Going Beyond Images: 3D-Aware Representation Learning for Visuomotor Control

Anonymous Author(s)
Affiliation
Address
email

Abstract:
Humans have a strong intuitive understanding of the 3D environment around us. The mental model of the physics in our brain applies to objects of different materials and enables us to perform a wide range of manipulation tasks that are far beyond the reach of current robots. In this work, we desire to learn models for dynamic 3D scenes purely from 2D visual observations. Our model combines Neural Radiance Fields (NeRF) and time contrastive learning with an autoencoding framework, which learns viewpoint-invariant 3D-aware scene representations. We show that a dynamics model, constructed over the learned representation space, enables visuomotor control for challenging manipulation tasks involving both rigid bodies and fluids, where the target is specified in a viewpoint different from what the robot operates on. When coupled with an auto-decoding framework, it can even support goal specification from camera viewpoints that are *outside the training distribution*. We further demonstrate the richness of the learned 3D dynamics model by performing future prediction and novel view synthesis. Finally, we provide detailed ablation studies regarding different system designs and qualitative analysis of the learned representations.

1 Introduction

Existing state-of-the-art model-based systems operating from vision treat images as 2D grids of pixels. The world, however, is three-dimensional. Modeling the environment from 3D enables amodal completion and allows the agents to operate from different views. Therefore, it is desirable to obtain good 3D-aware representations of the environment from 2D observations to achieve better task performance when an accurate inference of 3D information is essential, which can further make it easier to specify tasks and learn from third-person videos, etc.

One of the core questions of model learning in robotic manipulation is how to determine the state representation for learning the dynamics model. The desired representation should make it easy to capture the environment dynamics, exhibit a good 3D understanding of the objects in the scene, and be applicable to diverse object sets such as rigid or deformable objects, and fluids. One line of prior work learns the dynamics model directly in the image pixel space [1, 2, 3, 4]. However, modeling dynamics in such a high-dimensional space is challenging, and these methods typically generate blurry images when performing the long-horizon future predictions. Another line of work focused on only predicting task-relevant features identified as keypoints [5, 6, 7, 8, 9]. Such models perform well in terms of category-level generalization, i.e., the same set of keypoints can represent different instances within the same category, but are not sufficient to model objects with large variations like fluids and granular materials. Other methods learn dynamics in the latent space [10, 11, 12, 13, 14]. However, the majority of these methods learn dynamics models using 2D convolutional neural networks and reconstruction loss – which has the same problem as predicting dynamics in the image space, i.e. their learned representations lack equivariance to 3D transformations. Time contrastive networks [15], on the other hand, aim to learn viewpoint-invariant representations from multi-view inputs, but do not require detailed modeling of 3D contents. As a result, previously unseen scene configurations and camera poses are out-of-distribution samples for the state estimator. As we will see, this leads to wrong state estimates and results in faulty control trajectories.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

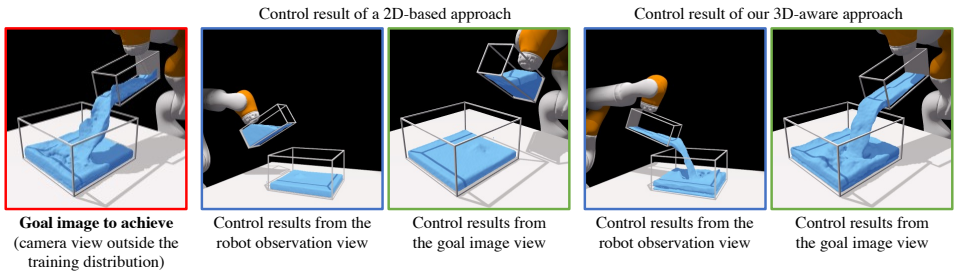


Figure 1: **Comparison of the control results between a 2D-based baseline and our 3D-aware approach.** The task here is to achieve the configuration shown on the left, observed from a viewpoint that is outside the training distribution. The agent only takes a single-view visual observation as input from a viewpoint that is vastly different from the goal (images with blue frames). Our method generalizes well in this scenario and outperforms the 2D-based baseline, demonstrating the benefits of the learned 3D-aware scene representations.

Meanwhile, recent work in computer vision has made impressive progress in the learning of 3D-structured neural scene representations. These approaches allow inference of 3D structure and appearance, trained only given 2D observations, either by overfitting on a single scene [16, 17, 18] or by generalizing across scenes [19, 20, 21]. Through their 3D inductive bias, the scene representations inferred by these models encode the scene contents with better accuracy and are invariant to changes in camera perspectives. It is desirable to push these ideas further to obtain a deeper understanding of how these methods, which directly reason over 3D, can bring in new characteristics and how they can be beneficial for dynamics modeling and complicated control tasks.

In this work, we aim to leverage recently proposed 3D-structure-aware implicit neural scene representations for visuomotor control tasks. We thus propose to embed neural radiance fields [18] in an auto-encoder framework, enabling tractable inference of the 3D-structure-aware scene state for dynamic environments. By additionally enforcing a time-contrastive loss on the estimated states, we ensure that the learned state representations are viewpoint-invariant. We then train a dynamics model that predicts the evolution of the state space conditioned on the input action, enabling us to perform control in the learned state space. Though the representation itself is 3D-structured, the convolutional encoder is not. At test time, we overcome this limitation by performing inference-via-optimization [22, 19], enabling accurate state estimation even for out-of-distribution camera poses and, therefore, control of tasks where the goal view is specified in an entirely unseen camera perspective. These contributions enable us to perform model-based visuomotor control of complex scenes, modeling both 3D dynamics of rigid objects and fluids. Through comparison with various baselines, the learned representation from our model is more precise at describing the contents of 3D scenes, which allows it to accomplish control tasks involving both rigid objects and fluids with significantly better accuracy (Figure 1). Please see our supplementary video for better visualization.

We summarize our contributions as follows: (i) We extend an autoencoding framework with a neural radiance field rendering module and time contrastive learning that allows us to learn 3D-aware scene representations for dynamics modeling and control purely from visual observations. (ii) By incorporating the auto-decoder mechanism at test time, our framework can adjust the learned representation and accomplish the control tasks with the goal specified from camera viewpoints outside the training distribution. (iii) We are the first to augment neural radiance fields using a time-invariant dynamics model, supporting future prediction and novel view synthesis across a wide range of environments with different types of objects.

2 Related Work

3D Scene Representation Learning. Prior work leverages the latent spaces of autoencoder-like models as learned representations of the underlying 3D scene to enable novel view synthesis from a single image [23, 24]. Eslami et al. [25] embed this approach in a probabilistic framework. To endow models with 3D structure, voxelgrids can be leveraged as neural scene representations [26, 27, 16, 28, 29] , while others have tried to embed an explicit 3D representation to enable inference from never-before-seen viewpoints [30]. Sitzmann et al. [19] propose to learn neural implicit representations of 3D shape and appearance supervised with posed 2D images via a differentiable renderer. Generalizing across neural implicit representations can also be realized by local conditioning on CNN features [31, 32, 33], but this does not learn a global representation of the scene state. Alternatively, gradient-based meta-learning has been proposed for faster inference of implicit neural

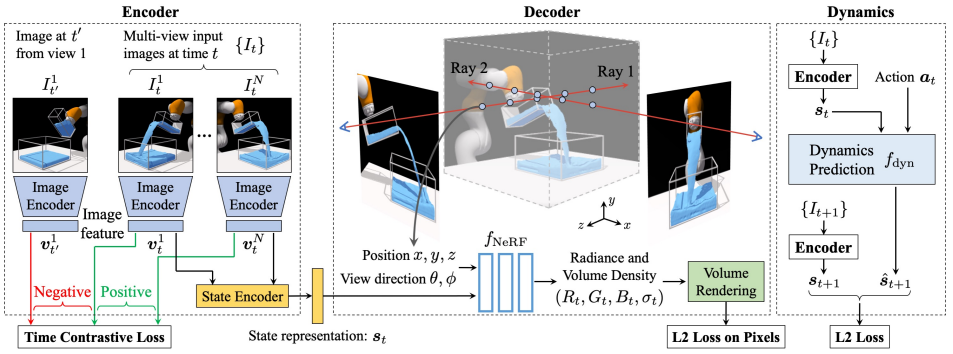


Figure 2: **Overview of the training procedure. Left:** an encoder that maps the input images into a latent scene representation. The images are first sent to an image encoder to generate the image feature representations $\boldsymbol{v}$. Then we combine the image features from the same time step using a state encoder to obtain the state representation $\boldsymbol{s}_t$. A time contrastive loss is applied to enable our model to be invariant to camera viewpoints. **Middle:** a decoder that takes the scene representation as input and generates the observation image conditioned on a given viewpoint. We use an L2 loss to ensure the reconstructed image to be similar to the ground truth image. **Right:** a dynamics model that predicts the future scene representations $\hat{\boldsymbol{s}}_{t+1}$ by taking in the current representation $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$. We use an L2 loss to enforce the predicted latent representation to be similar to the feature representation $\boldsymbol{s}_{t+1}$ extracted from the true visual observation I_{t+1}.

representations [34]. Deformable scenes can be modeled by transporting input coordinates to neural implicit representations with an implicitly represented flow field [35, 36, 37, 38, 39, 40] or time-variant latent code [41].

Model-Based RL in Robotic Manipulation. We can categorize model-based RL methods by whether they use physics-based or data-driven models, and whether they assume full state access or only visual observation. Methods that rely on first principles typically assume full-state information of the environment [42, 43] and require the knowledge of the object models, making them hard to generalize to novel objects or partially observable scenarios. For data-driven models, people have attempted to learn a dynamics model for closed-loop planar pushing [44] or dexterous manipulation [45]. Schenck and Fox [46, 47] tackle a similar fluid pouring task via closed-loop simulation. Although they have achieved impressive results, they rely on state estimators customized for specific tasks, limiting their applicability to more general and diversified manipulation tasks.

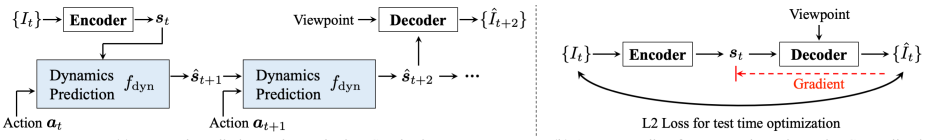
Various model-based RL methods have been proposed to learn state representations from visual observations, such as image-space dynamics [1, 2, 48, 4], keypoint representation [49, 6, 9], and low-dimensional latent space [10, 12, 13, 14]. Some works learn a meaningful representation space using reconstruction loss [12, 13]. Others jointly train the forward and inverse dynamics models [11], or use time contrastive loss to regularize the latent embedding [15]. We differ from the previous methods by explicitly incorporating a 3D volumetric rendering process during training.

3 3D-Aware Representations for Dynamics Modeling

Inspired by Neural Radiance Fields (NeRF) [18], we propose a framework that learns a viewpoint-invariant model for dynamic environments. As shown in Figure 2, our framework has three parts: (1) an encoder that maps the input images into a latent state representation, (2) a decoder that generates an observation image under a certain viewpoint based on the state representation, and (3) a dynamics model that predicts the future state representations based on the current state and the input action.

3.1 3D-Aware Scene Representation Learning

Neural Radiance Field. Given a 3D point $\boldsymbol{x} \in \mathbb{R}^3$ in a scene and a viewing direction unit vector $\boldsymbol{d} \in \mathbb{R}^3$ from a camera, NeRF learns to predict a volumetric radiance field. This is represented using a differentiable rendering function f_{NeRF} that predicts the corresponding density σ and RGB color $\boldsymbol{c}$ using $f_{\text{NeRF}}(\boldsymbol{x}, \boldsymbol{d}) = (\sigma, \boldsymbol{c})$. To render the color of an image pixel, NeRF integrates the information along the camera ray using $\hat{\boldsymbol{C}}(\boldsymbol{r}) = \int_{h_{\text{near}}}^{h_{\text{far}}} T(h)\sigma(h)\boldsymbol{c}(h)dh$, where $\boldsymbol{r}(h) = \boldsymbol{o} + h\boldsymbol{d}$ is the camera ray with its origin $\boldsymbol{o} \in \mathbb{R}^3$ and unit direction vector $\boldsymbol{d} \in \mathbb{R}^3$, and $T(h) = \exp(-\int_{h_{\text{near}}}^{h} \sigma(s)ds)$ is the accumulated transparency between the pre-defined near depth h_{near} and far depth h_{far} along that camera ray. The mean squared error between the reconstructed color $\hat{\boldsymbol{C}}$ and the ground truth $\boldsymbol{C}$ is:



(a) Forward Prediction and Novel-View Synthesis (b) Auto-Decoding for Extrapolate Viewpoint Generalization

Figure 3: **Forward prediction and viewpoint extrapolation.** (a) We first feed the input image(s) at time t to the encoder to derive the scene representation $\boldsymbol{s}_t$. The dynamics model then takes $\boldsymbol{s}_t$ and the corresponding action sequence as input to iteratively predict the future. The decoder synthesizes the visual observation conditioned on the predicted state representation and an input viewpoint. (b) We propose an auto-decoding inference-via-optimization framework to enable extrapolated viewpoint generalization. Given an input image I_t taken from a viewpoint outside the training distribution, the encoder first predicts the scene representation $\boldsymbol{s}_t$. Then the decoder reconstructs the observation $\hat{I}_t$ from $\boldsymbol{s}_t$ and the camera viewpoint from I_t. We calculate the L2 distance between I_t and $\hat{I}_t$ and backpropagate the gradient through the decoder to update $\boldsymbol{s}_t$. The updating process is repeated for K iterations, resulting in a more accurate $\boldsymbol{s}_t$ of the underlying 3D scene.

$$\mathcal{L}_{\text{rec}} = \sum_{\boldsymbol{r}} \|\hat{\boldsymbol{C}}(\boldsymbol{r}) - \boldsymbol{C}(\boldsymbol{r})\|_2^2. \tag{1}$$

Neural Radiance Field for Dynamic Scenes. One key limitation of NeRF is that it assumes the scene is static. For a dynamic scene, it must learn a separate radiance field f_{NeRF} for each time step. This severely limits the ability of NeRF to model environments that change over time, as it is both time-consuming and unclear how to transfer knowledge across time or when a new scene is similar to an old one. While other models have shown generalization across scenes [19, 50], these representations do not capture fine details. To enable f_{NeRF} to model dynamic scenes, we learn an encoding function f_{enc} that maps the visual observations to a feature representation $\boldsymbol{s}$ for each scene and learn the volumetric radiance field decoding function based on $\boldsymbol{s}$. Let $\{I_t\}$ denotes the set of 2D images that capture a 3D scene at time t from one or more camera viewpoints. The image taken from the ith viewpoint is represented as I_t^i. We use ResNet-18 [51] to extract a feature vector for each image. We take the output of ResNet-18 before the pooling layer and sent it to a fully-connected layer, resulting in a 256 dimension image feature $\boldsymbol{v}_t^i$. This image feature is concatenated with the corresponding camera viewpoint information (a 16-D vector obtained by flattening the camera view matrix). The concatenated feature is processed using a small multilayer perceptron (MLP) to generate the final image feature. The scene representation $\boldsymbol{s}_t$ at time t is generated by first averaging the image features across multiple camera viewpoints and then being encoded using another small MLP.

Given a 3D point $\boldsymbol{x}$, a viewing direction unit vector $\boldsymbol{d}$, and a scene representation $\boldsymbol{s}_t$, we learn a function $f_{\text{dec}}(\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{s}_t) = (\sigma_t, \boldsymbol{c}_t)$ to predict the radiance field represented by the density σ_t and RGB color $\boldsymbol{c}_t$. Similar to NeRF, we use the integrated information along the camera ray to render the color of image pixels from an input viewpoint and then compute the image reconstruction loss using Equation 1. During each training iteration, we render two images from different viewpoints to calculate more accurate gradient updates. f_{dec} depends on the scene representation $\boldsymbol{s}_t$, forcing it to encode the 3D contents of the scene to support rendering from different camera poses.

Time Contrastive Learning. To enable the image encoder to be viewpoint invariant, we regularize the feature representation of each image $\boldsymbol{v}_t^i$ using multi-view time contrastive loss (TCN) [15] (see Figure 2a). The TCN loss encourages features of images from different viewpoints at the same time step to be similar, while repulsing features of images from different time steps to be dissimilar. More specifically, given a time step t, we randomly select one image I_t^i as the anchor and extract its image feature $\boldsymbol{v}_t^i$ using the image encoder. Then we randomly select one positive image from the same time step but different camera viewpoint $I_t^{i'}$ and one negative image from a different time step but the same viewpoint $I_{t'}^i$. We use the same image encoder to extract their image features $\boldsymbol{v}_t^{i'}$ and $\boldsymbol{v}_{t'}^i$. Similar to [15], we minimize the following time contrastive loss:

$$\mathcal{L}_{\text{tc}} = \max\left(\|\boldsymbol{v}_t^i - \boldsymbol{v}_t^{i'}\|_2^2 - \|\boldsymbol{v}_t^i - \boldsymbol{v}_{t'}^i\|_2^2 + \alpha, 0\right), \tag{2}$$

where α is a hyper-parameter denoting the *margin* between the positive and negative pairs.

3.2 Learning the Predictive Model

After we have obtained the latent state representation $\boldsymbol{s}$, we use supervised learning to estimate the forward dynamics model, $\hat{\boldsymbol{s}}_{t+1} = f_{\text{dyn}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Given $\boldsymbol{s}_t$ and a sequence of actions $\{\boldsymbol{a}_t, \boldsymbol{a}_{t+1}, \dots\}$, we predict H steps in the future by iteratively feeding in actions into the one-step forward model. We implement f_{dyn} as an MLP network which is trained by optimizing the following loss function:

$$\mathcal{L}_{\text{dyn}} = \sum_{h=1}^{H} \|\hat{\boldsymbol{s}}_{t+h} - \boldsymbol{s}_{t+h}\|_2^2, \quad \text{where} \quad \hat{\boldsymbol{s}}_{t+h} = f_{\text{dyn}}(\hat{\boldsymbol{s}}_{t+h-1}, \boldsymbol{a}_{t+h-1}), \quad \hat{\boldsymbol{s}}_t = \boldsymbol{s}_t. \tag{3}$$

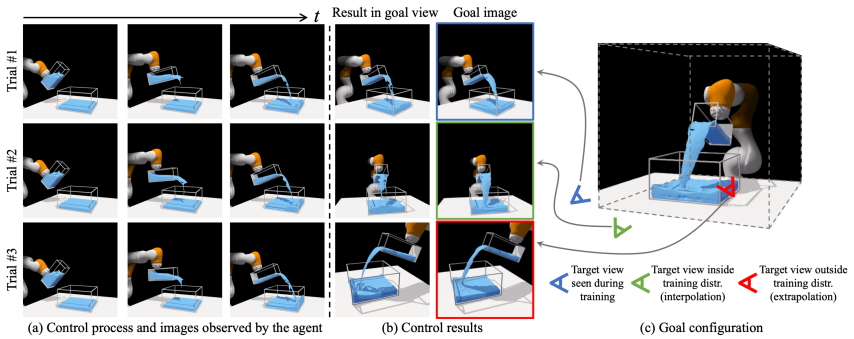


Figure 4: **Qualitative control results of our method on three types of testing scenarios.** The image on the right shows the target configuration we aim to achieve. The left three columns show the control process, which are also the input images to the agent. The fourth column is the control results from the same viewpoint as the goal image. Trial #1 specifies the goal using a different viewpoint from the agent's but has been encountered during training. Trial #2 uses a goal view that is an interpolation of training viewpoints. Trial #3 uses an extrapolated viewpoint that is outside the training distribution. Our method performs well in all settings.

We define the final loss as a combination of the image reconstruction loss, the time contrastive loss, and the dynamics prediction loss: $\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{tc}} + \mathcal{L}_{\text{dyn}}$. We first train the encoder f_{enc} and decoder f_{dec} together using stochastic gradient descent (SGD) by minimizing $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{tc}}$, which makes sure that the learned scene representation $\boldsymbol{s}$ encodes the 3D contents and is viewpoint-invariant. We then fix the encoder parameters, and train the dynamics model f_{dyn} by minimizing $\mathcal{L}_{\text{dyn}}$ using SGD. See the supplementary materials for the network architecture and training details.

4 Visuomotor Control

4.1 Online Planning for Closed-Loop Control

When given the goal image I^{goal} and its associated camera pose, we can feed them through the encoder f_{enc} to get the state representation for the goal configuration $\boldsymbol{s}^{\text{goal}}$. We use the same method to compute the state representation for the current scene configuration $\boldsymbol{s}_1$. The goal of the online planning problem is to find an action sequence $\boldsymbol{a}_1, \dots, \boldsymbol{a}_{T-1}$ that minimizes the distance between the predicted future representation and the goal representation at time T. As shown in Figure 3a, given a sequence of actions, our model can iteratively predict a sequence of latent state representations. The latent-space dynamics model can then be used for downstream closed-loop control tasks via online planning with model-predictive control (MPC), which is used to get feedback from the environment and give the agent a chance to adjust the current action sequence. We formally define the online planning problem as follows:

$$\min_{\boldsymbol{a}_1,\dots,\boldsymbol{a}_{T-1}} \quad \|\boldsymbol{s}^{\text{goal}} - \hat{\boldsymbol{s}}_T\|_2^2, \qquad \text{s.t.} \quad \hat{\boldsymbol{s}}_1 = \boldsymbol{s}_1, \hat{\boldsymbol{s}}_{t+1} = f_{\text{dyn}}(\hat{\boldsymbol{s}}_t, \boldsymbol{a}_t). \tag{4}$$

Many existing off-the-shelf model-based RL methods can be used to solve the MPC problem [2, 45, 12, 1, 6, 52, 53]. We experimented with random shooting, gradient-based trajectory optimization, cross-entropy method, and model-predictive path integral (MPPI) planners [54] and found that MPPI performed the best. MPPI is a sampling-based, gradient-free optimizer that considers temporal coordination between time steps when sampling action trajectories. At time t, the algorithm first samples M action sequences based on the current actions $\boldsymbol{a}_t, \dots, \boldsymbol{a}_{T-1}$ via $\hat{\boldsymbol{a}}_h^k = \boldsymbol{a}_h + \boldsymbol{n}_h^k, k \in \{1, \dots, M\}, h \in \{t, \dots, T-1\}$. Each noise sample $\boldsymbol{n}_h^k$, denoting the noise value at the hth time step of the kth trajectory, is generated using filtering coefficient β as the following:

$$\begin{aligned} &\boldsymbol{u}_h^k \sim \mathcal{N}(0, \Sigma) \quad \forall k \in \{1, \dots, M\}, h \in \{t, \dots, T-1\} \\ &\boldsymbol{n}_h^k = \beta \cdot \boldsymbol{u}_h^k + (1-\beta) \cdot \boldsymbol{n}_{h-1}^k, \text{where } \boldsymbol{n}_{h<t}^k = 0. \end{aligned} \tag{5}$$

We then roll them out in parallel using the learned model on the GPU to derive $\hat{\boldsymbol{s}}_T^k, k \in \{1, \dots, M\}$, and then re-weight the trajectories according to the reward to update the action sequence using a reward-weighting factor γ: $\boldsymbol{a}_h = (\sum_{k=1}^M \exp(\gamma \cdot R^k) \cdot \hat{\boldsymbol{a}}_h^k)/(\sum_{k=1}^M \exp(\gamma \cdot R^k)), \forall h \in \{t, \dots, T-1\}$, where $R^k = -\|\hat{\boldsymbol{s}}_T^k - \boldsymbol{s}^{\text{goal}}\|_2^2$. This procedure is repeated for L iterations at which point the best action sequence is selected. In our experiments, we specify the action space as the position and orientation of the arms' end-effector. The joint angle of the arm is calculated via inverse kinematics.

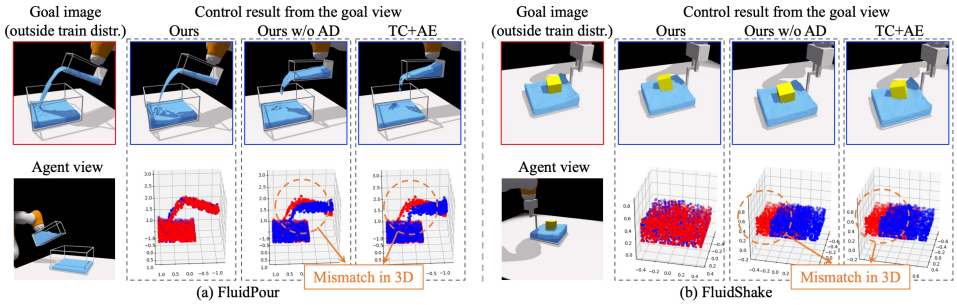


Figure 5: **Qualitative comparisons between our method and baseline approaches on the control tasks.** We show the closed-loop control results on the FluidPour and FluidShake environment. The goal image viewpoint (top-left image of each block) is outside the training distribution and is different from the viewpoint observed by the agent (bottom-left image of each block). Our final control results are much better than a variant that does not perform auto-decoding test-time optimization (Ours w/o AD) and the best-performing baseline (TC+AE), both of which fail to accomplish the task and their control results (blue points) exhibit an apparent deviation from the target configuration (red points) when measured in the 3D points space.

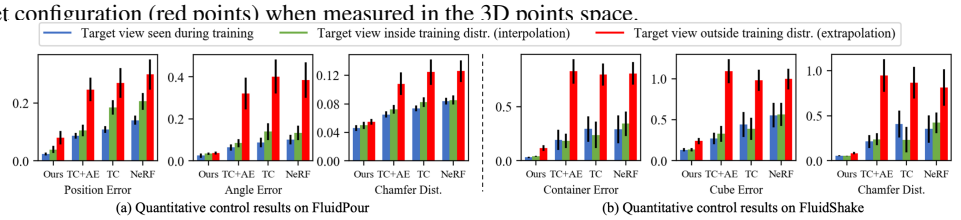


Figure 6: **Quantitative comparisons between our method and baselines on the FluidPour and FluidShake.** In each environment, we compare the results using three different evaluation metrics under three settings, i.e., (1) the target image view seen during training, (2) the target image view is inside the training distribution but not seen during training (interpolation), and (3) the target image view is outside the training distribution (extrapolation). The height of the bars indicates the mean, and the error bar denotes the standard error. Our model significantly outperforms all baselines under all testing settings.

4.2 Auto-Decoder for Viewpoint Extrapolation

End-to-end visuomotor agents can undergo significant performance drop when the test-time visual observations are captured from camera poses outside the training distribution. The convolutional image encoder suffers from the same problem as it is not equivariant to changes in the camera pose, meaning it has a hard time generalizing to out-of-distribution camera views. As shown in Figure 3b, when encountered an image from a viewpoint outside training distribution, with a pixel distribution vastly different from what the model is trained on, passing it through the encoder f_{enc} will give us an amortized estimation of the scene representation $\boldsymbol{s}_t$. It has a high chance that the decoded image is different from the ground truth as the viewpoint has never been encountered during training.

We fix this problem at test time by applying the inference-by-optimization (also named as an auto-decoding) framework that backpropagates through the volumetric renderer and through the neural implicit representation into the state estimate [22, 19]. This is inspired by the fact that the rendering function, $f_{\text{dec}}(\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{s}_t) = (\sigma_t, \boldsymbol{c}_t)$ is viewpoint equivariant, where the output only depends on the state representation $\boldsymbol{s}_t$, the location $\boldsymbol{x}$, and the ray direction $\boldsymbol{d}$, meaning that the output is invariant to the camera position along the camera ray, i.e., even we move the camera closer or farther away along the camera ray, f_{dec} still tends to generate the same results. We leverage this property and calculate the L2 distance between the input image and the reconstructed image $\mathcal{L}_{\text{ad}} = \|I_t - \hat{I}_t\|_2^2$, and then update the scene representation $\boldsymbol{s}_t$ using stochastic gradient descent. We repeat this updating process K times to derive the state representation of the underlying 3D scene. Note that this update only changes the scene representation while keeping the parameters in the decoder fixed. The resulting representation is used as $\boldsymbol{s}^{\text{goal}}$ in Equation 4 to solve the online planning problem.

5 Experiments

Environments. We consider the following four environments that involve both fluid and rigid objects for evaluating the proposed model and baseline approaches. The environments are simulated using NVIDIA FleX [55]. (1) FluidPour (Figure 8a): This environment contains a fully-actuated cup that pours fluids into a container at the bottom. (2) FluidShake (Figure 8b): A fully-actuated container moves on a 2D plane. Inside the container are fluids and a rigid cube floating on the surface. (3)

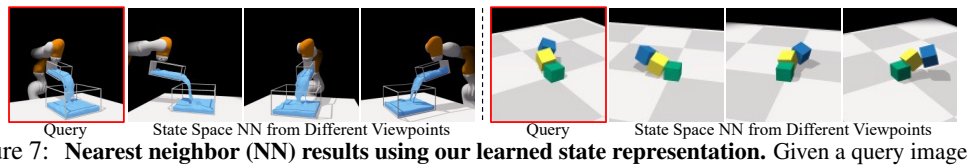


Figure 7: **Nearest neighbor (NN) results using our learned state representation.** Given a query image (red boundary), we search its nearest neighbors based on their state representation. Our learned scene representations can retrieve reasonable neighbor images, indicating that our state representations retain a good estimation of the contents inside the 3D scene and are invariant to camera poses.

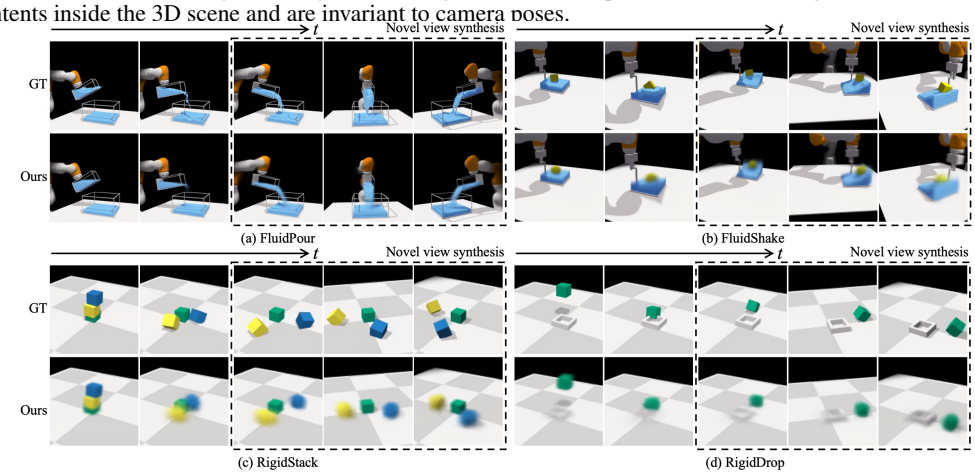


Figure 8: **Forward prediction and novel view synthesis on four environments.** Given a scene representation and an input action sequence, our dynamics model predicts the subsequent latent scene representation, which is used as the input of our decoder model to reconstruct the corresponding visual observation based on different viewpoints. In each block, we render images based on the open-loop future dynamic prediction. Images in the dotted box compare the novel view synthesis results of our model in the last time step with the ground truth from three different viewpoints.

RigidStack (Figure 8c): Three rigid cubes form a vertical stack and are released from a certain height but in different horizontal positions. They fall down and collide with each other and the ground. (4) RigidDrop (Figure 8d): A cube falls down from a certain height. There is a container fixed at a random position on the ground. The cube either falls into the container or bounces out. We use 20 camera views for all environments and generated $1,000$ trajectories of 300 time steps for both FluidPour and FluidShake, 800 trajectories of 80 time steps for RigidStack, and $1,000$ trajectories of 50 time steps for RigidDrop. Please see our supplementary materials for more environment details.

Evaluation Metrics. We use the first two environments, i.e., FluidPour and FluidShake, to measure the control performance, where we specify the target configuration of the control task using images from (1) one of the viewpoints encountered during training, (2) an interpolated viewpoint between training viewpoints, and (3) an extrapolated viewpoint outside the training distribution (Figure 4).

We provide quantitative evaluations on the control performance in FluidPour and FluidShake by extracting the particle set from the simulator, and measure the Chamfer distance between the result and the goal, which we denote as "Chamfer Dist.". In FluidPour, we provide additional measurements on the L2 distance of the position/orientation of the cup towards the goal, denoting as "Position Error" and "Angle Error" respectively. In FluidShake, we calculate the L2 distance of the container and cube's position towards the goal and denote them as "Container Error" and "Cube Error" respectively.

5.1 Baseline Methods

For comparison, we consider the following three baselines: **TC**: Similar to TCN [15], it only uses time contrastive loss for learning the image feature without having to reconstruct the scene. We learn a dynamics model directly on the image features for control. **TC+AE**: Instead of using Neural Radiance Fields to reconstruct the image, this method uses the standard convolutional decoder to reconstruct the target image when given a new viewpoint. This would then be similar to GQN [25] augmented with a time contrastive loss. **NeRF**: This method is a direct adaptation from the original NeRF paper [18] and is the same as ours except that it does not include the time contrastive loss during training and the auto-decoding test-time optimization. We use the same dynamics model shown in Figure 2b and train the model for each baseline respectively for dynamic prediction. We use the same feedback control method, i.e., MPPI [54], for our model and all the baselines.

5.2 Control Results

Goal Specification from Novel Viewpoints. Figure 4c shows the goal configuration, and we ask the learned model to perform three control trials where the goal is specified from different types of viewpoints. The left three columns show the MPC control process from the agent's viewpoint. The fourth column visualizes the final configuration the agent achieves from the same viewpoint as the goal image. Trial $\#1$ specifies the goal using a different viewpoint from the agent's but has been encountered during training. Trial $\#2$ uses a goal view that is an interpolation of training viewpoints. Our agent can achieve the target configuration with a decent accuracy. For trial $\#3$, we specify the goal view by moving the camera closer, higher, and more downwards with respect to the container. Note this goal image view is outside the distribution of training viewpoints. With the help of test-time auto-decoding optimization introduced in Section 4.2, our method can successfully achieve the target configuration as shown in the figure.

Baseline comparisons. We benchmark our model with the baselines by assessing their performance on the downstream control tasks. Figure 5 shows the qualitative comparison between our model (Ours), a variant of our model that does not perform the auto-decoding test-time optimization (Ours w/o AD), and the best-performing baseline (TC+AE) introduced in Section 5.1. We find that when the target view is outside the training distribution and vastly different from the agent view (bottom left image in each trial block), our full method shows a much better performance in achieving the target configuration. The variant without auto-decoding optimization and TC+AE fail to accomplish the task and exhibit an apparent deviation from the ground truth in the 3D points space.

We also provide quantitative comparisons on the control results. Figure 6 shows the performance in the FluidPour and FluidShake environments. We find our full model significantly outperforms the baseline approaches in both environments under all scenarios and evaluation metrics. The results effectively demonstrate the advantages of the learned 3D-aware scene representations, which contain a more precise encoding of the contents in the 3D environments.

5.3 Analysis of the Learned Model

To better understand why our 3D-aware model outperforms other baselines in the downstream control tasks, we provide a deeper investigation of the learned state representation.

Nearest Neighbor Search Using the Learned Representation. When the goal image is specified from a viewpoint different from the agent's view, to ensure the planning problem defined in Equation 4 still work, it is essential that the distance in the learned feature space reflects the distance in the actual 3D space, i.e., scenes that are more similar in the real 3D space should be closer in the learned feature space, even if the visual observations are captured from different viewpoints. We visualize the nearest neighbor results in Figure 7. Given a query image, we search its nearest neighbors based on their state representation $\boldsymbol{s}$ (introduced in Section 3.1). Even if the images look quite different from each other, the learned 3D-aware scene representations can retrieve reasonable neighborhood images that share similar 3D contents, indicating that the learned 3D-aware scene representations hold a good understanding of the real 3D scene and are invariant to camera viewpoints.

Dynamic Prediction and Novel View Synthesis. Conditioned on a scene representation and an input action sequence, our dynamics model f_{dyn} can iteratively predict the evolution of the scene representations. Our decoder can then take the predicted state representation and reconstruct the corresponding visual observation from a query viewpoint. Figure 8 shows that our model can accurately predict the future and perform novel view synthesis on four environments involving both fluid and rigid objects, suggesting its usefulness in trajectory optimization. Please check our video results in the supplementary material for better visualization.

6 Conclusion

In this paper, we proposed to learn viewpoint-invariant 3D-aware scene representations from visual observations using an autoencoding framework augmented with a neural radiance field rendering module and time contrastive learning. We show that the learned 3D representations perform well on the model-based visuomotor control tasks. When coupled with an auto-decoding test-time optimization mechanism, our method allows goal specification from a viewpoint outside the training distribution. We demonstrate the applicability of the proposed framework in a range of complicated physics environments involving rigid objects and fluids, which we hope can facilitate future visuomotor control works for complex 3D manipulation tasks.

References

[1] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *arXiv preprint arXiv:1605.07157*, 2016.

[2] F. Ebert, C. Finn, A. X. Lee, and S. Levine. Self-supervised visual planning with temporal skip connections. In *CoRL*, pages 344–356, 2017.

[3] L. Yen-Chen, M. Bauza, and P. Isola. Experience-embedded visual foresight. In *Conference on Robot Learning*, pages 1015–1024. PMLR, 2020.

[4] H. Suh and R. Tedrake. The surprising effectiveness of linear models for visual foresight in object pile manipulation. *arXiv preprint arXiv:2002.09093*, 2020.

[5] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2019.

[6] L. Manuelli, Y. Li, P. Florence, and R. Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2020.

[7] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066. IEEE, 2020.

[8] W. Gao and R. Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *arXiv preprint arXiv:2102.06279*, 2021.

[9] Y. Li, A. Torralba, A. Anandkumar, D. Fox, and A. Garg. Causal discovery in physical systems from videos. *Advances in Neural Information Processing Systems*, 33, 2020.

[10] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv preprint arXiv:1506.07365*, 2015.

[11] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *arXiv preprint arXiv:1606.07419*, 2016.

[12] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

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

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

[15] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[16] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. CVPR*, 2019.

[17] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

[18] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[19] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019.

[20] H.-Y. F. Tung, R. Cheng, and K. Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2595–2603, 2019.

[21] H.-Y. F. Tung, Z. Xian, M. Prabhudesai, S. Lal, and K. Fragkiadaki. 3d-oes: Viewpoint-invariant object-factorized environment simulators. *arXiv preprint arXiv:2011.06464*, 2020.

[22] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[23] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Interpretable transformations with encoder-decoder networks. In *Proc. ICCV*, volume 4, 2017.

[24] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Single-view to multi-view: Reconstructing unseen views with a convolutional network. *CoRR abs/1511.06702*, 1(2):2, 2015.

[25] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al. Neural scene representation and rendering. *Science*, 360 (6394):1204–1210, 2018.

[26] D. J. Rezende, S. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. *arXiv preprint arXiv:1607.00662*, 2016.

[27] T. Nguyen-Phuoc, C. Li, S. Balaban, and Y.-L. Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. *arXiv preprint arXiv:1806.06575*, 2018.

[28] H.-Y. F. Tung, R. Cheng, and K. Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. *Proc. CVPR*, 2019.

[29] J.-Y. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. Tenenbaum, and B. Freeman. Visual object networks: image generation with disentangled 3d representations. In *Proc. NIPS*, pages 118–129, 2018.

[30] G. Saponaro, L. Jamone, A. Bernardino, and G. Salvi. Beyond the self: Using grounded affordances to interpret and describe others' actions. *IEEE Transactions on Cognitive and Developmental Systems*, 12(2):209–221, 2019.

[31] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019.

[32] A. Trevithick and B. Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. *arXiv preprint arXiv:2010.04595*, 2020.

[33] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. *arXiv preprint arXiv:2012.02190*, 2020.

[34] V. Sitzmann, E. R. Chan, R. Tucker, N. Snavely, and G. Wetzstein. Metasdf: Meta-learning signed distance functions. *arXiv preprint arXiv:2006.09662*, 2020.

[35] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5379–5389, 2019.

[36] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R.-M. Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

[37] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[38] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a deforming scene from monocular video. *arXiv preprint arXiv:2012.12247*, 2020.

[39] Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020.

[40] Y. Du, Y. Zhang, H.-X. Yu, J. B. Tenenbaum, and J. Wu. Neural radiance flow for 4d view synthesis and video processing. *arXiv e-prints*, pages arXiv–2012, 2020.

[41] T. Li, M. Slavcheva, M. Zollhoefer, S. Green, C. Lassner, C. Kim, T. Schmidt, S. Lovegrove, M. Goesele, and Z. Lv. Neural 3d video synthesis. *arXiv preprint arXiv:2103.02597*, 2021.

[42] F. R. Hogan and A. Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. *arXiv preprint arXiv:1611.08268*, 2016.

[43] J. Zhou, Y. Hou, and M. T. Mason. Pushing revisited: Differential flatness, trajectory planning, and stabilization. *The International Journal of Robotics Research*, 38(12-13):1477–1489, 2019.

[44] M. Bauza, F. R. Hogan, and A. Rodriguez. A data-efficient approach to precise and controlled pushing. In *Conference on Robot Learning*, pages 336–345. PMLR, 2018.

[45] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[46] C. Schenck and D. Fox. Reasoning about liquids via closed-loop simulation. *arXiv preprint arXiv:1703.01656*, 2017.

[47] C. Schenck and D. Fox. Perceiving and reasoning about liquids using fully convolutional networks. *The International Journal of Robotics Research*, 37(4-5):452–471, 2018.

[48] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[49] T. D. Kulkarni, A. Gupta, C. Ionescu, S. Borgeaud, M. Reynolds, A. Zisserman, and V. Mnih. Unsupervised learning of object keypoints for perception and control. *Advances in neural information processing systems*, 32:10724–10734, 2019.

[50] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[51] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

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

[53] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake. Propagation networks for model-based control under partial observation. In *ICRA*, 2019.

[54] G. Williams, A. Aldrich, and E. Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[55] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.