3D-IntPhys: Learning 3D Visual Intuitive Physics for Fluids, Rigid Bodies, and Granular Materials

Anonymous Author(s)
Affiliation
Address
email

Abstract

Given a visual scene, humans have strong intuitions about how a scene can evolve over time under given actions. The intuition, often termed visual intuitive physics, is a critical ability that allows us to make effective plans to manipulate the scene to achieve desired outcomes without relying on extensive trial and error. In this paper, we present a framework capable of learning 3D-grounded visual intuitive physics models purely from unlabeled images. Our method is composed of a conditional Neural Radiance Field (NeRF)-style visual frontend and a 3D point-based dynamics prediction backend, in which we impose strong relational and structural inductive bias to capture the structure of the underlying environment. Unlike existing intuitive point-based dynamics works that rely on the supervision of dense point trajectory from simulators, we relax the requirements and only assume access to multi-view RGB images. This enables the proposed model to handle scenarios where accurate point estimation and tracking are hard or impossible. We evaluate the models on three challenging scenarios involving fluid, granular materials, and rigid objects, where standard detection and tracking methods are not applicable. We show our model can make long-horizon future predictions by learning from raw images and significantly outperforms models that do not employ an explicit 3D representation space. We also show that, once trained, our model can achieve strong generalization in complex scenarios under extrapolate settings.

1 Introduction

Humans can achieve a strong intuitive understanding of the 3D physical world around us simply from visual perception [5, 8, 51, 50, 45, 11]. As we constantly make physical interactions with the environment, the intuitive physical understanding applies to objects of a wide variety of materials [6, 55]. For example, after watching videos of water pouring and doing the task ourselves, we can develop a mental model of the interaction process and predict how the water will move when we apply actions like tilting or shaking the cup (Figure 1). The ability to predict the future evolution of the physical environment is extremely useful for humans to plan our behavior and perform everyday manipulation tasks. It is thus desirable to develop computational tools that learn 3D-grounded models of the world purely from visual observations that can generalize and apply to objects with complicated physical properties like fluid and granular materials.

There has been a series of works on learning intuitive physics models of the environment from data. However, most existing work either focuses on 2D environments [59, 1, 21, 64, 20, 4, 43, 28, 67, 24, 23, 48, 32, 19, 30, 57, 22, 12, 65] or has to make strong assumptions about the accessible information of the underlying environment [35, 34, 46, 42, 70, 53, 47, 7, 2, 26] (e.g., full-state information of

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

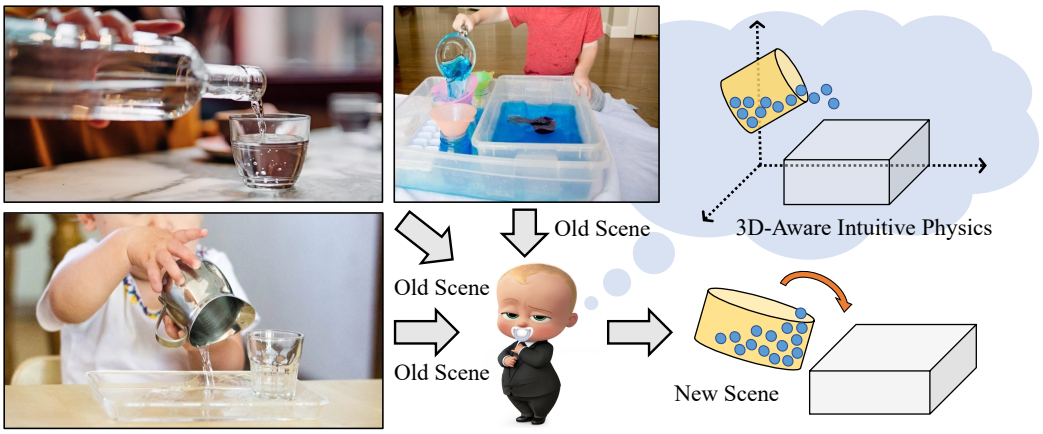


Figure 1: **Visual Intuitive Physics Grounded in 3D Space.** Humans have a strong intuitive understanding of the physical environment. We can predict how the environment would evolve when applying specific actions. This ability roots in our understanding of 3D and applies to objects of diverse materials, which is essential when planning our behavior to achieve specific goals. In this work, we leverage a combination of implicit neural representation and particle representation to build 3D-grounded visual intuitive physics models of the world that applies to objects with complicated physical properties, such as fluids, rigid objects, and granular materials.

the fluids represented as points). The limitations prevent their use in tasks requiring an explicit 3D understanding of the environments and make it hard to extend to more complicated real-world environments where only visual observations are available. There are works aiming to address this issue by learning 3D-grounded representation of the environment and modeling the dynamics in a latent vector space [33, 31]. However, these models typically encode the entire scene into one single vector. Such design does not capture the structure of the underlying systems, limiting its generalization to compositional systems or systems of different sizes (e.g., unseen container shapes or different numbers of floating ice cubes when pouring water).

In this work, we propose 3D Visual Intuitive Physics (3D-IntPhys), a framework that learns intuitive physics models of the environment with explicit 3D and compositional structures, purely from visual observations. Specifically, the model consists of (1) a perception module based on conditional Neural Radiance Fields (NeRF) [40, 68] that transforms the input images into 3D point representations and (2) a dynamics module instantiated as graph neural networks to model the interactions between the points and predict their evolutions over time. Since our model only assumes access to visual observation and does not rely on the tracking information of the points, we train the dynamics model using (1) a distribution-based loss function measuring the difference between the predicted point sets and the actual point distributions at the future timesteps and (2) a spacing loss to avoid degenerated point set predictions. Our perception module learns spatial-equivariant representations of the environment grounded in the 3D space, which then transforms into points as a flexible representation to describe the system's state. Our dynamics module regards the point set as a graph and exploits the compositional structure of the point systems. The structures allow the model to capture the compositionality of the underlying environment, handle systems involving objects with complicated physical properties (e.g., fluid and granular materials), and perform extrapolated generalization, which we show via experiments greatly outperform various baselines without a structured 3D representation space.

2 Related Work

Visual dynamics learning. Existing works learn to predict object motions from pixels using frame-centric features [1, 20, 4, 24, 23, 52, 29, 58, 69, 10, 25, 62] or object-centric features [21, 60, 28, 43, 27, 57, 14, 22, 44, 66], yet, most works only demonstrate the learning in 2D scenes with objects moving only on a 2D plane. We argue that one reason that makes it hard for these existing methods to be applied to general 3D visual scenes is because existing methods often operate on view-dependent features that can change dramatically due to changes in the camera viewpoint, which shouldn't have any effect on the actual motion of the objects. Recent works [9] have shown that only methods

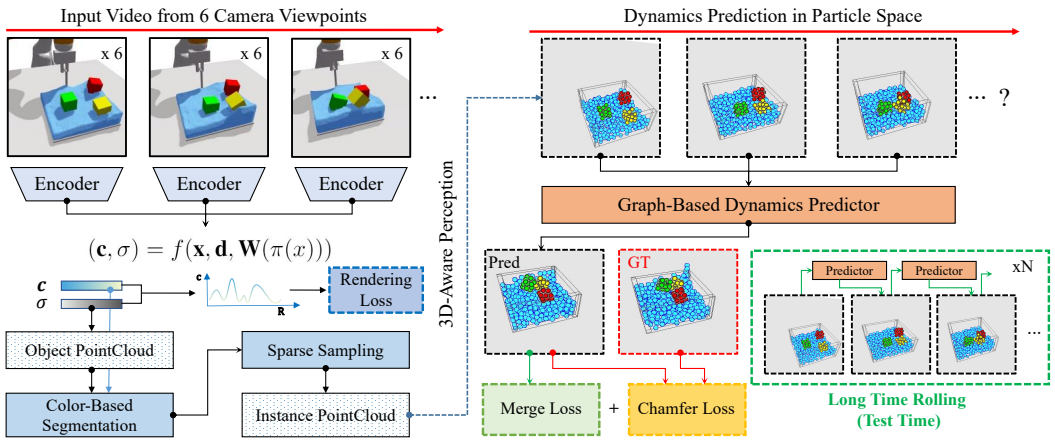


Figure 2: **Overview of 3D Visual Intuitive Physics (3D-IntPhys).** Our model consists of two major components: **Left:** The perception module maps the visual observations into implicit neural representations of the environment. We then subsample from the reconstructed implicit volume to obtain a particle representation of the environment. **Right:** The dynamics module, instantiated as graph neural networks, models the interaction within and between the objects and predicts the evolution of the particle set.

that use 3D view-invariant representations can pave the way toward human-level physics dynamics prediction in diverse scenarios.

Researchers have attempted to learn object motion in 3D [54, 63, 39, 33]. Tung et al. and Xu et al. [54, 63] use object-centric volumetric representations inferred from RGB-D to predict object motion, yet, these volumetric approaches have much higher computation costs than 2D methods due to the 4D representation bottleneck, which hinders them from scaling up to more complex scenes. Manuelli et al. [39] use self-supervised 3D keypoints and Driess et al. [15, 16] use implicit representations to model multi-object dynamics but cannot handle objects with high degrees of freedom like fluid and granular materials. Li et al. [33] uses neural implicit representation to reduce the potential computational cost, yet the works have not shown how the approach can generalize to unseen scenarios. Our works aim to solve the tasks of learning generalizable object dynamics in 3D by combining the generalization strength of input-feature-conditioned implicit representation and point-based models.

Point-based dynamics models. Existing works in point- and mesh-based dynamics models [35, 41, 56, 46, 42] have shown impressive results in predicting the dynamics of rigid objects, fluid [35, 41, 56, 46, 3, 13], deformable objects [35, 41, 46], and clothes [42, 37]. Most works require that the 3D states of the points are accessible during training and testing time, yet, such information is usually not accessible in a real-world setup. Li et al. [34] learn a visual frontend to infer 3D point states from images, yet, the approach still requires 3D point states and trajectories during training time. Shi et al. [49] propose to learn point dynamics directly from vision, but they only consider elasto-plastic objects consisting of homogeneous materials. How to learn about 3D point states and their motion from raw pixels remain a question. Our paper tries to build the link from pixels to points using recent advances in unsupervised 3D inference from images using NeRF [40, 68].

3 Methods

We present 3D Visual Intuitive Physics (3D-IntPhys), a model that learns to simulate physical events from unlabeled images (Figure 2). 3D-IntPhys contains a perception module that transforms visual observations into a 3D point cloud that captures the geometry of the objects in the scene (Section 3.1) and a point-based simulator that learns to simulate the rollout trajectories of the points (Section 3.2). The design choice of learning physics simulation in a 3D-point representation space enables stronger simulation performance and generalization ability. The performance gain mainly comes from the fact that describing/learning objects' motion and interactions in 3D are easier compared to doing so in 2D since objects live and move persistently in the 3D space. 3D-IntPhys also supports better

generalization ability since its neural architecture explicitly models how local geometries of two objects/parts interact, and these local geometries and interactions can be shared across different combinations of objects and stuff.

Although 3D-IntPhys learns to simulate in a 3D representation space, we show it can learn without any 3D supervision such as dense point trajectories as in previous work [46, 35]. Dense point trajectories are hard and sometimes impossible to obtain in the real world, e.g., capturing the trajectories of each water point. 3D-IntPhys does not require such 3D supervision and can simply learn by observing videos of the scene evolution.

3.1 2D-to-3D Perception Module

Given a static scene, the perception module learns to transform one or a few posed RGB images, $\mathbf{I} = \{(I_i, \pi_i) | i \in \{1, 2, \cdots, N_v\}\}$, taken from N_v different views, into a complete 3D point cloud of the scene, $\mathbf{X}$. We train the model in an unsupervised manner through view reconstruction, using a dataset consisting of N_t videos, where each video has N_f frames, and each frame contains images taken from N_v viewpoints.

Neural Radiance Field (NeRF). NeRF [40] learns to reconstruct a volumetric radiance field of a scene from unlabeled multi-view images. After training, the model learns to predict the RGB color $\mathbf{c}$ and the corresponding density σ of a query 3D point $\mathbf{x} \in \mathbb{R}^3$ from the viewing direction $\mathbf{d} \in \mathbb{R}^3$ with a function $(\mathbf{c}, \sigma) = f(\mathbf{x}, \mathbf{d})$. We can formulate a camera ray as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $o \in \mathbb{R}^3$ is the origin of the ray and $d \in \mathbb{R}^3$ is a unit direction vector for the ray. The volumetric radiance field can then be rendered into a 2D image via $\hat{\mathbf{C}}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)c(t)dt$, where $T(t) = \exp(-\int_{t_n}^{t} \sigma(s)ds)$ handles occlusion. The rendering range is controlled by the depths of the near and far plane (i.e., t_n and t_f). We can train NeRF through view prediction by:

$$\mathcal{L} = \sum\nolimits_{\mathbf{r} \in \mathcal{R}(\mathbf{P})} \|\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})\|, \tag{1}$$

where $\mathcal{R}(\mathbf{p})$ is the set of camera rays sampled from target camera pose $\mathbf{p}$.

Image-conditioned NeRF. To infer the NeRF function from an image, previous work proposed to encode the input image into a vector, with a CNN encoder, as a conditioning input to the target NeRF function [33]. We found this type of architecture is in general hard to train and does not generalize well. Instead, we adopt pixelNeRF [68], which conditions NeRF rendering with local features, as opposed to global features. Given an image $\mathbf{I}$ in a scene, pixelNeRF first extracts a feature volume using a CNN encoder $\mathbf{W} = E(\mathbf{I})$. For a point $\mathbf{x}$ in the world coordinate, we retrieve its feature vector by projecting it onto the image plane, so that we can get the feature vector $\mathbf{W}(\pi(x))$. pixelNeRF combines the feature vector together with the 3D position of that point and predict the RGB color and density information:

$$\mathbf{V}(\mathbf{x}) = (\mathbf{c}, \sigma) = f(\mathbf{x}, \mathbf{d}, \mathbf{W}(\pi(x))). \tag{2}$$

PixelNeRF can also incorporate multiple views to improve predictions when more input views are available; this will help decrease ambiguity caused by occlusion and will greatly help us get better visual perceptions of the target scene.

3D point representation from pixelNeRF. From a few posed RGB images, $\mathbf{I}$, of a scene s, we infer a set of points for O_s target object (such as fluid, cube) in the scene. We achieve this by first sampling a set of points according to the predicted occupancy measure, then clustering the points into objects using color information.

We found that sampling with low resolution will hurt the quality of the rendered point cloud to generate objects with inaccurate shapes, while sampling with high resolution will increase the computation for training the dynamics model since the input size increases. To speed up training while maintaining the quality of the reconstructed point cloud, we first infer the points with higher resolution and do sparse sampling of each point cloud using FPS (Farthest Point Sampling) [17].

Next, we cluster the inferred points into objects using color prior. Here we assume the model is encoded with the prior that water is blue, sand is yellow, and cubes can be either red, green or yellow.

If a new color is introduced, we can incorporate the information into the color prior to help us better segment the scene. We include more details in the supplementary materials.

3.2 Point-Based Dynamics Simulator

Given the point representation at the current time step, $\mathbf{X}_t$, the dynamics simulator predicts the points' evolution T steps in the future, $\{\mathbf{X}_{t+1}, \mathbf{X}_{t+2}, \cdots \mathbf{X}_{t+T}\}$, using graph-based networks [46, 35]. Given point representation, $\mathbf{X}$, we form a graph (V, E) based on the distance between points. If the distance between two points is smaller than a threshold δ, we include an edge between these two points. Each vertex $v_i = (\dot{x}_i, a_i^v) \in V$ contains the velocity of the point, $\dot{x}_i$, and point attributes, a_i^v, to indicate the point's type. For each relation, $(i, j) \in E$, we have its associated relation attribute a_{ij}^e, indicating the types of relation and the relative distance between the connected points.

Spatial message passing and propagation. At time step t, we can do message passing to update the point and relation representations in the graph:

$$g_{ij,t} = Q_e(v_{i,t}, v_{j,t}, a_{ij}^e) \qquad (i,j) \in E \tag{3}$$

$$h_{i,t} = Q_v(v_{i,t}, \sum\nolimits_{k \in \{j|(i,j)\in E\}} g_{ik,t}) \qquad v_i \in V \tag{4}$$

where Q_v and Q_e are encoders for vertices and relations respectively. Though this kind of message passing can help with updating representation or passing of forces, it can only share one-hop information in each step, limiting its performance on instantaneous passing of forces. To improve long-range instantaneous effect propagation, we use multi-step message propagation as in [36, 35]. Starting from propagation step 0 with an initialization step where $h_{i,t}^0 = h_{i,t}$ for all vertices and $g_{ij,t}^0 = g_{ij,t}$ for all edges, we do L steps of propagation by

$$\text{Step } l \in \{1, 2, ..., L\}: \qquad g_{ij,t}^l = P_e(g_{ij,t}^{l-1}, h_{i,t}^{l-1}, h_{j,t}^{l-1}) \qquad (i,j) \in E \tag{5}$$

$$h_{i,t}^l = P_v(h_{i,t}^{l-1}, \sum\nolimits_{k \in \{j|(i,j)\in E\}} g_{ik,t}^l) \quad v_i \in V \tag{6}$$

where P_e, P_v are propagation functions of nodes and edges, respectively, and $g_{ij,t}^l$ is the effect of relation (i, j) in propagation step l. $h_{i,t}^l$ is the hidden states for each point in the propagation process. Finally, we have the predicted states of points at time step $t + 1$ after L steps of propagation:

$$\hat{v}_{i,t+1} = f_{\text{pred}}(h_{i,t}^L). \tag{7}$$

Environments. We assume that the surrounding environment (e.g., the background, the table) is known and the robot/tool/container are of known shape and fully actuated, where the model has access to their complete 3D state information. We convert the full 3D states into points through sampling on the 3D meshes and include these points in the prediction of the graph-based dynamics.

Fluids, Rigid Bodies, and Granular Materials. We distinguish these different materials by using different point attributes a_i^v. To model interactions between different materials, we set different relation attributes a_{ij}^e in Equation 3 to distinguish different interaction (e.g., Rigid-Fluids, Fluids-Fluids, Granular-Pusher). For rigid objects, to ensure the object shapes remain consistent throughout the rollout predictions, we add a differentiable rigid constraint in the prediction head following [35].

Training dynamics model without point-level correspondence. Since our perception model parses each RGB image into object-centric point clouds independently, there does not exist an explicit one-to-one correspondence for points across frames. To handle this, we measure the Chamfer distance between the prediction $\hat{\mathbf{X}}_t = (\hat{V}_t, \hat{E}_t)$ from the dynamics network and the inferred point state $\mathbf{X}_t = (V_t, E_t)$ from the perception module and treat it as the objective function. The Chamfer distance between two point cloud $\hat{V}$ and V is defined as:

$$L_c(\hat{V}, V) = \frac{1}{\|\hat{V}\|} \sum_{x \in \hat{V}} \min_{y \in V} \|x - y\|_2^2 + \frac{1}{\|V\|} \sum_{x \in V} \min_{y \in \hat{V}} \|x - y\|_2^2. \tag{8}$$

We found that training the model with Chamfer distance in dense scenes with granular materials will often lead to predictions with unevenly distributed points where some points stick too close to each other. To alleviate this issue, we further introduce a spacing loss L_s, which penalizes the gated

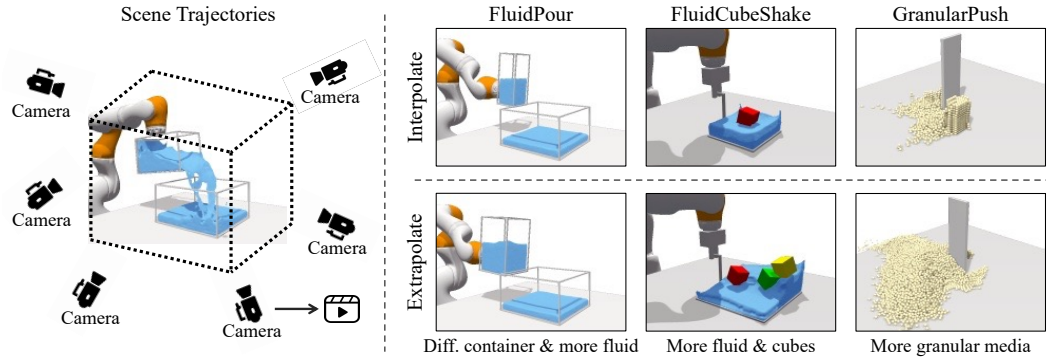


Figure 3: **Data Collection and Evaluation Setups. Left:** We collect multi-view videos of the environment from six cameras. **Right:** We consider a diverse set of evaluating environments involving fluids, rigid objects, granular materials, and their interactions with the fully-actuated container and the environment. We evaluate the learned visual intuitive physics model on both the interpolated settings (i.e., seen environment but with different action sequences) and extrapolated settings (i.e., unseen environment with different amounts of fluids, cubes, granular pieces, and containers of different sizes).

distance (gated by d_{min}) of nearest neighbor of each point to ensure enough space between points:

$$L_s(\hat{V}) = \sum\nolimits_{v \in \hat{V}} (\text{ReLU}(d_{\text{min}} - \min_{v' \in \{\hat{V} \setminus v\}} \|v' - v\|_2^2))^2. \tag{9}$$

The one-step prediction loss L_{dy} for training the dynamics model is $L_c(\hat{V}, V) + \sigma L_s(\hat{V})$ where σ reweights the second loss. To improve long-term rollout accuracy, we train the model with two-step predictions using the first predicted state as input and feed it back into the model to generate the second predicted state. With the two-step loss, the model becomes more robust to errors generated from its own prediction. Finally, the L_{dy} losses for all rolling steps are summed up to get the final loss for this trajectory. More implementation details are included in the supplementary material.

4 Experiments

The experiment section aims to answer the following three questions. (1) How well can the visual inference module capture the content of the environment, i.e., can we use the learned representations to reconstruct the scene? (2) How well does the proposed framework perform in scenes with objects of complicated physical properties (e.g., fluids, rigid and granular objects) compared to baselines without explicit 3D representations? (3) How well do the models generalize in extrapolate scenarios?

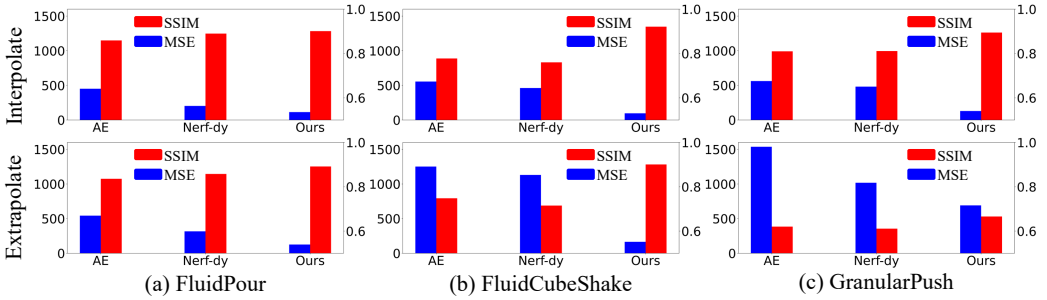
Datasets. We generated three simulated datasets using the physics simulator Nvidia FleX [38]. Each of the datasets represents one specific kind of manipulation scenario, where a robot arm interacts with rigid, fluid, and granular objects (Figure 3). For each of the three scenarios, we change some properties of objects in the scene, e.g., the shape of the container, the amount of water, and the color/number of cubes, to make it diverse. To test the generalization capability of the trained model, we design extrapolated datasets where the data is generated from an extrapolated set of parameters outside the training distribution.

a) FluidPour. This scenario contains a fully-actuated cup pouring fluid into a container. We design the extrapolate dataset to have a larger container, more quantity of fluid, and different pouring actions.

b) FluidCubeShake. This scenario contains a fully-actuated container that moves on top of a table. Inside the container are fluids and cubes with diverse colors. We design the extrapolate dataset to have different container shapes, number of cubes, cube colors, and different shaking actions.

c) GranularPush. This environment contains a fully-actuated board pushing a pile of granular pieces. We design the extrapolate dataset to have a larger quantity of granular objects in the scene than the model has ever seen during training.

Baselines. We compare our method with two baselines, NeRF-dy [33] and autoencoder (AE) (similar to GQN [18] augmented with a latent-space dynamics model). NeRF-dy is a 3D-aware framework



(a) FluidPour (b) FluidCubeShake (c) GranularPush

Figure 4: **Quantitative Results of the Perception Module.** We compare our method with two baselines: autoencoder (AE) and NeRF-dy [33]. The blue bar shows the mean squared error (MSE) between the reconstructed image and the ground truth (lower is better). The red bar indicates the structural similarity index measure (SSIM) of the prediction (higher is better). Our method performs the best across all metrics and has shown huge improvements especially in extrapolate settings.

that also learns intuitive physics from multi-view videos. Yet, instead of learning the object dynamics with explicit and compositional 3D representations, the model learns dynamics models with implicit 3D representations in the form of a single latent vector. We also compare our method with an autoencoder-based reconstruction model (AE) [18] that can perform novel-view synthesis but is worse at handling 3D transformations than neural implicit representations. AE first learns scene representations through per-frame image reconstruction, and then it learns a dynamics model on top of the learned latent representations.

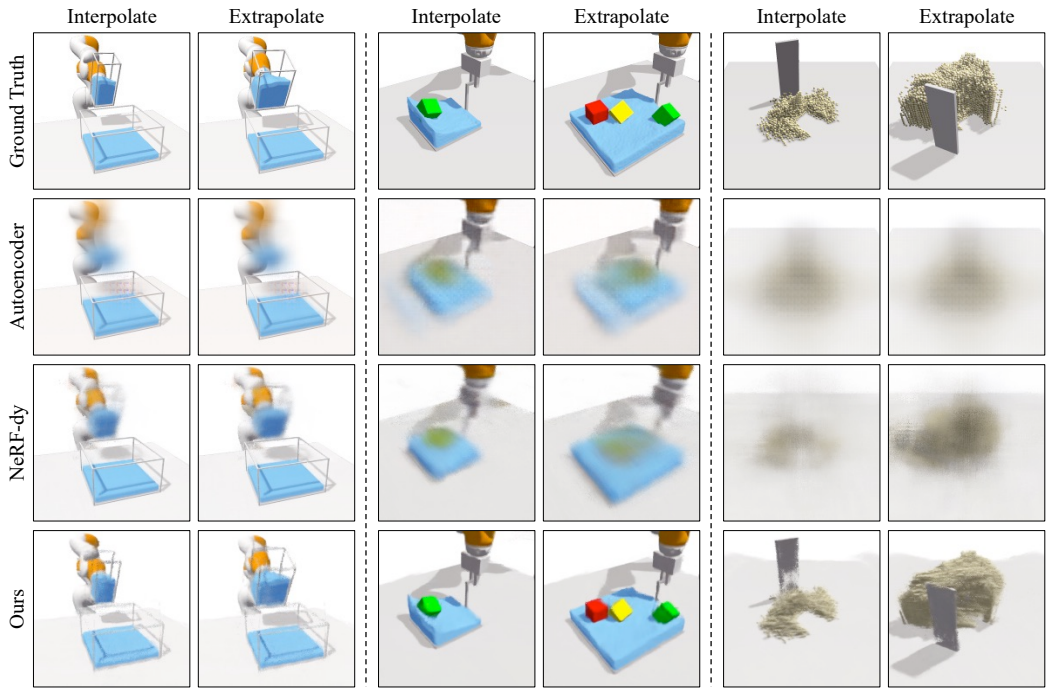
Implementation details. We train and test the perception module with 6 camera views. To obtain a set of points from the learned perception module, we sample points on a $40 \times 40 \times 40$ grid from an area of $2.5\text{cm} \times 2.5cm \times 2.5cm$ at the center of the table for FluidPour and FluidCubeShake, and on a $70 \times 70 \times 70$ grid from an area of $6\text{cm} \times 6cm \times 6cm$ for GranularPush. We evaluate and include points with a density (measured by the occupancy in the predicted neural radiance fields) larger than 0.99. We subsample the inferred points with FPS with a ratio of around 5% for FluidPour and 10% for FluidCubeShake and GranularPush. The threshold d_{min} is set to 0.08 and the weighting σ is set to 10. Details for data generation parameters, model architecture, training schema, and more data samples are included in the supplementary material.

4.1 Image Reconstruction From Learned Scene Representations

To learn the scene dynamics well, a model will need to operate in a space where critical scene information is well-preserved. We test how well the perception modules capture scene information by evaluating the visual front-end of all models on their ability to reconstruct the observed scene from the inferred representations. We measure the difference between the reconstructed and ground truth images with Mean Squared Error (MSE) and Structural Similarity (SSIM) in pixel level (Figure 4). Our perception module outperforms all baselines in all three environments. The performance gap is exaggerated in extrapolate settings, especially in scenarios that involve complex interactions between rigid and deformable materials (see Figure 5 for qualitative comparisons).

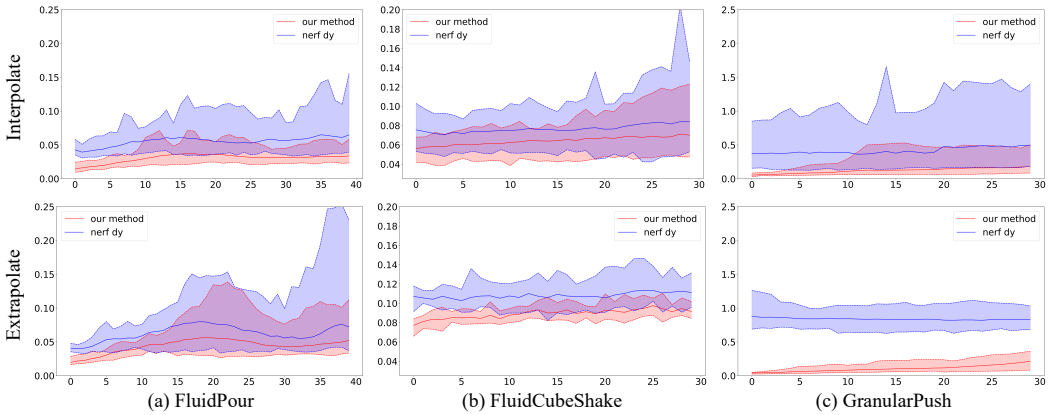
4.2 Learned Visual Dynamics On In-Distribution Held-Out Scenes

Next, we compare long-term rollouts in the 3D space. We evaluate the models using the Chamfer distance between the predicted point cloud and the ground truth. For NeRF-dy, we first predict the rollouts latent vectors, then decode the vectors into the point cloud with the learned NeRF decoder. We exclude the comparison with AE since it is unclear how to decode the learned representations into point clouds. We show quantitative comparison in Figure 6 and qualitative results in Figure 7. 3D-IntPhys can learn reasonable scene dynamics in all scenarios and significantly outperforms NeRF-dy. While NeRF-dy can learn relatively reasonable movements of fluids, it fails to learn complex dynamics such as the floating cube and the morphing of the granular materials. The results suggest that the proposed explicit 3D point-based representations are critical to learning complex multi-material dynamics.



(a) FluidPour (b) FluidCubeShake (c) GranularPush

Figure 5: **Qualitative Reconstruction Results of the Perception Module.** This figure compares the image reconstruction performance of our method with autoencoder (AE) and NeRF-dy [33]. The images generated by our method contain more visual details and are much better aligned with the ground truth. Our method is also more consistent in the face of 3D transformations than AE and much better at handling large scene variations than NeRF-dy, especially in extrapolate settings (e.g., more cubes than during training FluidCubeShake).



(a) FluidPour (b) FluidCubeShake (c) GranularPush

Figure 6: **Quantitative Results of the Dynamics Module.** This figure compares our method and the NeRF-dy [33] baseline on their long-horizon open-loop future prediction performance. The loss is measured as the Chamfer distance between the predicted particle set evolution and the actual future. Our method outperforms the baseline in both interpolate and extrapolate settings, showing the benefits of explicit 3D modeling.

4.3 Generalization on Out-of-Distribution Scenes

To test the generalization ability of the models, we introduce extrapolate settings of all of the three scenarios. See "Extrapolate" results in Figure 4, 5, 6, and 7. The proposed 3D-IntPhys generalizes well to extrapolate settings both at the visual perception stage and the dynamics prediction stage, whereas NeRF-dy and autoencoder both fail at generalizing under extrapolate settings. For example, in **FluidShake**, both baselines cannot capture the number and the color of the rigid cubes (Figure 5). And in **GranularPush**, both baselines fail to capture the distributions of the granular materials. NeRF-dy performs much worse on extrapolation scenes compared to in-distribution scenes, suggesting that capturing 3D information in an explicit way, as opposed to implicit, is much better at capturing the structure of the underlying environment, thus leading to better generalization.

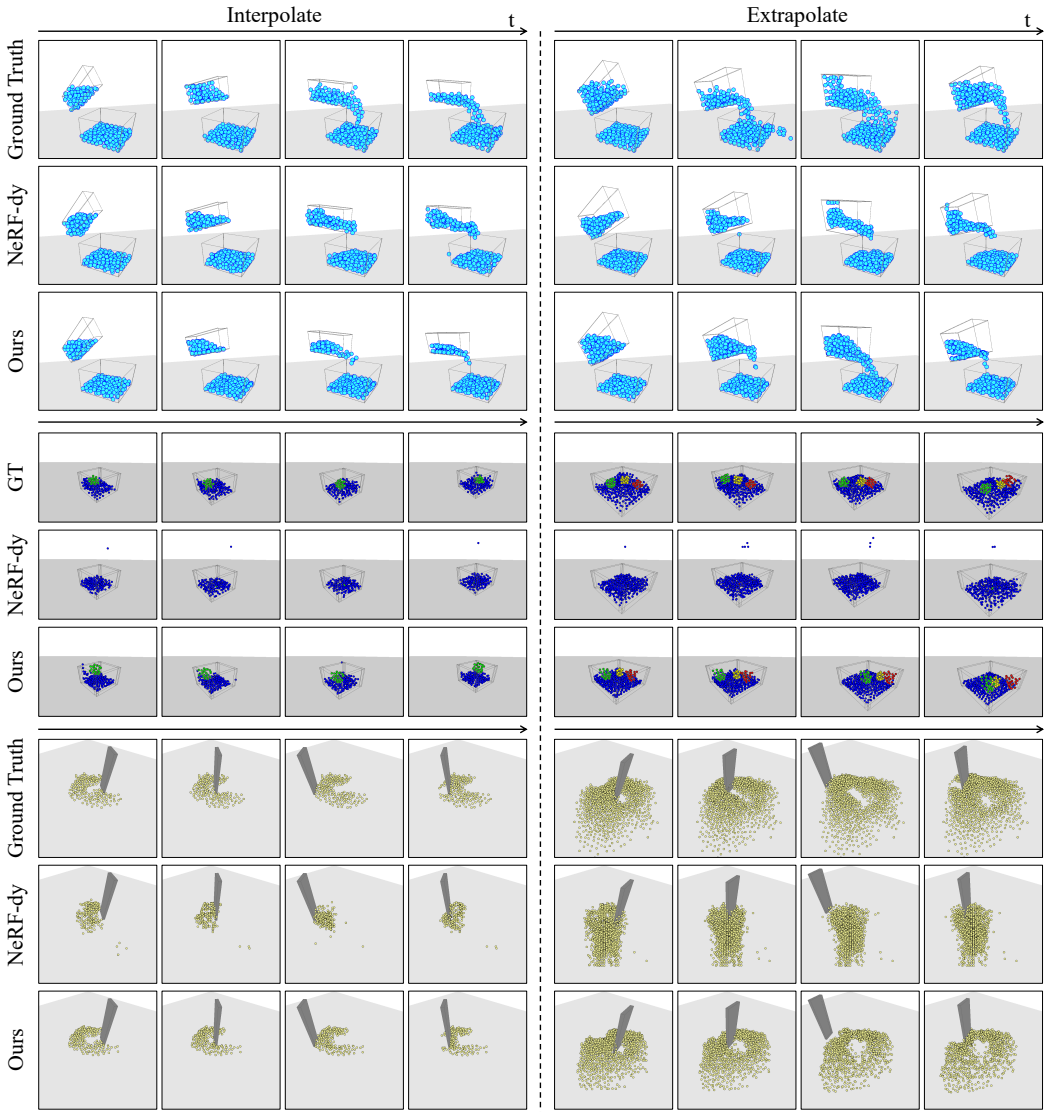


Figure 7: **Qualitative Results of the Dynamics Module on Future Prediction.** Here we visualize our model's predicted future evolution of the particle set as compared with the NeRF-dy [33] baseline in both interpolate and extrapolate settings. Our method correctly identifies the shape/distribution of the fluids, rigid objects, and granular pieces with much better accuracy than NeRF-dy. The future evolution predicted by our method also matches the ground truth much better and produces reasonable results even in extrapolate settings.

5 Conclusions

In this work, we propose a 3-D aware and compositional framework, 3D-IntPhys, to learn intuitive physics from unlabeled visual inputs. Our framework can work on complex scenes involving fluid, rigid objects, and granular materials, and generalize to unseen scenes with containers of different sizes, more objects, or larger quantities of fluids and granular pieces. We show the proposed model outperforms baselines by a large margin, highlighting the importance of learning dynamics models in an explicit 3D representations space. Our current system has two main limitations: (1) our model segments and identifies rigid objects with the strong assumption that objects in the scene have distinctive colors and the color is associated with the material type. To handle real-world scenes, an exciting future direction is to learn the segmentation and material properties from the data. (2) Our model assumes the liquid is not transparent, which only considers a small subset of liquid we encounter in the real world (e.g., coke, orange juice). The problem can be addressed with recent advance in neural rendering that captures transparent texture [61].

References

[1] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

[2] A. Ajay, M. Bauzá, J. Wu, N. Fazeli, J. B. Tenenbaum, A. Rodriguez, and L. P. Kaelbling. Combining physical simulators and object-based networks for control. *CoRR*, abs/1904.06580, 2019.

[3] K. R. Allen, T. Lopez-Guevara, K. L. Stachenfeld, A. Sanchez-Gonzalez, P. W. Battaglia, J. B. Hamrick, and T. Pfaff. Physical design using differentiable learned simulators. *CoRR*, abs/2202.00728, 2022.

[4] M. Babaeizadeh, M. T. Saffar, S. Nair, S. Levine, C. Finn, and D. Erhan. Fitvid: Overfitting in pixel-level video prediction. *CoRR*, abs/2106.13195, 2021.

[5] R. Baillargeon, E. S. Spelke, and S. Wasserman. Object permanence in five-month-old infants. *Cognition*, 20:191–208, 1985.

[6] C. Bates, I. Yildirim, J. B. Tenenbaum, and P. W. Battaglia. Modeling human intuitions about liquid flow with particle-based simulation. *CoRR*, abs/1809.01524, 2018.

[7] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

[8] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110:18327 – 18332, 2013.

[9] D. M. Bear, E. Wang, D. Mrowca, F. J. Binder, H.-Y. F. Tung, R. Pramod, C. Holdaway, S. Tao, K. Smith, L. Fei-Fei, et al. Physion: Evaluating physical prediction from vision in humans and machines. *arXiv preprint arXiv:2106.08261*, 2021.

[10] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

[11] S. Carey and F. Xu. Infants' knowledge of objects: beyond object files and object tracking. *Cognition*, 80(1):179–213, 2001. Objects and Attention.

[12] M. B. Chang, T. D. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *CoRR*, abs/1612.00341, 2016.

[13] F. de Avila Belbute-Peres, T. D. Economon, and J. Z. Kolter. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. *CoRR*, abs/2007.04439, 2020.

[14] D. Ding, F. Hill, A. Santoro, and M. M. Botvinick. Object-based attention for spatio-temporal reasoning: Outperforming neuro-symbolic models with flexible distributed architectures. *CoRR*, abs/2012.08508, 2020.

[15] D. Driess, J.-S. Ha, M. Toussaint, and R. Tedrake. Learning models as functionals of signed-distance fields for manipulation planning. In *Conference on Robot Learning*, pages 245–255. PMLR, 2022.

[16] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint. Learning multi-object dynamics with compositional neural radiance fields. *arXiv preprint arXiv:2202.11855*, 2022.

[17] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.

[18] S. A. Eslami, D. Jimenez 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.

[19] C. Finn, I. J. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, abs/1605.07157, 2016.

[20] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[21] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[22] R. Girdhar, L. Gustafson, A. Adcock, and L. van der Maaten. Forward prediction for physical reasoning. *CoRR*, abs/2006.10734, 2020.

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

[24] 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.

[25] D. Hafner, T. P. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *CoRR*, abs/1912.01603, 2019.

[26] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. *CoRR*, abs/1906.08253, 2019.

[27] M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu. Reasoning about physical interactions with object-oriented prediction and planning. In *International Conference on Learning Representations*, 2019.

[28] T. Kipf, E. van der Pol, and M. Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

[29] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine. Stochastic adversarial video prediction. *CoRR*, abs/1804.01523, 2018.

[30] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. *CoRR*, abs/1603.01312, 2016.

[31] 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.

[32] W. Li, S. Azimi, A. Leonardis, and M. Fritz. To fall or not to fall: A visual approach to physical stability prediction. *CoRR*, abs/1604.00066, 2016.

[33] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba. 3d neural scene representations for visuomotor control. *arXiv preprint arXiv:2107.04004*, 2021.

[34] Y. Li, T. Lin, K. Yi, D. Bear, D. L. Yamins, J. Wu, J. B. Tenenbaum, and A. Torralba. Visual grounding of learned physical models. In *International Conference on Machine Learning*, 2020.

[35] 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.

[36] 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 *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019.

[37] X. Lin, Y. Wang, Z. Huang, and D. Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, 2021.

[38] 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.

[39] L. Manuelli, Y. Li, P. Florence, and R. Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. *arXiv preprint arXiv:2009.05085*, 2020.

[40] 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.

[41] D. Mrowca, C. Zhuang, E. Wang, N. Haber, L. Fei-Fei, J. B. Tenenbaum, and D. L. K. Yamins. Flexible neural representation for physics prediction. *CoRR*, abs/1806.08047, 2018.

[42] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

[43] H. Qi, X. Wang, D. Pathak, Y. Ma, and J. Malik. Learning long-term visual dynamics with region proposal interaction networks. In *ICLR*, 2021.

[44] R. Riochet, J. Sivic, I. Laptev, and E. Dupoux. Occlusion resistant learning of intuitive physics from videos. *CoRR*, abs/2005.00069, 2020.

[45] A. N. Sanborn, V. K. Mansinghka, and T. L. Griffiths. Reconciling intuitive physics and newtonian mechanics for colliding objects. *Psychological review*, 120 2:411–37, 2013.

[46] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[47] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. A. Riedmiller, R. Hadsell, and P. W. Battaglia. Graph networks as learnable physics engines for inference and control. *CoRR*, abs/1806.01242, 2018.

[48] 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.

[49] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. *arXiv preprint arXiv:2205.02909*, 2022.

[50] K. Smith, L. Mei, S. Yao, J. Wu, E. Spelke, J. Tenenbaum, and T. Ullman. Modeling expectation violation in intuitive physics with coarse probabilistic object representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[51] E. S. Spelke. Principles of object perception. *Cognitive Science*, 14(1):29–56, 1990.

[52] H. Suh and R. Tedrake. The surprising effectiveness of linear models for visual foresight in object pile manipulation. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 347–363. Springer, 2020.

[53] A. Tacchetti, H. F. Song, P. A. M. Mediano, V. F. Zambaldi, N. C. Rabinowitz, T. Graepel, M. M. Botvinick, and P. W. Battaglia. Relational forward models for multi-agent learning. *CoRR*, abs/1809.11044, 2018.

[54] 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.

[55] T. Ullman, E. Kosoy, I. Yildirim, A. A. Soltani, M. H. Siegel, J. Tenenbaum, and E. S. Spelke. Draping an elephant: Uncovering children's reasoning about cloth-covered objects. In *Proceedings of the 41st Annual Conference of the Cognitive Science Society*, pages 3008–3014, 2019.

[56] B. Ummenhofer, L. Prantl, N. Thuerey, and V. Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2019.

[57] R. Veerapaneni, J. D. Co-Reyes, M. Chang, M. Janner, C. Finn, J. Wu, J. B. Tenenbaum, and S. Levine. Entity abstraction in visual model-based reinforcement learning. *CoRR*, abs/1910.12827, 2019.

[58] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating the future by watching unlabeled video. *CoRR*, abs/1504.08023, 2015.

[59] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28, 2015.

[60] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video. *Advances in neural information processing systems*, 30, 2017.

[61] S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, and S. Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8534–8543, 2021.

[62] B. Wu, S. Nair, R. Martín-Martín, L. Fei-Fei, and C. Finn. Greedy hierarchical variational autoencoders for large-scale video prediction. *CoRR*, abs/2103.04174, 2021.

[63] Z. Xu, Z. He, J. Wu, and S. Song. Learning 3d dynamic scene representations for robot manipulation. In *Conference on Robotic Learning (CoRL)*, 2020.

[64] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. In *Robotics: Science and Systems (RSS)*, 2019.

[65] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances In Neural Information Processing Systems*, 2016.

[66] Y. Ye, D. Gandhi, A. Gupta, and S. Tulsiani. Object-centric forward modeling for model predictive control. In *CoRL*, 2019.

[67] Y. Ye, M. Singh, A. Gupta, and S. Tulsiani. Compositional video prediction. In *International Conference on Computer Vision (ICCV)*, 2019.

[68] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[69] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. J. Johnson, and S. Levine. SOLAR: deep structured latent representations for model-based reinforcement learning. *CoRR*, abs/1808.09105, 2018.

[70] R. Zhang, J. Wu, C. Zhang, W. T. Freeman, and J. B. Tenenbaum. A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding. *CoRR*, abs/1605.01138, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] . We have included limitations of the proposed work in Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We have included the discuss in the supplementary material.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see the paragraph "implementation details" in Section 4 and the supplementary material for the details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] . We have illustrated standard deviation in Figure 6.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The details are provided in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]