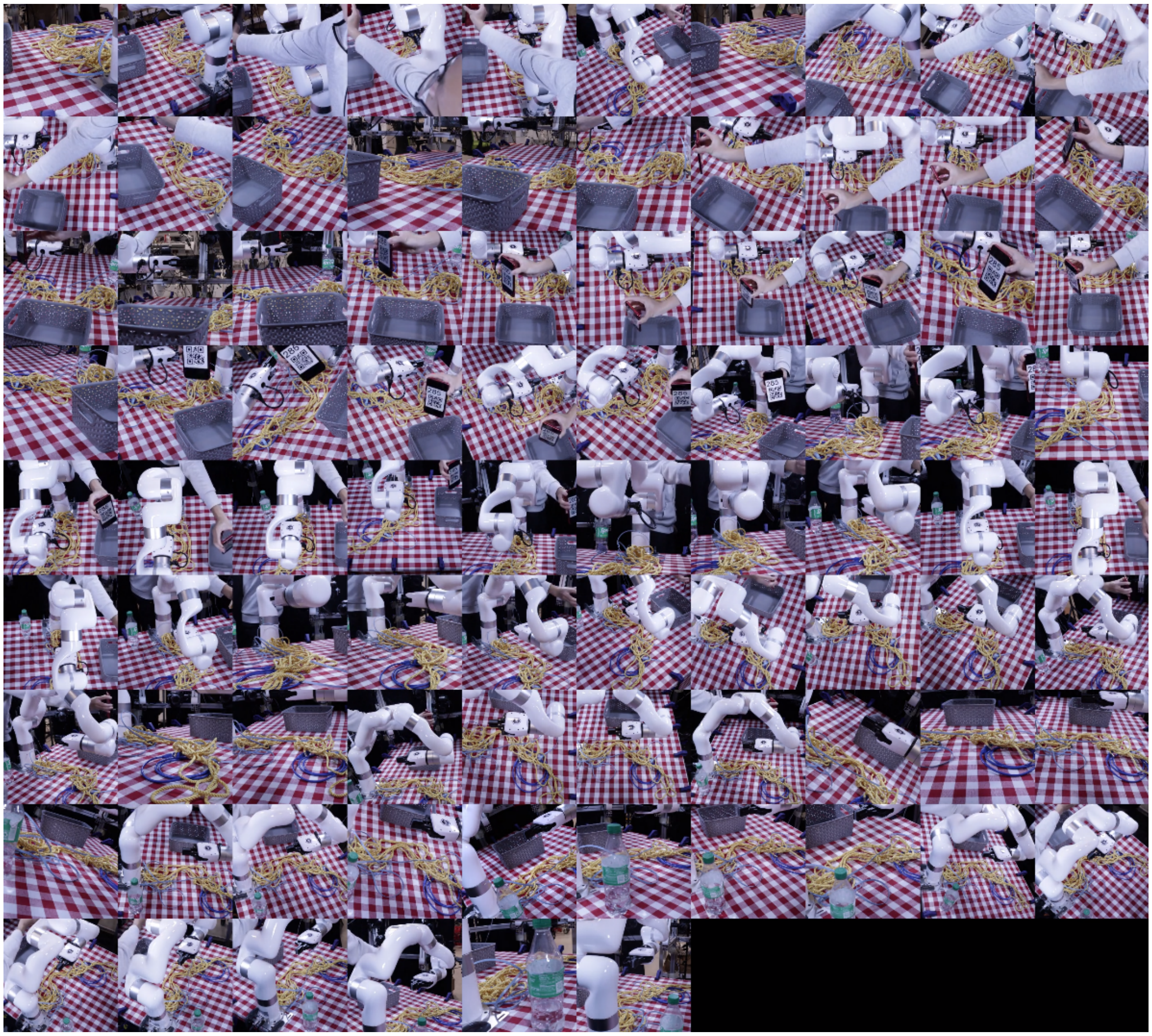


Figure 12. Example QR code alignment event captured by 86 cameras over 30 seconds. The teleoperator holds a phone playing a 30 fps QR code video around the inside of RoboStage to make sure the video recorded from all views contains a portion of the footage that records the QR code at some point during the entire 30 seconds.