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

Anonymous Author(s)
Affiliation
Address
email

1 Additional Results

To better understand the performance of our framework visually, we prepare test time rollouts of our framework as well as those of various baselines in the supplementary video. The video is published anonymously and can be accessed in https://sites.google.com/view/3d-intphys

1.1 Ablation Study

We find 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 the issue, we introduce the spacing loss to penalize the distance between these points. We set the threshold of penalty d_{min} to be 0.08 and the loss weight σ to be 10. We find that spacing loss can help improve the performance of the dynamics learner especially under extrapolate settings, as shown in Figure 1. We provide qualitative results in the supplementary video.

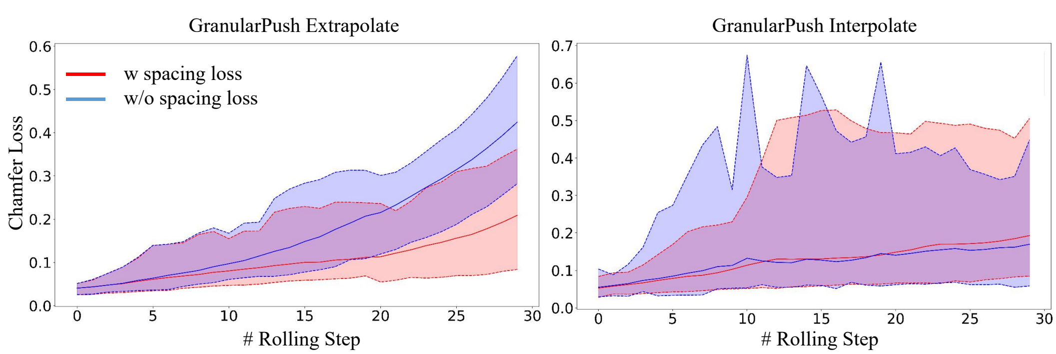

Figure 1: **Ablation Study on the Spacing Loss.** Training dynamics models in the GranularPush scenario with spacing loss results in better rolling prediction. The performance gap is even more substantial in the extrapolate setting.

2 Implementation Details

2.1 Dataset Generation

Our datasets are generated by the NVIDIA Flex simulator. Each of the three scenarios (Pour, Shake and Push) has 500 videos of trajectories taken from 6 views, with each trajectory consisting of 300

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

	X-Range	Y-Range	Z-Range
FluidPour	[-29.11, -12.66]	[42.00, 60.00]	[-7.78, 7.78]
FluidCubeShake	[-3.25, 42.25]	[19.25, 19.25]	[-24.50, 24.00]

Table 1: **Robot Action Space(centimeters):** we show the range the robot arms can move in the FluidPour and FluidCubeShake environments.

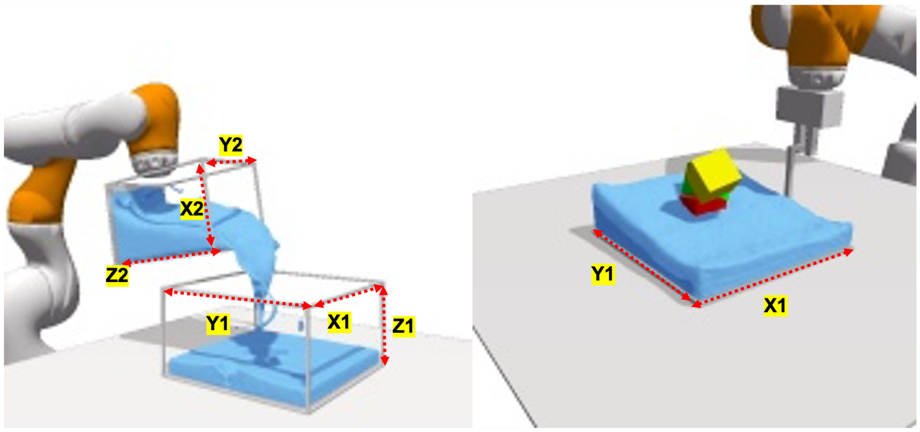

Figure 2: **Illustration of the Environment Settings.** In the FluidPour scenario, a robot arm holds a container and tries to pour some fluid into another container. In the FluidShake scenario, a robot moves a container with some fluid and cubes. We show the parameters for the container shape referred in Table 2.

frames. We manually select the 6 views with reasonable coverage of the tabletop space to minimize the occlusion. The 500 trials are generated from five different sets of environmental parameters, detailed in Table 2. We take one set of parameters that are outside the training distribution as the **extrapolate** dataset for evaluating model generalization. For the rest of the four settings, we randomly split them into train and test sets with a ratio of 0.8.

Next, we provide more details for each scenario:

- In the FluidPour environment, we randomly initialize the position of the upper container and then generate random back-and-forth actions by tilting the container. The action space is then the position and tilting angle of the upper container.
- In FluidCubeShake, we also randomly initialize the position of the container and the cubes inside the container. We then generate random but smooth action sequences moving the container in the 2D plane. The action space is then the x-y location of the container.
- In GranularPush, we randomly initialize the position of the granular pile. Then, for each push, we randomly generate the starting and ending positions of the pusher and move the pusher along the straight line with an angle perpendicular to the pushing direction. The action space is a four-number tuple stating the starting and ending position on the 2D plane.

The following table shows the moving range of the robot arms in the FluidPour and FluidCubeShake environments after normalizing the robot into a size that is the same as in the real world (unit: centimeters). For GranularPush, the pusher is moving over the entire table; we ignore the specific number in this environment as we do not have robot arms as a reference.

Additional dataset samples. We show samples from the FluidPour, FluidCubeShake and GranularPush dataset in Figure 3, 4 and 5, respectively. Note that all trajectories for the extrapolate settings are used only for testing and will not show up during the training process. We include more samples from the dataset in the video format in the supplementary video.

SceneName	Params	Env1	Env2	Env3	Env4	Extrapolate
FluidPour	X2	0.53	0.53	0.81	0.81	0.81
	Y2	0.53	0.81	0.53	0.81	0.81
	Z2	1.24	1.24	1.24	1.24	1.24
	X1	1.35	1.35	1.35	1.35	1.35
	Y1	1.35	1.35	1.35	1.35	1.35
	Z1	0.74	0.74	0.74	0.74	0.74
	AmountofWater	5125	5125	6125	5375	7625
FluidCubeShake	X1	0.88	0.88	1.32	1.32	1.32
	Y1	0.88	1.32	0.88	1.32	1.32
	CubeNumber	1	1	2	2	3
	Water	2173	3322	3322	4858	4983
GranularPush	GranularNumber	2197	4032	5832	9261	12167

Table 2: **Scene Parameters for Generating the Interpolate and Extrapolate Datasets.** We generate the datasets by varying the shape of container, amount of water, number of cubes, and quantity of the granular material. Z_i, X_i, Y_i are the height, width, and depth for a container i. Please refer to Figure 2 for more details.

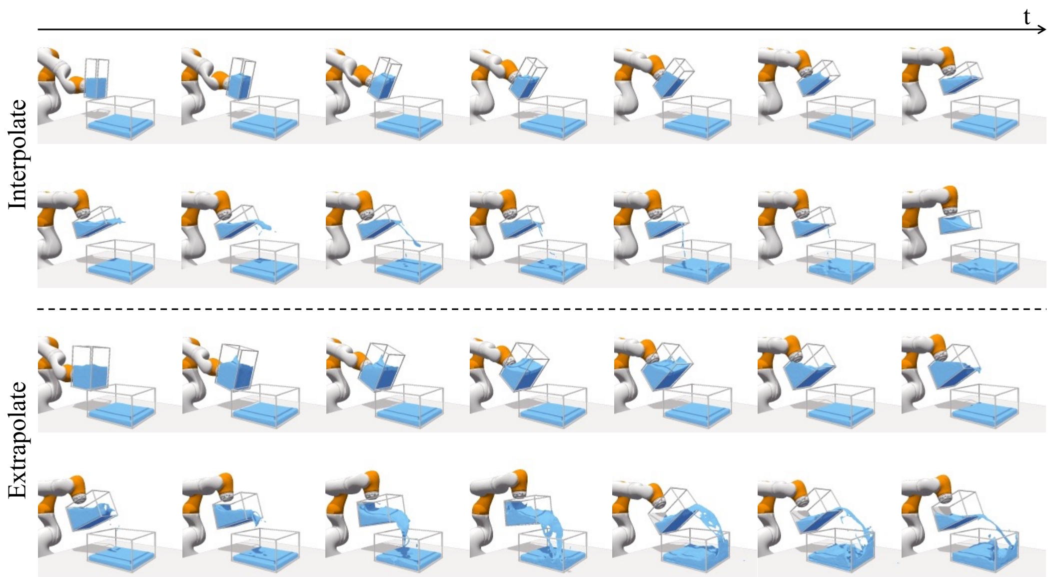

Figure 3: **Samples from FluidPour Dataset.** We show sequences of frames over time with an interval of 20 frames. The sequences above the dashed line are for **interpolate** data, and the bottom images illustrate the **extrapolate** data.

2.2 Model Architecture

Image-conditional NeRF. We follow the architectural design by [68]. For the feature encoder, we employ a ResNet-34 backbone to extract features. We use the output layers prior to the first four pooling layers, upsampling them using bilinear interpolation to the same size, and then concatenating these four feature maps. We initialize the weight of the feature extractor of the scene using ImageNet pre-trained weight. For the NeRF function f, We use fully-connected ResNet architecture with 5 ResNet blocks with a width of 512.

Dynamics predictor. For the edge and vertice encoders, Q_e and Q_v, we use 3-layer fully-connected networks activated by the ReLU function with 150 hidden units. For the propagators, P_e and P_v, we use a 1-layer fully-connected network followed by ReLU activation. The output dimension of the linear layer is 150.

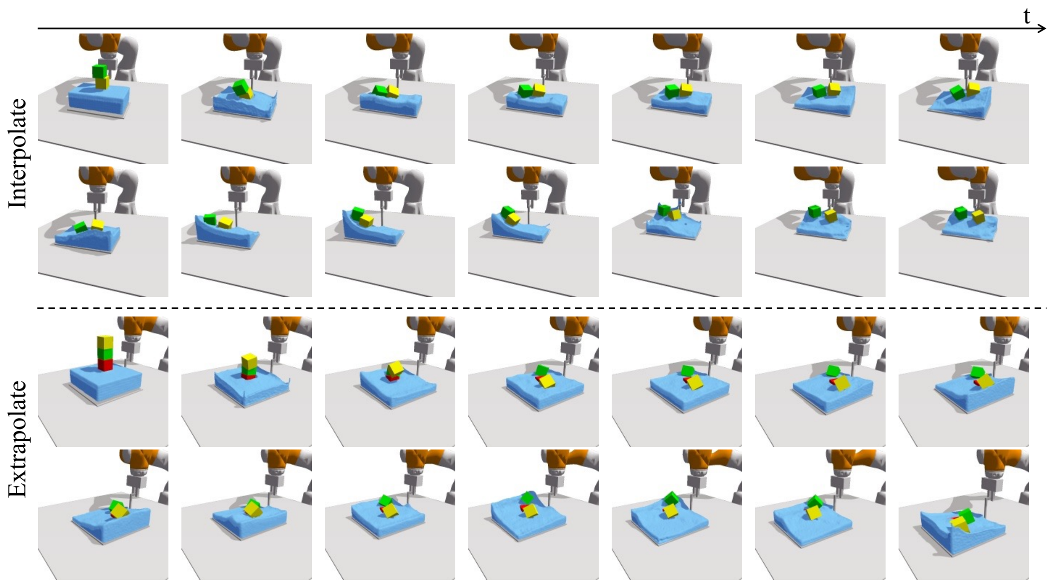

Figure 4: **Samples from FluidCubeShake Dataset.** We show sequences of frames over time with an interval of 20 frames. The sequences above the dashed line are for **interpolate** data, and the bottom images illustrate the **extrapolate** data.

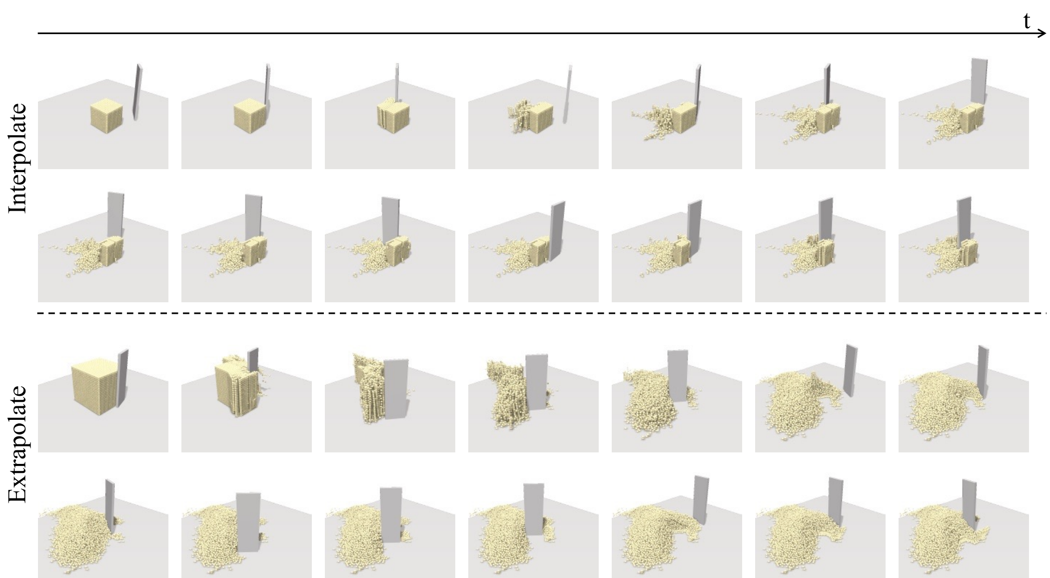

Figure 5: **Samples from GranularPush Dataset.** We show sequences of frames over time with an interval of 20 frames. The sequences above the dashed line are for **interpolate** data, and the bottom images illustrate the **extrapolate** data.

Sampling 3D points from the trained visual perception module. We sample points on a $40\times40\times40$ grid from an area of 55cm $\times$ $55cm$ $\times$ $55cm$ and 63cm $\times$ $63cm$ $\times$ $63cm$ at the center of the table for FluidPour and FluidCubeShake respectively, and on a $70 \times 70 \times 70$ grid from an area of 6cm $\times$ $6cm$ $\times$ $6cm$ for GranularPush. We evaluate and include points with a density (measured by the occupancy in the predicted neural radiance fields) larger than 0.99. To reduce the total number of points, we subsample the inferred points with FPS with a ratio of 5% for FluidPour and 10% for FluidCubeShake and GranularPush.

Graph building. We set the neighbour distance threshold δ to be $0.2, 0.15, 0.15$ for FluidPour, FluidCubeShake and GranularPush respectively. We select the threshold so that each point will have on average 20 30 neighbors. Since, in FluidPour, we sample the points with lower density 2000points$/m^2$, we use a larger threshold for this scenario. For FluidShape and GranularPush, since the density is around 3000 points$/m^2$, we cut down the number by 25%.

We found that if the threshold is too small, the performance will degrade significantly since each particle will only receive messages from a few neighbors (and miss out on the larger context). On the other hand, setting the threshold too large will cause the training time to increase since the graph will have more edges. We found that setting the threshold around the right scale generally leads to more effective training of a reasonable dynamics network.

2.3 Training Details

The models are implemented in PyTorch. We train the perception module using Adam optimizer with a learning rate of $1e{-}4$, and we reduce the learning rate by 80% when the performance on the validation set has stopped improving for 3 epochs. To compute the rendering loss when training the perception module, we sample 64 points through each ray in the scene and set the ray-batch size of the NeRF query function f to be 1024×32. Training the perception module on a single scenario takes around 5 hours on one RTX-3090.

We train the dynamics simulator using Adam optimizer with a learning rate of $1e{-}4$, and we reduce the learning rate by 80% when the performance on the validation set has stopped improving for 3 epochs. The batch size is set to 4. We train the model for 20, 30, and 40 epochs for FluidPour, FluidCubeShake, and GranularPush, respectively. It takes around $10 \sim 15$ hours to train the dynamics model in one environment on one single RTX-3090.

2.4 Graph-Based Dynamics Model without Particle-level Correspondence

The velocity of an object provides critical information on how the object will move in the future, yet, we do not have access to such information when tracking the object is impossible. As described in Section 3.2, the attributes a_i^v of a vertex v_i in the built graph consists of (1) velocity of this point in the past frames and (2) attributes of the point (rigid, fluid, granular). To get the velocity of a vertex v, we should have the history position of this vertex. However, since the point clouds are inferred from each frame independently, we do not know how each point moves over time since we do not have point correspondence between frames.

To address the problem, we leverage the fact that some objects in the scene are easier to track, and we try to use the motion of these trackable objects to infer motion for the untrackable units. We assume that we know the dense-labeled states of some known fully-actuated shapes like desks and cups connected to the robot arms. Here we will list one specific scenario where a cup of water is poured into another cup. In this case, we have two different types of points: points for fluid and points for cups, we name the states of them in time step t as $V_P^t = \{v_{P,i}^t\}$ and $V_S^t = \{v_{S,i}^t\}$ respectively. For the particle encoder Q_v, if the particle belongs to the cups, then the input of particle encoder contains n_s history states before $t_0 : \{V_S^{(t_0-n_s):t_0}\}$. If the particle belongs to the water, then we have no history states, so the input of Q_v is all-zero.

By adding the relative position between receiver and sender points, we can pass the momentum of V_P to V_S. Compared with human intuition, we can get an intuitive prediction of the movement of water by simply knowing the past movement of the cup without knowing the past movement of water.

Following [47], we use the velocity of points and their relative position as inputs to the dynamics module instead of using the absolute positions of the points. This ensures the model is translation-invariant so the learned dynamics model can be shared across different spatial locations.

2.5 Inference Speed of Our Model

The prediction speed of the dynamics module depends on the number of input particles, and it takes around 0.1s for graphs with around 300 nodes in FluidShake and FluidPour, and around 0.2s for scenes with 700+ nodes in GranularPush.

For our visual module, the main time consumption comes from NeRF sampling, it takes 0.2s to sample from a grid space introduced in the experiment section of our paper, this was run in blocks, with block-size=1000, made up 4G of a V100 GPU. And it can be even faster with larger blocks. The sub-sampling process (FPS, segmentation) is fast since they are all written in parallel versions, which takes less than 5ms.

3 Potential Society Impact

Our work shows the possibility of learning dynamics models from raw sensory inputs, opening up opportunities to automate the design of differentiable physics engines through data-driven learning algorithms. The resulting system can potentially benefit many downstream tasks, including general scene understanding, robotics manipulation, the construction of 3D generative models, and inverse tasks like planning/control and inverse design. Furthermore, predictions from our model are highly interpretable, which makes it straightforward to explain model behaviors and re-purpose the outputs for other downstream applications.

Though data-driven approaches are potentially more scalable with enough data, concerns still exist that it might be hard to ensure the robustness of the model under sensor noise and adversarial attacks. It also becomes less clear how to fully mitigate data biases. Therefore, bringing in advanced techniques from ML robustness will be one critical future avenue to pursue.

4 Some Discussions

Q: *What is the novelty of the proposed framework?*

The proposed work aims to tackle the challenging problems of learning visual dynamics from raw images, which neither pixel-NeRF nor graph-based dynamics models alone can solve.

Simply combining the two methods, unfortunately, does not provide a valid solution to the problem since existing point-based dynamics models need to learn from strong supervision provided by 3D ground truth point trajectories, which are hard to obtain in most real setups. For example, in our water experiments, it is impossible for any existing tracking method to successfully track each water particle. To tackle the problem, we propose several new techniques to facilitate dynamics learning without dense correspondence, including momentum passing from containers to fluids and new training loss (e.g., Chamfer distance loss and spacing loss). They allow more robust learning of dynamics models on raw point clouds sampled from the learned occupancy field (instead of the original simulator).

Q: *Is the color segmentation of the fluid objects a reasonable assumption?*

It should be noted that the color-based segmentation will not degrade the challenging problem of learning 3D Intuitive Physics, since the task focuses more on learning complex visual dynamics from images.

We want to emphasize that the work focuses more on learning complex visual dynamics from images, as opposed to solving object segmentation in general. Learning fluids dynamics from videos is a challenging task, and there are only a few existing works. NeRF-dy is the closest to us, yet the model's generalization ability is limited. We have shown in the proposed work that we can significantly improve the generalization ability by operating with a hybrid of implicit and explicit, as opposed to pure implicit, 3D representations. We agree object segmentation is a critical visual understanding problem, and solving it is an important next step to getting a more general visual dynamics learning framework.

With recent advancements such as SAM [29] and SEER [71], which focus on segmentation in real-world scenarios, the possibility of video segmentation without the need for annotations has emerged (as is shown in Figure 6). This development paves the way for leveraging existing large-scale models to enhance the segmentation pipeline, offering great promise for future applications.

Q: *Since the fluid has zero velocitys, how to predict the intuitive dynamics?*

Figure 6: **SAM Working on FluidCube Shake:** Recent large segmentation models can well generate masks for different objects in the scene.

The intuition is that we can infer the water movement from the container's movement. We also assume that the initial velocity of water is **nearly zero**, which is also used in [50], so the momentum can be gradually passed from the container to the water.

We propose this assumption so that the intuitive physics model can be learned from (1) particles sampled from the neural radiance field, which is not stable (2) point clouds without one-to-one correspondence. The results show that we can learn reasonable dynamics (water poured out from a cup, water falling in the container, cubes moving in water, and granular materials pushed away by a pusher). It also shows the potential of distribution-based loss in learning visual dynamics.

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. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

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

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

[32] 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.

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

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

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

[53] 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.

[54] 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.

[55] 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.

[56] 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.

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

[58] 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.

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

[60] 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.

[61] 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.

[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.

[71] X. Zou, J. Yang, H. Zhang, F. Li, L. Li, J. Gao, and Y. J. Lee. Segment everything everywhere all at once. *arXiv preprint arXiv:2304.06718*, 2023.