Learning to Simulate Unseen Physical Systems with Graph Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recently there is an increasing interest in learning to simulate the dynamics of physic systems via machine learning. However, existing approaches fail to generalize to physical substances not in the training set, such as liquids with different viscosities or elastomers with different elasticities. Here we present a machine learning method embedded with physical priors and material parameters, which we term as "Graph-based Physics Engine" (GPE), to efficiently model the physical dynamics of different substances in a wide variety of challenging scenarios. We demonstrate that GPE can generalize to different material properties not seen in the training set by simply modifying the physical parameters, and also performs well from single-step predictions to multi-step roll-out simulations. GPE provides new insights into the construction of learnable simulators and is a key step toward predicting unknown physics problems in the real world.

1 Introduction

Simulation of the dynamic of unknown physics systems is crucial in many scientific and engineering areas [1–6]. Building high precision physics simulators requires extensive domain knowledge and substantial engineering effort. Also the approximation techniques used to ensure perceptual realism make the simulations deviate from the laws of reality in the long term. Recently, several neural network based simulators were proposed to learn physical dynamics from large-scale training data [7–14]. Their success prove the feasibility of using machine learning to improve physics simulation.

One common drawback of the aforementioned methods is the lack of generalization for unseen physical processes and substances. For example, if the training dataset consist dynamics of softer elastomers and harder elastomers, it is expected that the model can generalize to elastomers with other levels of elasticity. However, most ML-based simulators fail to do so since they can only simulate the physic systems in the training dataset.

To this end, we introduce a new machine learning approach to model physical interaction dynamics with graphs, which we term "Graph-based Physics Engine" (GPE). Our network represents discrete physical systems by a graph structure which models force interactions using a message passing neural network (MPNN) [15]. The contribution of GPE is multifaceted. First, the physical parameters such as elasticity of elastomers are introduced to enable the model to generalize across different systems. Second, a unified model architecture is used to learn different types of physical dynamics, from viscous fluid simulations to finite element mechanics (FEM) analysis. Additionally, we introduce the law of momentum conservation in MPNN to increase stability of learning. Various experiment results show that GPE has good long-term roll-out stability even if it is learned over single-step loss.

Submitted to NeurIPS 2021 AI for Science Workshop.

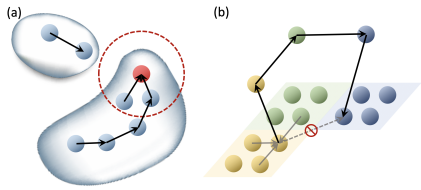

Figure 1: Graph topology for unstructured and structured physical systems. (a) Nearest neighbor graph for unstructured systems. (b) Multi-scale grid graph for structured systems.

1.1 Related Work

Learning physical dynamics from data is one of the most important research areas in physics [16]. A variety of neural network architectures, such as RNN [17], CNN [13] and GNN [7, 9, 11, 12], have been adopted to simulate dynamics of physical systems. In particular, Ummenhofer et al. [13] developed a novel CNN based on smoothed-particle hydrodynamics (SPH) theory to simulate the motion of a Lagrangian fluid. Battaglia et al. [7, 18] and Sanchez-Gonzalez et al. [11, 14, 19] demonstrated the effectiveness and generality to model the dynamics by message passing. On the other hand, Mrowca et al. [10] and Li et al. [9, 12] used hierarchical graph neural networks to simulate non-rigid deformation and collision deformation of structured physical systems such as elastomers. However, all the aforementioned work can only simulate the systems seen in the training set, which inspires us to allow the learned model to simulate unseen materials.

2 Model Framework

Assume $(G^0, G^1, ..., G^T)$ is the evolutionary trajectory of a physical system over time, where the directed graph $G_t =< O_t, R_t >\in \mathcal{G}$ represents the state of the system at time t. Vertices $O = \{o_i\}$ represent the collection of discrete units of objects, and edges $R = \{(o_i, o_j)\}$ represent the (directed) interaction relationship between the particles.

A learnable graph-based physics engine (GPE) $s_\theta : \mathcal{G} \to \mathcal{G}$ models physical dynamics by mapping the graph representation G^t to the graph representation at the next time step G^{t+1}. Similar to [14], we use message passing neural networks (MPNN) [15] to model the evolutionary dynamics of physical systems as follows.

$$\text{Encoder}: \quad h_i^0 = \text{Enc}_v(\text{features}(o_i)), \quad \forall o_i \in O \tag{1}$$

$$h_{i,j}^0 = \text{Enc}_e(\text{features}(o_i, o_j)), \quad \forall (o_i, o_j) \in R \tag{2}$$

$$\text{For} \quad l = 1, ..., L: \quad m_{i,j}^l = \text{Proc}_{e,l}(h_i^{l-1}, h_j^{l-1}, h_{i,j}^{l-1}), \quad \forall (o_i, o_j) \in R \tag{3}$$

$$m_{j,i}^l = -m_{i,j}^l, \quad \forall (o_i, o_j) \in R \tag{4}$$

$$h_i^l = h_i^{l-1} + \text{Proc}_{v,l}(\sum_{j \in \mathcal{N}_i} m_{i,j}^l) \quad \forall o_i \in O \tag{5}$$

$$h_{ij}^l = h_{i,j}^{l-1} + m_{i,j}^l \quad \forall (o_i, o_j) \in R \tag{6}$$

$$\text{Decoder}: \quad x_i = \text{Dec}_v(h_i^L), \quad \forall o_i \in O \tag{7}$$

The encoders map the raw features of the nodes and the edges into hidden vectors. The processors iteratively compute messages on the edges, aggregate messages on nodes and update node hidden vectors. Finally, the decoder predicts the position of the nodes at the next step. The model is trained by minimizing the mean-squared-error (MSE) between predicted positions and the ground truth.

2.1 Graph Representation of a Physical System

The topology of the graph depends on how the physical system is being discretized. For unstructured systems such as liquids, snow or sand, a nearest neighbor graph is used to model the connectivity between particles, where two particles o_i and o_j are connected if their distance is within some

Methods	Sand	Snow	Elastomer	FEM	Viscous Fluid
DPI-Net[9]	5.15	4.91	2.07	1.76	3.31
GNS[14]	3.80	5.49	3.58	5.03	3.78
ConvNet[13]	3.51	5.24	2.82	4.96	**2.75**
GPE	**2.37**	**3.94**	**1.35**	**1.27**	3.22

Table 1: The average MSE ($\times 10^{-3}$) for roll-out simulations on unseen materials.

threshold r. The nearest neighbor topology reflects the fact that the strength of interaction between two particles vanishes as their distance increases. For structured systems such as elastomers, we use a mesh to discretize the system. In order to take consideration of the long-range interaction during the deformation of solids, we use a multi-scale grid graph topology similar to [9, 10]. We merge four neighboring nodes into one "macro-mesh" described as a virtual node, which is connected to all the four lower-layer nodes and other virtual nodes. The merging process continues until all the nodes are merged into one super node. These virtual nodes and edges serve to describe the macro structure and build a high-speed path for force transmission. Notice that the connectivity of nearest neighbor graph needs to be updated during every roll-out step whereas the multi-scale grid graph does not.

Input feature. The input features on GPE nodes contain the material properties of each unit (such as the viscosity of water, the friction angle of sand, the hardening coefficient of snow, the Young's modulus of an elastomer), the velocity of the unit, and the external forces applied such as gravity. Position information is not included in the input feature, which makes GPE naturally satisfy the translation and permutation invariance.

Momentum conservation. Either the nearest neighbor graph or multi-scale grid graph contains bi-directional edges between connected nodes. We only calculate the message $m_{i,j}$ on one of the edges, and introduce the law of momentum conservation by simply requiring the message on the opposite edge to be $m_{j,i} = -m_{i,j}$.

3 Experiments

3.1 Physical Domains and Datasets

We evaluate the performance of GPE in several different physical domains containing different types of materials and dynamics. The SAND, SNOW and ELASTOMER domains generated by Taichi-MPM engine [20, 21] contain dynamics of sand, snow and elstomers with different physical parameters hitting a wall then deforming. The VISCOUS FLUID domain generated by FLIPViscosity3D engine [22, 23] contains dynamics of fluids with different levels of viscousity. The FINITE ELEMENT MECHANICAL (FEM) domain contains dynamics of elastomers with different Young's modulus hitting an uneven obstacle. The domains represent a wide variety of physical systems but can be simulated by a unified model. The details of data generation processes, model architectures and hyperparameters are relegated to Appendix A.

3.2 Results

In Figure 2, we visualize the SAND and FEM dynamics predicted by GPE. We observe that the roll-out trajectories predicted by GPE are very similar to ground truths, even if the models are trained only by single-step error. Moreover, it can be seen that GPE can simulate physical systems with unseen parameters. Visualizations of other domains are relegated to Appendix B.

To quantify the performance of the simulation on unseen materials, we compare GPE with three baselines, DPI-Net with hierarchy [9], GNS [14] and ConvNet [13]. We calculate the average MSE of each time step in roll-out for the aforementioned domains on unseen physical parameters. For GPE and DPI-Net, nearest neighbor graph are used for the SAND, SNOW and VISCOUS FLUID, while multi-scale grid graphs are used for the ELATOMER and FEM. For GNS and ConvNet, the graph topology are consistent with original papers. As shown in Table 1, GPE outperforms the baseline methods in most domains, demonstrating its better ability to generalize to unseen systems.

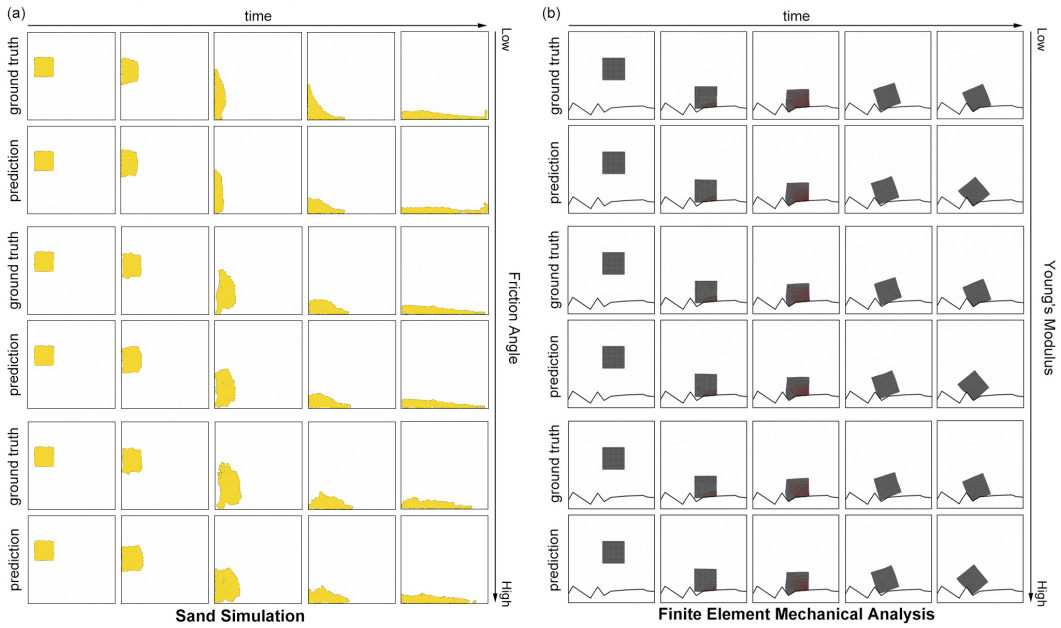

Figure 2: The dynamics of SAND and FEM predicted by GPE. (a) The motion of a sand pile with low, medium and high friction angles. (b) FEM results of collision deformation of an elastomer with different Young's modulus on an irregular obstacle.

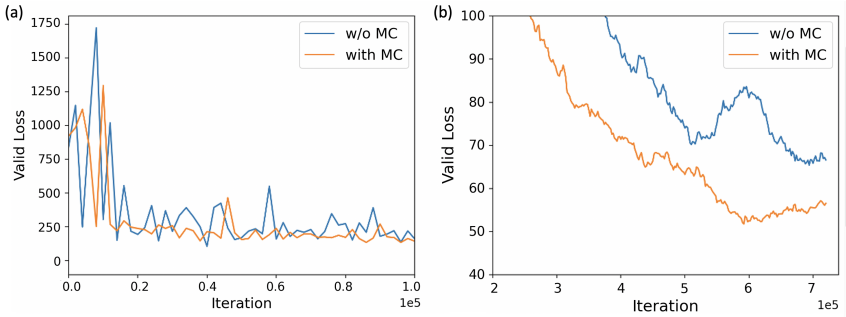

Figure 3: Validation losses of GPE with and without momentum conservation. (a): First 100,000 training iterations. (b): 200,000-700,000 training iterations.

Finally, we demonstrate the effect of momentum conservation on VISCOUS FLUID domain. By introducing momentum conservation, only half of the messages on the edges are computed, leading to a 37% reduction of forward computation time. As shown in Figure 3, GPE with momentum conservation is more stable at the beginning of training, converges faster, and performs better on the test set, compared to GPE without such a constraint.

4 Conclusion

In this paper, we proposed GPE, a graph-based learnable physical engine that simulates the dynamics of complex systems by learning the message passing between nodes. Our experiments show that GPE can simulate complex evolutionary trajectories in a variety of domains, and has good generalization ability to unseen materials. We also introduce momentum conservation in the model to reduce computational cost and stabilize the training process. Thanks to the universality, expressiveness and generalization of modeling, graph-based learnable physics engines show great potential for applications in high precision simulation of complex systems and inverse problem solving. Our study helps shed light on using machine learning for physical reasoning.

References

[1] Ted Belytschko, Yun Yun Lu, and Lei Gu. Element-free galerkin methods. *International journal for numerical methods in engineering*, 37(2):229–256, 1994.

[2] Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1):236–252, 1995. ISSN 0010-4655. doi: https://doi.org/10.1016/0010-4655(94)00170-7. URL `https://www.sciencedirect.com/science/article/pii/0010465594001707`. Particle Simulation Methods.

[3] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, pages 1–52. 2016.

[4] Eftychios Sifakis and Jernej Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *Acm siggraph 2012 courses*, pages 1–50. 2012.

[5] Jiequn Han, Linfeng Zhang, Roberto Car, et al. Deep potential: A general representation of a many-body potential energy surface. *arXiv preprint arXiv:1707.01478*, 2017.

[6] Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*, 2020.

[7] Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*, 2016.

[8] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

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

[10] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel LK Yamins. Flexible neural representation for physics prediction. *arXiv preprint arXiv:1806.08047*, 2018.

[11] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR, 2018.

[12] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019.

[13] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2019.

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

[15] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[16] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, page 9–20, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919998. doi: 10.1145/280814.280816. URL `https://doi.org/10.1145/280814.280816`.

[17] JCS Kadupitiya, Geoffrey C Fox, and Vikram Jadhao. Deep learning based integrators for solving newton's equations with large timesteps. *arXiv preprint arXiv:2004.06493*, 2020.

[18] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[19] Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

[20] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

[21] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):201, 2019.

[22] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics (TOG)*, 26(3):100–es, 2007.

[23] Christopher Batty and Robert Bridson. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. 2008.

A Implementation Details

Here we illustrate the details of the implementation of GPE which are omitted in Section 3.1.

Data generation. The SAND, SNOW and ELASTOMER dataset are generated by Taichi-MPM engine [20, 21]. The VISCOUS FLUID dataset are generated by FLIPViscosity3D engine [22, 23]. We independently sample 2000 trajectories with 10 different physical parameters as training set, 300 trajectories as test set and 300 trajectories as validation set with physical parameters that did not appear in the training set, where each trajectories contain 1000 frames. Most systems contain approximately 2500 particles, while the viscous liquid system contain about 5000 particles. The FEM dataset is obtained from standard simulation formulas. We independently sample 2000 trajectories as training set, 300 trajectories as test set and 300 trajectories as validation set, where each trajectories contain 800 frames and the system approximately 1000 meshes. We train the model on FEM data with Young's modulus from 1 MPa to 4 MPa, and test the model on data with Young's modulus from 0.5 MPa to 10 MPa.

Implementation Details. The encoders Enc_v and Enc_e are MLPs with 2 layers and 128 neurons per layer. The processors Proc_v and Proc_e are MLPS with 2 layers and 128 neurons per layer. In Processor, the number of steps for message passing is 12. The decoder Dec_v is an MLP with 2 layers and 128 neurons per layer. All activation functions are PReLU. For SAND and SNOW domains, the cutoff radius for nearest neighbor graph is 0.4. The model is trained 1000 epochs, using the Adam optimizer and decaying the learning rate from $5 \times 10^{-5} to 1 \times 10^{-6}$. We compare the performance of GPE with baselines GNS [14], DPI-Net [9], and ConvNet [13].

Boundary nodes. In GPE, the boundaries and obstacles are modeled as a special type of nodes, which allow the learned model to generalize to scenarios with more complex boundaries.

B Visualization Results

In Figure 4, the enlarged image of the mechanical finite element analysis shows the distribution of stresses in deformation in more detail.

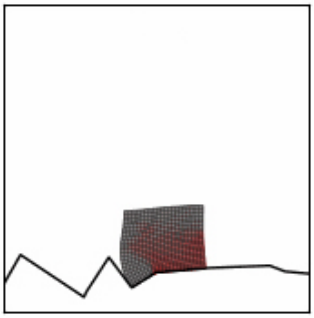

Figure 4: The stress distribution during the deformation of the meshed elastic body is shown in red.

In Figure 5, we present the visualization results of SNOW, ELASTOMER and VISCOUS FLUID predicted by GPE.

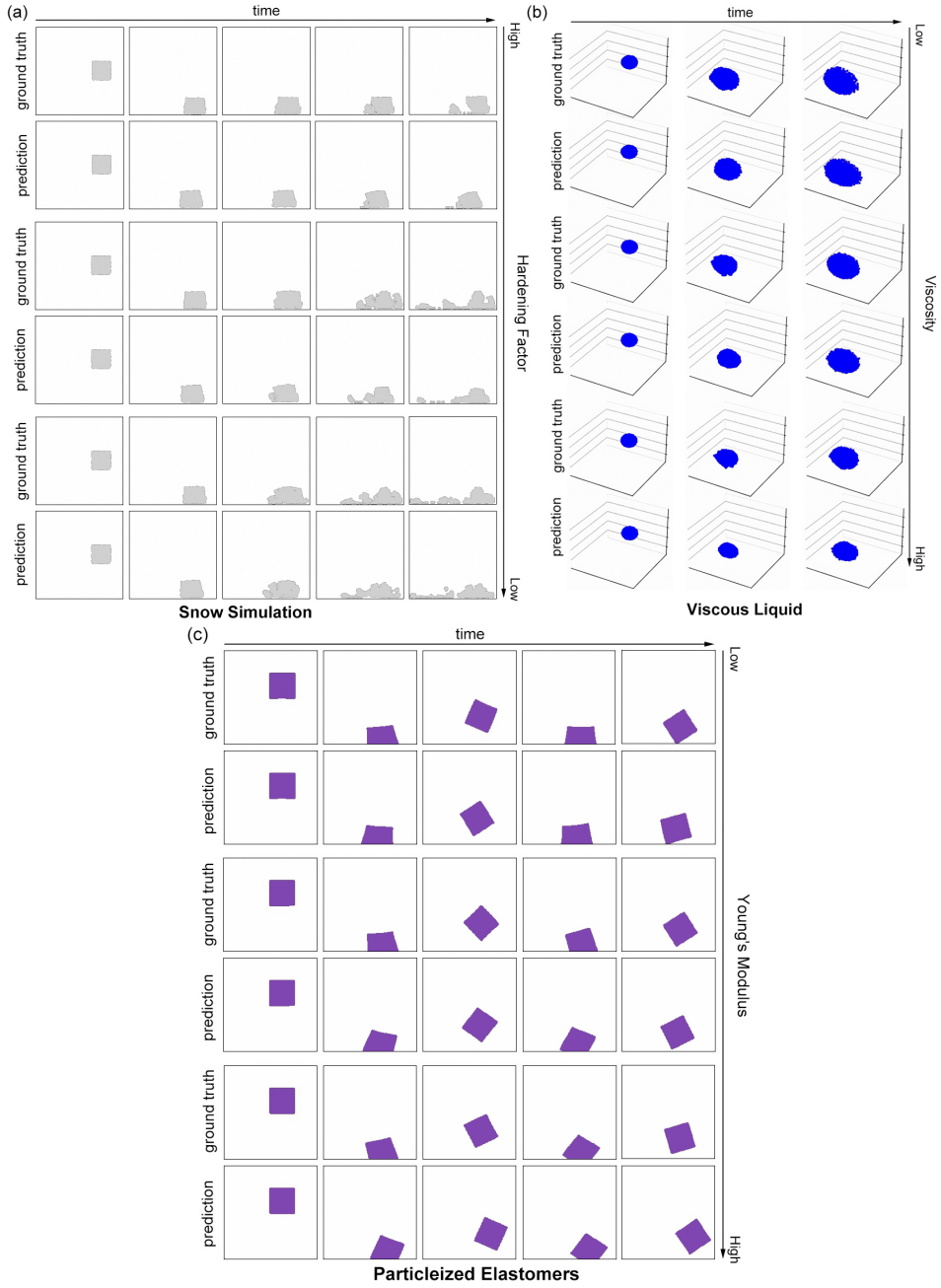

Figure 5: The dynamics of (a) SNOW, (b) VISCOUS LIQUID and (c) PARTICLEIZED ELASTOMER predicted by GPE.