RoboPack: Learning Tactile-Informed Dynamics Models for Dense Packing

Bo Ai[1,2*], Stephen Tian[2*], Haochen Shi[2], Yixuan Wang[3], Cheston Tan[1], Yunzhu Li[3], Jiajun Wu[2]

Abstract*—Tactile feedback is critical for understanding the dynamics of both rigid and deformable objects in many manipulation tasks, such as non-prehensile manipulation and dense packing. We introduce an approach that combines visual and tactile sensing for robotic manipulation by learning a neural, tactile-informed dynamics model. Our proposed framework, RoboPack, employs a recurrent graph neural network to estimate object states, including particles and object-level latent physics information, from historical visuo-tactile observations and to perform future state predictions. This dynamics model, learned from real-world data, can solve downstream robotics tasks with model-predictive control. We demonstrate our approach on a real robot equipped with a compliant Soft-Bubble tactile sensor on non-prehensile manipulation and dense packing tasks. Trained on only an average of 30 minutes of real-world interaction data per task, our model can perform online adaptation and make touch-informed predictions. Our method demonstrates superior effectiveness compared to previous learning-based and physics-based simulation systems. *See the supplementary material for additional analysis and full technical details.

I. Introduction

Imagine packing an item into a nearly full suitcase. As humans, we typically first form a visual representation of the scene and then make attempts to insert the object, *feeling* the compliance of the objects already inside to decide where and how to insert the new object. If a particular region feels soft, we can then apply additional force to make space and squeeze the new object in. This process is natural for us humans but very challenging for current robotic systems.

What would it take to produce adept packing capabilities in robots? Firstly, a robot needs to understand how its actions will affect the objects in the scene and how those objects will interact with each other. Second, it should be able to estimate unobserved physics properties of objects, such as friction and deformability, to perform dynamics prediction for accurate planning. Third, in scenarios where visual observations are hindered by significant occlusions, the robot should adeptly utilize tactile signals to perceive the environment.

In this work, we propose **RoboPack**, a framework that integrates a learned tactile-informed dynamics model with model-predictive control (MPC) to achieve challenging robotics tasks. We use the SoftBubble tactile sensor [20] to perceive the force applied to the gripper, integrate the force reading with our particle-based object representation, and then perform state estimation and dynamics prediction for planning.

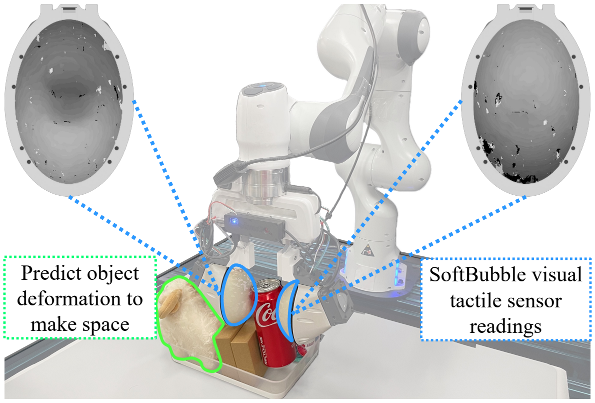

Predict object deformation to make space
SoftBubble visual tactile sensor readings

Fig. 1: **Dense packing**. We use compliant tactile sensors to provide crucial information to a robot to perform dense packing tasks, such as placing a can into a packed tray with both rigid and deformable objects.

We deploy RoboPack on two real-world settings—a tool-use manipulation and a dense packing task. These tasks involve multi-object interactions with complex dynamics that cannot be determined from vision alone. We find that our method can successfully leverage histories of visuo-tactile information to improve prediction, with just 30 minutes of interaction data per task for training.

II. Method

The objective of RoboPack is to manipulate objects with unknown physical properties in environments with heavy occlusions like dense packing. To formulate this problem, we define the observation space as $\emptyset$, the state space as $\mathcal{S}$, and the action space as A. Our goal is to learn a state estimator g that maps $\emptyset$ to $\mathcal{S}$ and a transition function $\mathcal{T}$: $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$.

Our system has four main components: perception, state estimation, dynamics prediction, and model-predictive control.

First, the perception system extracts particles from the scene as a visual representation o^{vis} and encodes tactile readings into latent embeddings o^{tact} attached to those particles.

Secondly, the state estimator g infers object states s from any prior interactions, which includes a single visual frame o_0^{vis}, the subsequent tactile observations $o_{0:t}^{tact}$, and the corresponding robot actions $a_{1:t-1}$:

$$\hat{s}_t = g(o_0^{vis}, o_{0:t}^{tact}, a_{1:t-1}).$$

Thirdly, to enable model-predictive control, we learn a dynamics prediction model f that predicts future states given

[*] These authors contributed equally to this work.

[1] Agency for Science, Technology and Research (A*STAR), Singapore [2] Stanford University, USA [3] University of Illinois Urbana-Champaign, USA

the estimated current states and potential actions:

$$\hat{s}_{t+1} = f(\hat{s}_t, a_t).$$

Lastly, the future predictions are used to evaluate and optimize the cost of sampled action plans. The objective is to find a sequence of actions $a_0, a_1, ..., a_{H-1}$ to minimize a cost function $\mathcal{J}$ between the final states and a given target state s_g:

$$(a_0, ..., a_{H-1}) = \operatorname*{arg\,min}_{a_0,...,a_{H-1} \in \mathcal{A}} \mathcal{J}(\mathcal{T}(s_0, (a_0, .., a_{H-1})), s_g).$$

The robot executes the best actions and receives tactile feedback from the environment, with which it updates its estimates about object properties. Details of the implementation of these modules are provided in Appendix B.

III. Experiments

In this section, we investigate the following questions.

i. Does integrating tactile sensing information from prior interactions improve future prediction accuracy?
ii. Do the latent representations learned by tactile dynamics models discover meaningful properties such as the physical properties of objects?
iii. Does our tactile-informed model-predictive control framework enable robots to solve tasks involving objects of unknown physical properties?

We compare our approach against three baselines:

i. **RoboPack (no tactile)**: To study the effects of using tactile sensing in state estimation and dynamics prediction.
ii. **RoboCook + tactile**: This approach differs from ours in that it treats the observations, i.e., visual and tactile observations $\langle o^{vis}, o^{tact} \rangle$, directly as the state representation. This can be viewed as an adaptation of previous work [28, 48, 50, 49] to include an additional tactile observation component. With this baseline, we seek to study different state representations and our strategy of separating state estimation from dynamics prediction.
iii. **Physics-based simulator**: We also compare our method to using a physics-based simulator [2] for dynamics prediction after performing system identification.

We answer these questions via empirical evaluation of two tasks, *Non-Prehensile Box Pushing* where the robot pushes a box with unknown mass distribution to a target configuration with a loosely held in-hand object, and *Dense Packing* where the robot inserts an object into a nearly full box and the key is to identify insertable region via tactile sensing.

A. Evaluating Dynamics Prediction

Results are summarized in Table I. On the *Non-Prehensile Box Pushing* task, RoboPack is significantly better than alternative methods. For the *Dense Packing* task, our model outperforms the best baseline on the pushing task, RoboPack (no tactile). This shows that integrating history tactile information is helpful to dynamics prediction.

Task	Method	MSE *1e-3 ↓	Success Rate ↑
Box Pushing	**RoboPack**	**1.48 ± 0.14**	**80%**
	RoboPack (no tactile)	1.75 ± 0.15	40%
	RoboCook + tactile	2.11 ± 0.17	30%
	Physics-based sim.	2.65 ± 0.18	50%
Dense Packing	**RoboPack**	**0.070 ± 0.005**	**80%**
	RoboPack (no tactile)	0.088 ± 0.006	40%

TABLE I: **Long-horizon dynamics prediction results and robotic task performance on the two tasks.** Errors represent a 95% confidence interval.

B. Analysis of Learned Physics Parameters

To answer the second question, we first attempt to train a linear classifier that predicts the box type from the generated latent physics vectors ξ_t. We find that the accuracy of such a classifier increases as the amount of historical interaction provided to the representation grows. Qualitatively, we can also observe that the model learns distinguishable latent vectors for different box types, showing that the model learns meaningful properties of objects without explicit supervision from data. Details on the analysis are provided in Appendix G-D.

C. Benchmarking Real-World Planning Performance

Next, we evaluate the performance of our approach in solving real-world robotic planning tasks. Quantitative results are presented in Table I.

For *Non-Prehensile Box Pushing*, we can see that our method has the highest success rate across all box configurations. On the *Dense Packing* task, we compare against the best method on object pushing, i.e., the physics-based simulator. However, it is not feasible to obtain corresponding object models for the diverse and complex objects in this task, so we use RoboPack (no tactile) as the baseline. We test both methods on packing in 15 object configurations and they achieve success rates of 80% and 40% respectively. This shows that our method is much more effective in identifying objects that are deformable or pushable, which consequently enables the robot to insert the object at feasible locations.

IV. Discussion

We presented RoboPack, a framework for learning tactile-informed dynamics models for manipulating objects in multi-object scenes with varied physical properties. By integrating information from prior interactions from a compliant visual tactile sensor, our method adaptively improves predicted dynamics, resulting in improved physical prediction and downstream planning performance on two challenging manipulation tasks, *Non-Prehensile Box Pushing* and *Dense Packing*. We hope that this is a step towards robots that can seamlessly integrate information with multiple modalities from their environments to guide their decision-making.

References

[1] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. Interaction

Networks for Learning about Objects, Relations and Physics. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[2] Victor Blomqvist. Pymunk, November 2023. URL https://pymunk.org.

[3] Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H. Adelson, and Sergey Levine. More Than a Feeling: Learning to Grasp and Regrasp Using Vision and Touch. *IEEE Robotics and Automation Letters*, 3(4):3300–3307, October 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2852779.

[4] Peng Chang and Taşkın Padır. Model-Based Manipulation of Linear Flexible Objects with Visual Curvature Feedback. In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1406–1412, Boston, MA, USA, July 2020. IEEE Press. doi: 10.1109/AIM43001.2020.9159044.

[5] Haonan Chen, Yilong Niu, Kaiwen Hong, Shuijing Liu, Yixuan Wang, Yunzhu Li, and Katherine Rose Driggs-Campbell. Predicting Object Interactions with Behavior Primitives: An Application in Stowing Tasks. In *7th Annual Conference on Robot Learning*, August 2023.

[6] Ravinder S. Dahiya, Giorgio Metta, Maurizio Valle, and Giulio Sandini. Tactile Sensing—From Humans to Humanoids. *IEEE Transactions on Robotics*, 26(1):1–20, February 2010. ISSN 1941-0468. doi: 10.1109/TRO.2009.2033627.

[7] Elliott Donlon, Siyuan Dong, Melody Liu, Jianhua Li, Edward Adelson, and Alberto Rodriguez. GelSlim: A High-Resolution, Compact, Robust, and Calibrated Tactile-sensing Finger. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1927–1934, October 2018. doi: 10.1109/IROS.2018.8593661.

[8] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex X. Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *CoRR*, abs/1812.00568, 2018. URL http://arxiv.org/abs/1812.00568.

[9] Ruohan Gao, Yiming Dou, Hao Li, Tanmay Agarwal, Jeannette Bohg, Yunzhu Li, Li Fei-Fei, and Jiajun Wu. The Object Folder Benchmark : Multisensory Learning with Neural and Real Objects. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17276–17286, Vancouver, BC, Canada, June 2023. IEEE. ISBN 9798350301298. doi: 10.1109/CVPR52729.2023.01657.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

[11] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 2019. URL http://proceedings.mlr.press/v97/hafner19a.html.

[12] Haoyang He. A survey on offline model-based reinforcement learning. *CoRR*, abs/2305.03360, 2023. doi: 10.48550/ARXIV.2305.03360. URL https://doi.org/10.48550/arXiv.2305.03360.

[13] Todd Hester and Peter Stone. TEXPLORE: real-time sample-efficient reinforcement learning for robots. *Mach. Learn.*, 90(3):385–429, 2013. doi: 10.1007/S10994-012-5322-7. URL https://doi.org/10.1007/s10994-012-5322-7.

[14] Philipp Holl, Nils Thuerey, and Vladlen Koltun. Learning to Control PDEs with Differentiable Physics. In *International Conference on Learning Representations*, September 2019.

[15] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2): 99–134, 1998.

[16] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *CoRR*, abs/2104.08212, 2021. URL https://arxiv.org/abs/2104.08212.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[18] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[19] Naveen Kuppuswamy, Alex Alspach, Avinash Uttamchandani, Sam Creasey, Takuya Ikeda, and Russ Tedrake. Soft-bubble grippers for robust and perceptive manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9917–9924, October 2020. doi: 10.1109/IROS45743.2020.9341534.

[20] Naveen Kuppuswamy, Alex Alspach, Avinash Uttamchandani, Sam Creasey, Takuya Ikeda, and Russ Tedrake. Soft-bubble grippers for robust and perceptive manipulation. In *IEEE/RSJ International Conference on Intelligent*

Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021, pages 9917–9924. IEEE, 2020. doi: 10.1109/IROS45743.2020.9341534. URL https://doi.org/10.1109/IROS45743.2020.9341534.

[21] Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In Oliver Brock, Jeff Trinkle, and Fabio Ramos, editors, *Robotics: Science and Systems IV, Eidgenössische Technische Hochschule Zürich, Zurich, Switzerland, June 25-28, 2008*. The MIT Press, 2008.

[22] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning Plannable Representations with Causal InfoGAN. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[23] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, Dinesh Jayaraman, and Roberto Calandra. DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor With Application to In-Hand Manipulation. *IEEE Robotics and Automation Letters*, 5(3):3838–3845, July 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.2977257.

[24] Christian Legaard, Thomas Schranz, Gerald Schweiger, Ján Drgoňa, Basak Falay, Cláudio Gomes, Alexandros Iosifidis, Mahdi Abkar, and Peter Larsen. Constructing Neural Network Based Models for Simulating Dynamical Systems. *ACM Computing Surveys*, 55(11): 236:1–236:34, February 2023. ISSN 0360-0300. doi: 10.1145/3567591.

[25] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

[26] Hao Li, Yizhi Zhang, Junzhe Zhu, Shaoxiong Wang, Michelle A. Lee, Huazhe Xu, Edward Adelson, Li Fei-Fei, Ruohan Gao, and Jiajun Wu. See, Hear, and Feel: Smart Sensory Fusion for Robotic Manipulation. In *Proceedings of The 6th Conference on Robot Learning*, pages 1368–1378. PMLR, March 2023.

[27] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids. In *International Conference on Learning Representations*, September 2018.

[28] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[29] Yunzhu Li, Antonio Torralba, Animashree Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, pages 9180–9192, Red Hook, NY, USA, December 2020. Curran Associates Inc.

[30] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3D Neural Scene Representations for Visuomotor Control. In *5th Annual Conference on Robot Learning*, June 2021.

[31] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1509.02971.

[32] Changyi Lin, Han Zhang, Jikai Xu, Lei Wu, and Huazhe Xu. 9DTact: A Compact Vision-Based Tactile Sensor for Accurate 3D Shape Reconstruction and Generalizable 6D Force Estimation, December 2023.

[33] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning Visible Connectivity Dynamics for Cloth Smoothing. In *Proceedings of the 5th Conference on Robot Learning*, pages 256–266. PMLR, January 2022.

[34] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[35] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees. In *International Conference on Learning Representations*, September 2018.

[36] Carolyn Matl and Ruzena Bajcsy. Deformable Elasto-Plastic Object Shaping using an Elastic Hand and Model-Based Reinforcement Learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3955–3962, Prague, Czech Republic, September 2021. IEEE. ISBN 978-1-66541-714-3. doi: 10.1109/IROS51168.2021.9636808.

[37] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=3hGNqpI4WS.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015. doi: 10.1038/NATURE14236. URL https://doi.org/10.1038/

nature14236.

[39] J. Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradSim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*, October 2020.

[40] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 1101–1112. PMLR, 2019. URL http://proceedings.mlr.press/v100/nagabandi20a.html.

[41] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *Proceedings of the Conference on Robot Learning*, pages 1101–1112. PMLR, May 2020.

[42] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *CoRR*, abs/2304.07193, 2023.

[43] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[44] Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General In-hand Object Rotation with Vision and Touch. In *7th Annual Conference on Robot Learning*, August 2023.

[45] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In Ali Jadbabaie, John Lygeros, George J. Pappas, Pablo A. Parrilo, Benjamin Recht, Claire J. Tomlin, and Melanie N. Zeilinger, editors, *Proceedings of the 3rd Annual Conference on Learning for Dynamics and Control, L4DC 2021, 7-8 June 2021, Virtual Event, Switzerland*, volume 144 of *Proceedings of Machine Learning Research*, pages 1154–1168. PMLR, 2021. URL http://proceedings.mlr.press/v144/rafailov21a.html.

[46] Marc Rigter, Bruno Lacerda, and Nick Hawes. RAMBO-RL: robust adversarial model-based offline reinforcement learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6691c5e4a199b72dffd9c90acb63bcd6-Abstract-Conference.html.

[47] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to Simulate Complex Physics with Graph Networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8459–8468. PMLR, November 2020.

[48] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. RoboCraft: Learning to See, Simulate, and Shape Elasto-Plastic Objects with Graph Networks. In *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation, June 2022.

[49] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools. In *7th Annual Conference on Robot Learning*, 2023.

[50] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. RoboCraft: Learning to see, simulate, and shape elasto-plastic objects in 3D with graph networks. *The International Journal of Robotics Research*, 2023.

[51] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 2164–2172, 2010.

[52] H.J. Terry Suh, Naveen Kuppuswamy, Tao Pang, Paul Mitiguy, Alex Alspach, and Russ Tedrake. SEED: Series Elastic End Effectors in 6D for Visuotactile Tool Use. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[53] Sudharshan Suresh, Haozhi Qi, Tingfan Wu, Taosha Fan, Luis Pineda, Mike Lambeta, Jitendra Malik, Mrinal Kalakrishnan, Roberto Calandra, Michael Kaess, Joseph Ortiz, and Mustafa Mukadam. Neural feels with neural fields: Visuo-tactile perception for in-hand manipulation, 2023.

[54] Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-Resolution Model Learning for Object Pile Manipulation. In *Robotics: Science and Systems XIX*, 2023.

[55] Yixuan Wang, Zhuoran Li, Mingtong Zhang, Katherine Driggs-Campbell, Jiajun Wu, Li Fei-Fei, and Yunzhu

Li. D^{3}fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation. *arXiv preprint arXiv:2309.16118*, 2023.

[56] R. Weinstein, J. Teran, and R. Fedkiw. Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):365–374, May 2006.

[57] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, 2016.

[58] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 2226–2240. PMLR, 2022. URL https://proceedings.mlr.press/v205/wu23c.html.

[59] Wenzhen Yuan, Siyuan Dong, and Edward H. Adelson. GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force. *Sensors*, 17(12):2762, December 2017.

[60] Ying Yuan, Haichuan Che, Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Kang-Won Lee, Yi Wu, Soo-Chul Lim, and Xiaolong Wang. Robot Synesthesia: In-Hand Manipulation with Visuotactile Sensing, 2023.

[61] Marvin Zhang, Sharad Vikram, Laura M. Smith, Pieter Abbeel, Matthew J. Johnson, and Sergey Levine. SOLAR: deep structured representations for model-based reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7444–7453. PMLR, 2019. URL http://proceedings.mlr.press/v97/zhang19m.html.

[62] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. *arXiv preprint arXiv:2210.11339*, 2022. doi: 10.48550/arXiv.2210.11339.

Appendix A
Related Work

A. Learning Dynamics Models

Simulators developed to model rigid and non-rigid bodies approximate real-world physics, often creating a significant sim-to-real gap [56, 14, 39]. To address this, we use a graph neural network (GNN)-based dynamics model trained directly on real-world robot interaction data, aligning with data-driven approaches for learning physical dynamics [41, 35]. Recent works have demonstrated inspiring results in learning the complex dynamics of objects such as clothes [33], ropes [4], and fluid [24], with various representations including low-dimensional parameterized shapes [36], keypoints [29], latent vectors [22], and neural radiance fields [30]. RoboPack, inspired by previous works [27, 47, 1], focuses on the structural modeling of objects with minimal assumptions about underlying physics. This approach overcomes the limitations of physics simulators by directly learning from real-world dynamics. Prior work on GNN-based dynamics learning [48, 49, 50, 54, 5] heavily relies on visual observations for predicting object dynamics, failing to capture unobserved latent variables that affect real-world dynamics, such as object physical properties. To address this challenge, our method incorporates tactile sensing into dynamics learning and leverages history information for state estimation, offering a robust solution to overcome the constraints of vision-only models.

B. Model-Free and Model-Based Reinforcement Learning

Reinforcement learning (RL) aims to derive policies directly from interactions. Our method contrasts with model-free RL approaches [38, 31, 10, 16, 25], by incorporating an explicit dynamics model, enhancing interpretability and including structured priors for improved generalization. Our work is closer to model-based RL [13, 11, 40, 46, 37, 61] in that we combine learned world models with planning via trajectory optimization. In particular, we learn world models in an offline manner from pre-collected interaction data, avoiding risky trial-and-error interactions in the real world. However, our approach is different from existing offline model-based RL [45, 58, 8, 12] as it leverages multiple sensing modalities, i.e., tactile and visual perception. This multi-modal approach provides a more comprehensive understanding of both global geometry and the intricate local physical interactions between the robot gripper and objects. Moreover, our method addresses challenges in scenarios where visual observations are not always available. It uses tactile observation histories to estimate partially observable states, enabling online adaptation to different dynamics. This integration of offline model learning, multi-modal perception, and online adaptation equips our system with adaptive control behaviors for complex tasks.

C. Tactile Sensing for Robotic Manipulation

Tactile sensing plays an important role in both human and robot perception [6]. Among all categories of tactile sensors, vision-based sensors such as [59, 7, 23, 32] can achieve accurate 3D shape perception of their sensing surfaces. In our work, we use the Soft-Bubble tactile sensor [20] which offers a unique combination of compliance, lightweight design, robustness to continuous contact, and the ability to capture detailed geometric features through high-resolution depth images [19, 52]. Previous studies have successfully integrated vision and tactile feedback in robotic manipulation using parallel grippers [3, 9, 26] and dexterous hands [44, 53, 60]. In these tasks, vision effectively offers a comprehensive understanding of the scene's semantics, while tactile sensing delivers accurate geometry estimation for objects in contact that are often occluded. In our study, we explore the potential of integrating vision and tactile feedback for learning dynamics in tasks involving rich contact, occlusions, and a diverse set of objects with unknown physical properties, such as box pushing and dense packing.

Appendix B
Method

A. Perception

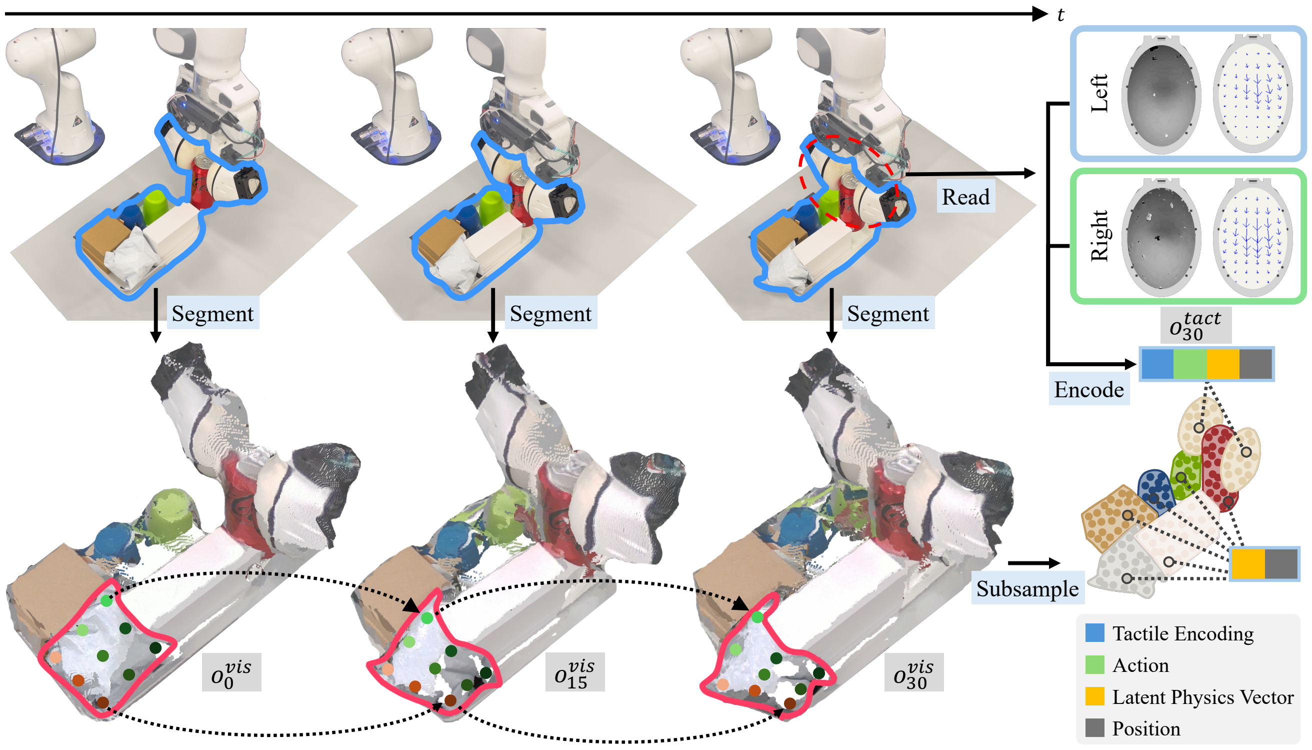
1) Visual Perception: Our visual perception module extends the formulation of D^{3}Fields [55], with an additional deformation term to handle non-rigid objects and mask-based closeness loss to better support multi-object scenes with occlusion. As shown in Figure 2(a), it takes in multi-view RGB-D observations and outputs tracked 3D keypoints for each object of interest. Critically for our training procedure, these keypoints maintain correspondences over time—a tracked point stays at the same region of an object throughout the trajectory.

First, we extract visual features for each object with a pre-trained DINOv2 model [42] and masks using GroundedSAM [43, 18, 34]. Through projection and interpolation, we can then compute semantic, instance, and geometric features for arbitrary 3D points. We initialize desired tracking points on object surfaces for an initial frame and formulate 3D keypoint tracking for subsequent frames as an optimization problem. The tracking objective has the following terms:

- **Distance to surface**. Use depth information to encourage points to be close to object surfaces.
- **Semantic alignment**. Align DINOv2 features between projected points in the current and initial frame.
- **Motion regularization**. Penalize large motion between consecutive frames to avoid jitter.
- **Mask consistency**. For multi-object packing settings with significant occlusion, we introduce an objective that constrains tracked points to be near the corresponding object masks, providing more consistent optimization signal for object pose than semantic alignment.

We optimize a translation and rotation transformation for each object with this objective. For deformable objects, we also predict axis-aligned shearing scales apart from a rigid transformation to track deformations.

2) Tactile Perception: As shown in the top right of Figure 2, our tactile perception module takes global force-torque and local force vectors as input and outputs embeddings for the tactile reading. Each Soft-Bubble tactile sensor provides its



(a) 3D Point Tracking on Point Cloud Observations

(b) Scene Representation

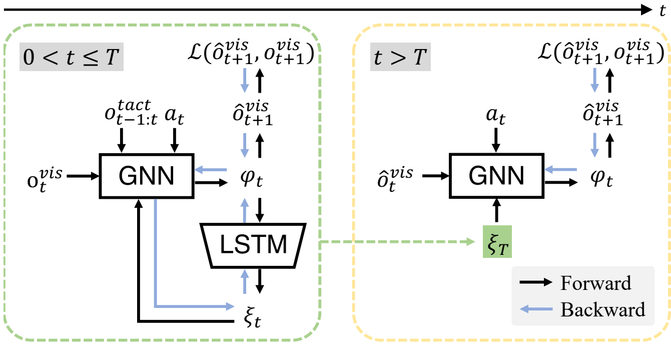
Fig. 2: **RoboPack's perception module.** (a) We construct a trajectory comprising particle representations of the scene, maintaining correspondence via 3D point tracking on the point cloud data. (b) These particles facilitate the creation of a visual scene representation, denoted as o_t^{vis}. For points representing the Soft-Bubble grippers, tactile encodings o_t^{tact} and latent physics vectors are integrated as extra attributes of the particles.

surface force distribution. This includes (1) shear force vectors $\{\langle q^x_{i,j}, q^y_{i,j}\rangle\}_{i,j}$, where i, j is the coordinate of a point on the 2D surface of the bubble and x, y denote the vertical and horizontal axis of the tangent plane at that point, as well as (2) a global shear force torque vector and the overall force magnitude $\langle Q^x, Q^y, |Q|\rangle$. F^x, F^y are the mean of local force vectors across spatial dimensions, and $|Q|$ is defined as

$$|Q| = \sqrt{\max_{i,j} |q^x_{i,j}|^2 + \max_{i,j} |q^y_{i,j}|^2}, \tag{1}$$

where H and W are tactile reading's spatial dimensions.

3) Integrating Visual and Tactile Perception: As depicted in Figure 2(b), to integrate tactile observations with particle-based object representation, we first extract particles from the surface of the soft-bubble gripper by projecting the depth camera reading inside the gripper into 3D space. Next, we define a point-wise tactile signal as $\langle q^x_{i,j}, q^y_{i,j}, Q^x, Q^y, |Q|\rangle$ and train an auto-encoder that projects the point-wise signals independently into latent embeddings. We denote the collection of embeddings as the tactile observation o^{tact}. Lastly, we combine the object particles from the visual observation o^{vis} with the tactile sensor particles o^{tact} to form a unified particle representation of the scene.



(a) State Estimator

(b) Dynamics Predictor

Fig. 3: **RoboPack's dynamics module.** We perform state estimation and dynamics reasoning with a state estimator and a dynamics predictor respectively. They share the same architecture, except that the state estimator has an LSTM module that integrates history information and predicts physics parameters for each object in the scene.

B. State Estimation and Latent Physics Vector Inference

In real-world robotic manipulation, visual observations are not always available due to occlusion, but knowledge about object dynamics requires interactive feedback. In this work, we leverage tactile feedback to help estimate world states.

History information is often used to estimate the current state in POMDPs [15, 21, 51]. Similarly, we seek to incorporate tactile history information into state estimation by employing a combination of graph neural networks (GNNs) and long-short term memory (LSTM), as shown in Figure 3(a). We define our state as a tuple of object particles and an object-level latent physics vector, which capture the geometry and physics properties of objects respectively. In the following paragraphs, we describe how our method performs state estimation using history information and future prediction.

At time $0 < t \leq T$, our state estimator g infers all states for $t = 1, ..., T$ autoregressively. Given the estimated previous state $\hat{s}_{t-1}$ and the tactile feedback at the previous and the current state $o^{tact}_{t-1:t}$, we construct a graph $G_{t-1} = \langle V_{t-1}, E_{t-1} \rangle$ with V_{t-1} as vertices and E_{t-1} as edges. For each node, $v_{i,t-1} = \langle x_{i,t-1}, c^o_{i,t-1} \rangle$, where $x_{i,t-1}$ is the particle position i at time $t-1$, and $c^o_{i,t-1}$ are particle attributes. The particle attributes contain (1) the previous and current tactile readings, $o^{tact}_{t-1:t}$, and (2) the latent physics vector of the object that the particle belongs to, $\xi_{_i,t-1}$, where $_i$ is the object index corresponds to the i-th particle, $1 \leq_i \leq Z$ and Z is the maximum number of objects in the scene. Formally, $c^o_{i,t-1} = \langle \xi_{t-1}, o^{tact}_{t-1:t} \rangle$. Note that here we implicitly assume that is constant (i.e., objects only exhibit elastic and plastic deformations but not break apart), which generally holds for a large number of common manipulation tasks. Moreover, edges between pairs of particles are denoted $e_k = \langle u_k, v_k \rangle$, where u_k and v_k are the receiver and sender particle indices respectively, and $1 \leq u_k, v_k \leq |V_{t-1}|$ where k is the edge index. We construct graphs by connecting any nodes within a certain radius of each other.

Given the graph, we first use a node encoder f_V^{enc} and an edge encoder f_E^{enc} to obtain node and edge features, respectively:

$$h^v_{i,t-1} = f_V^{enc}(v_{i,t-1}), \qquad h^e_{k,t-1} = f_E^{enc}(e_{k,t-1}). \qquad (2)$$

Then, the features are propagated through the edges in multiple steps, during which node effects are processed by neighboring nodes through learned MLPs. We summarize this procedure as f_E^{dec}, which outputs an aggregated effect feature for each node called ϕ_i:

$$\phi_{i,t-1} = f_E^{dec}(h^v_{i,t-1}, \sum_{k \in_i} h^e_{k,t-1})_{k=1,\dots,|E_{t-1}|}. \qquad (3)$$

where $_i$ is a set of relations with particle i as the receiver.

Next, the model predicts node (particle) positions and updates the latent physics vector:

$$\hat{o}^{vis}_{i,t} = f_V^{dec}\left(h^v_{i,t-1}, \phi_{i,t-1}\right)_{i=1,\dots,|V_{t-1}|}, \qquad (4)$$

$$\xi_{\eta,t}, m_t = f_\xi^{dec}\left(\sum_{\substack{i \\ _i=\eta}} h^v_{i,t-1}, \sum_{\substack{i \\ _i=\eta}} \phi_{i,t-1}, m_{t-1}\right)_{\eta=1,\dots,Z} \qquad (5)$$

where f_ξ^{dec} is an LSTM, m_t is its internal cell state at the current step, and $\xi_{\eta,t}$ is the updated physics latent vector for η-th object. At $t = 0$ the LSTM state m_0 is initialized as zero. The physics vector for each object is initialized as Gaussian noise: $\xi_{\eta,0} \sim \mathcal{N}(0, 0.1^2)$ for all η. All other encoder and decoder functions (i.e., f_V^{enc}, f_V^{dec}, f_E^{enc}, and f_E^{dec}) are MLPs.

C. Dynamics Prediction

After the state estimator produces an estimated state $\hat{s}_T = \langle \hat{o}^{vis}_T, \xi_T \rangle$ from the T-step history, our dynamics model predicts into the future to evaluate potential action plans. The dynamics predictor f is constructed similarly to the state estimator g, with two key differences: (i) it does not use tactile observations as input, and (ii) it is conditioned on frozen physics parameters estimated by g. Figure 3 illustrates this process. The forward prediction happens recursively: For a step $t > T$, we construct a graph in the same way as in Section B-B, but excluding tactile observations from the particle attributes, i.e., $c^o_{i,t} = \xi_t$. Then, the dynamics predictor infers the particle positions at the next step $\hat{o}^{vis}_{t+1}$ as formulated in Equations 2-4. The final state prediction is then $\hat{s}_{t+1} = \langle \hat{o}^{vis}_{t+1}, \xi_t \rangle$. Note that the estimated physics parameters are not modified by the dynamics predictor.

Training procedure and objective. We train the state estimator and dynamics predictor jointly end-to-end on trajectories of sequential interaction data containing observations and robot actions. For a training trajectory of length H, the state estimator estimates the first T states, and the dynamics predictor predicts all remaining states. The estimation and prediction are all computed autoregressively. The loss is computed only on visual observations:

$$\mathcal{L} = \frac{1}{H} \sum_{t=0}^{H-1} ||\hat{o}^{vis}_t - o^{vis}_t||_2^2. \qquad (6)$$

Previous works [48, 50, 49] use the earth mover's distance (EMD) or chamfer distance (CD) as the training loss, but these provide noisier gradients because EMD requires estimating point-to-point correspondence and CD is prone to outliers. Instead, we use mean squared error (MSE) as the objective, enabled by the point-to-point correspondences from our 3D point tracking (Section B-A). The details of the architecture and training procedure of the state estimator and dynamics predictor are in Appendix C-B.

Note that the learning of the latent physics information is not explicitly supervised. The model is allowed to identify any latent parameters that enhance its ability to accurately estimate the current state and predict future outcomes. We provide an analysis on the learned physics parameters in Section III.

D. Model-Predictive Control

With the learned state estimator and dynamics predictor, we perform planning toward a particular goal by optimizing a cost function on predicted states over potential future actions. Concretely, we use Model Predictive Path Integral (MPPI) to perform this optimization [57].

Planning begins with sampling actions from an initial distribution performing forward prediction with the dynamics models. The cost is then computed on predicted states. Based on the estimated costs, we re-weight the action samples by importance sampling and update the distribution parameters. The process repeats for multiple interactions and we select the optimal execution plan.

For computational efficiency, we execute the first K planning steps. While executing the actions, the robot records its tactile readings. After execution, it performs state estimation with the history of observations and re-plans for the next execution. More implementation details on planning can be found in Appendix F.

To summarize this section, a diagram of the entire system workflow including training and test-time deployment is available in Figure 5.

Appendix C
Model Architecture and Training

A. Tactile Autoencoder

Both the encoder and decoder are two-layer MLPs with hidden dimension 32 and ReLU activations. The encoder maps the raw point-wise tactile signal to latent space, then the decoder maps it back to the original dimension. The autoencoder is trained with MSE loss using the following hyper-parameters:

Hyperparameter	Value
Learning rate	5e-4
Optimizer	Adam [17]
Batch size	32
Latent space dimension	5

TABLE II: Hyperparameters for auto-encoder training.

B. State Estimator and Dynamics Predictor

We use the same hyperparameters to train dynamics models for the nonprehensile box pushing and dense packing tasks, which are shown in Table III For graph construction, we connect any points within a radius of 0.15. We train the state estimator and dynamics model jointly, using sequences of length 25. To prevent the model from overfitting to a specific history length, which could vary at deployment time, we use the first k steps in a sequence as the history, $k \sim \text{Uniform}(0, 24)$. To stabilize training, we restrict the magnitude of the rotation component of predicted rigid transformations for a single step to be at most 30 degrees, which is much larger than any rotation that occurs in our datasets. Model training converges within 25 and 8 hours on the two tasks respectively with one NVIDIA RTX A5000 GPU.

For baselines RoboPack (no tactile) and RoboCook + tactile, we performed a hyper-parameter sweep and the optimal training parameters are the same as RoboPack described above.

Appendix D
Hardware Setup

The hardware setup is depicted in Figure 4.

Robot. We use a Franka Emika Panda robot arm, controlled using the Deoxys open-source controller library [62]. In our experiments, we use the `OSC_POSITION` and `OSC_YAW` controllers provided by the Deoxys library.

Sensors. We attach the Soft-Bubble sensors to the Franka Panda gripper using custom-designed 3D-printed adapters. We inflate both Soft-Bubble sensors to a width of 45mm measured from the largest distance sensor frame to the rubber sensor surface. While there can be slight variations in the exact amount of air in the sensor due to measurement error, we do not find this to be a significant cause of domain shift for learned models, likely because the signals that are used as input to our model are largely calculated using differences between the current reading and a reference frame captured when the gripper does not make contact with any object that we reset upon each inflation. While we contribute a novel method for integrating tactile information into the particle-based scene representation, the computation of raw tactile features is computed by the Soft-Bubble sensor API [20] and is not part of our contribution.

Hyperparameter	Value
Learning rate	5e-4
Optimizer	Adam [17]
Batch size	4
Graph construction criteria	Radius
Graph connection radius	0.15m
Training sequence length	25 steps
Training history length	15 steps
# graph points per object	20
# graph points per tactile sensor	20
Node encoder MLP width	150
Node encoder MLP layers	3
Edge encoder MLP width	150
Edge encoder MLP layers	3
Edge effect MLP width	150
Edge effect MLP layers	3
Edge propagation steps	3
Latent physics vector size (dim(ξ))	16
Tactile encoding dimension (per point in o^{tact})	5

TABLE III: Hyperparameters for dynamics model training. We use the same hyperparameters for the nonprehensile box pushing and dense packing tasks.

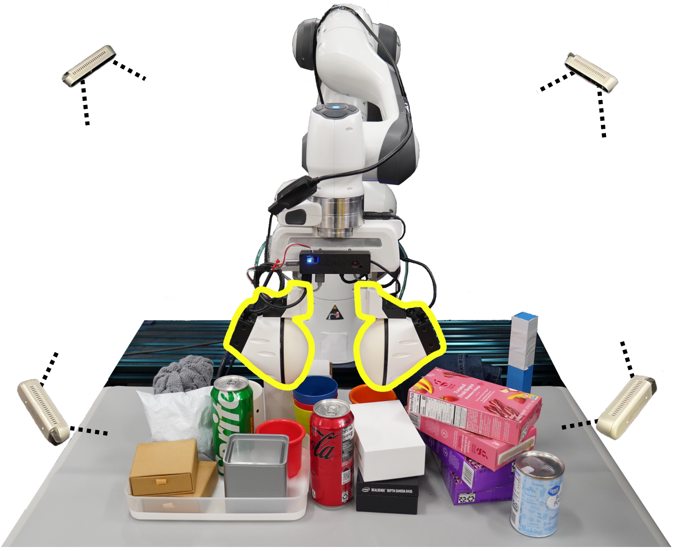
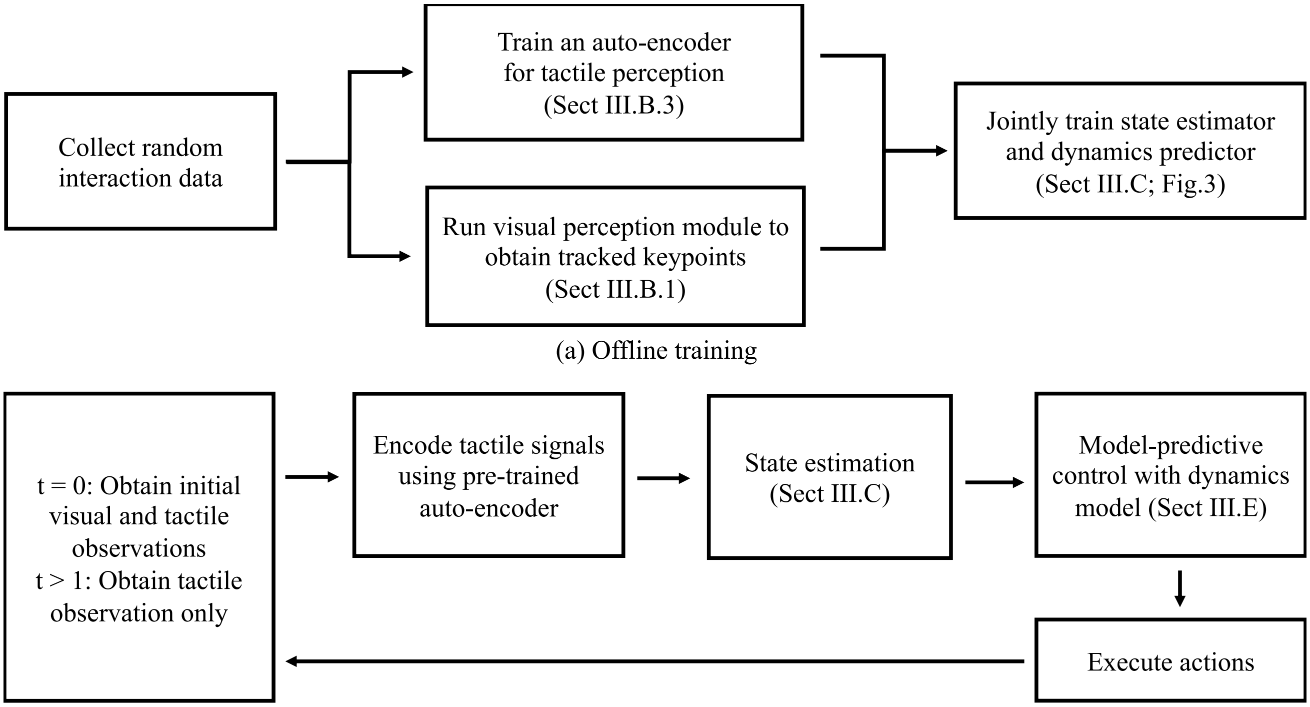


Fig. 4: **Hardware overview**. Our experimental platform consists of a Franka Panda arm, two Soft-Bubble grippers, four RealSense D415 RGBD cameras, and a diverse set of objects.


Collect random interaction data
Train an auto-encoder for tactile perception (Sect III.B.3)
Run visual perception module to obtain tracked keypoints (Sect III.B.1)
Jointly train state estimator and dynamics predictor (Sect III.C; Fig.3)
t = 0: Obtain initial visual and tactile observations
t > 1: Obtain tactile observation only
Encode tactile signals using pre-trained auto-encoder
State estimation (Sect III.C)
Model-predictive control with dynamics model (Sect III.E)
Execute actions

(a) Offline training

(b) Online planning

Fig. 5: **The complete workflow of the RoboPack system. There are two main stages of the system: (a) offline training and (b) online planning.**

Appendix E
System Workflow

To present the offline training and online planning processes more clearly, a system diagram is provided in Figure 5.

Appendix F
Planning Implementation Details

We provide hyperparameters for the MPPI optimizer that is used for planning with learned dynamics models in Table IV. We use the same planning hyperparameters for baselines as we do our method.

Hyperparameter	Box pushing	Dense packing
History length	22	25
Action sampler temporal correlation β*	0.2	N/A
MPPI # action samples	400	150
MPPI action horizon	20	80
MPPI # iterations	2	1
MPPI scaling temperature γ*	100	N/A
# steps executed before replanning K	20	45

TABLE IV: Hyperparameters for real world planning experiments. For the parameters denoted by *, we use the notation from Nagabandi et al. [41]. K refers to the number of steps in the best plan found that is actually executed on the real robot before replanning. For box pushing it is the entire plan, while for dense packing it is 45 out of 80 steps.

Appendix G
Experiments

A. Box Configurations

For the non-prehensile box pushing task, we use boxes that have the same geometry different weight configurations to test the ability of each model to adapt its prediction based on the physical characteristics of each scenario. Specifically, we empty cardboard boxes of dimensions $18 \times 9.5 \times 3.8$ cm, and then add metal weights in the following configurations:

- **Box 1**: Two 100g weights placed at opposing corners of the box.
- **Box 2**: One 200g weight placed at the geometric center of the box.
- **Box 3**: No additional weight added.
- **Box 4 (unseen during training)**: This is the original unmodified box, which contains roughly uniformly distributed food items. The items are not affixed to the inner sides of the box, and there could be relative movement between the box and its contents if force is applied.

B. Qualitative Results on Planning

Additional qualitative results on non-prehensile box pushing and dense packing are presented in Figure 7 and Figure 8 respectively. Please additionally see our supplementary video for video examples of planning executions.

C. Physics-Based Simulator Baseline

Here we provide additional details about the physics-based simulator baseline used for the box pushing experiments in Section III.

First, we construct a 2D version of the task in the open source Pymunk simulator [2] that emulates a top-down view of the real scene. The simulated scene contains replicas of the rod and box produced by measuring the dimensions of the real versions of those objects.

Then, given two visual observations o^{vis}_{init} and o^{vis}_{final}(tracked points for each object) and a sequence of actions $\vec{a}$ taken by the robot, we perform system identification to optimize simulated parameters to fit the real system. Note that our method also uses only two visual observations from the history, but also can use tactile information. Because tactile simulation is not available, the baseline has access to just visual observations. To convert tracked points from real observations into simulator states, we project all points into 2D by truncating the z dimension, and then for each object we compute the object center with the spatial mean of points and the 2D rotation by finding the first two principal components of the 2D points with PCA. Thus the visual observations are converted into tuples $(pos^{rod}_{init}, rot^{rod}_{init}), (pos^{box}_{init}, rot^{box}_{init})$, $(pos^{rod}_{final}, rot^{rod}_{final}), (pos^{box}_{final}, rot^{box}_{final})$.

We optimize a vector of parameters $\vec{\mu} \in \mathbb{R}^5$, detailed in Table V. We de-normalize values from $\vec{mu}$ to the actual system parameters and clamp them to prevent unrealistic values based on the minimum and maximum values shown. The initial standard deviation for optimization is $\sigma = 0.3$, which we found to work well empirically. The objective function is

$$\mathcal{L}(\vec{\mu}) = \|(pos^{box}_{final}, rot^{box}_{final}) - \text{SIM}_{\vec{\mu}}(pos_{init}, rot_{init}, \vec{a})\|_2.$$

where $\text{SIM}_{\vec{\mu}}(pos, rot, \vec{a})$ represents the box position and rotation after running a simulated trajectory with actions $\vec{a}$ in the Pymunk simulator starting from box and rod positions pos and rotations rot with simulator parameters set to $\vec{\mu}$.

We optimize the objective using CMA-ES, a gradient free optimizer, using the implementation from https://github.com/CyberAgentAILab/cmaes. Parameters are initialized to have the center of mass at the center of the object uniformly distributed mass, and reasonable friction and mass defaults. We use a population size of 8 based on the implementation-suggested default of $4 + \lfloor 3 * \log(ndim) \rfloor$ and optimize for 100 generations.

Finally, we use the optimized set of parameters to perform forward prediction. After forward prediction, we convert the sequence of simulated 2D object positions into a sequence of pointcloud predictions by estimating a rotation matrix and translation (in 2D) and applying them to the 3D pointcloud for the initially provided observation. The z values (height) of all particles are assumed to be fixed at their initial values throughout the prediction.

D. Analysis of Learned Physics Parameters

In this subsection, we seek to provide some quantitative and qualitative analyses of the latent representation learned by the state estimator. As it gives more direct control of object properties, we use our dataset collected for the *Non-Prehensile Box Pushing* task for the analysis.

To understand if the representation contains information about box types, we first attempt to train a linear classifier to test if there the features learned for different boxes are linearly separable in the latent space. We test the state estimator on 145-step trajectories in the testing data, which typically involves three to five pushes on the box. The classification accuracy of physics parameters ξ_t as more and more interaction information is processed is shown in Figure 6. It can be observed that as history information accumulates, the latent physics vectors become more indicative of the box type. In particular, the state estimator can extract considerable information in the first 20 steps, which is approximately the average number of steps it takes to complete an initial push. Furthermore, note that the state estimator only observes a history of no more than 25 steps during training, but it can generalize to sequences four times longer in this case.

To qualitatively inspect the learned representations, we perform principal component analysis, reducing the learned latent vectors from $\mathcal{R}^{16}$ to $\mathcal{R}^2$. Figure 6 shows the low-dimensional embeddings as the number of interaction time steps incorporated into the latents grows. We can see that as time progresses, the estimated latents become increasingly separated into clusters based on the physical properties (i.e., mass distributions in this case) of the manipulated object. The separation increases the most between $t = 1$ and $t = 20$, which is consistent with our observation in Figure 6 that longer histories than a certain threshold yield marginal returns.

Collectively, the results indicate that our state estimator indeed learns information related to physical properties based on interaction histories.

Appendix H
Tracking Module Details

After sampling initial sets of points for each object $\vec{p}_{init}$, we formulate point tracking as optimization for the points at each step $\vec{p}$. Specifically, the new points are computed as a 3D transformation of the points output at the previous step, represented by a rigid rotation $R \in \mathbb{R}^3$, translation $T \in \mathbb{R}^3$ and optional per-axis shearing $S \in \mathbb{R}^3$. The transform is a composition of rotation by R, scaling by S, and translation by T in that order. We abuse notation to sometimes use $\vec{p}$ for ease of reading, but $\vec{p}$ is a function of the actual optimized parameters R, S, T. Thus the optimization objective has the following loss terms:

1) **Distance to surface.**

$$\mathcal{L}_{depth}(\vec{p}) = \frac{1}{|\vec{p}|} \sum_{p \in \vec{p}} \max(0, depth_{interp(p)} - depth_{proj(\vec{p})})$$

where $depth_{interp(p)}$ is the depth estimation from interpolating information from multi-view depth observations, and $depth_{proj(\vec{p})}$ is the expected depth at each point when projected into each camera frame.

2) **Semantic alignment.**

$$\mathcal{L}_{align}(\vec{p}) = \frac{1}{|\vec{p}|} \sum_{p \in \vec{p}} \min(\|dinov2(p_{init}) - dinov2(p)\|_2, 30)$$

Hyperparameter	Initial value	Min	Max	Optimization space μ to sim. param p transform
Box mass	10	0.001	N/A	$p = 10(\mu + 1)$
Box friction	0.5	0.0001	N/A	$p = 0.5(\mu + 1)$
Moment of inertia	34520.83	10	N/A	$p = 35420.833(\mu + 1)$
Center of gravity x	0	-42.5	42.5	$p = 42.5\mu$
Center of gravity y	0	-90	90	$p = 90\mu$

TABLE V: Parameters optimized during system identification for the physics-based simulator baseline. Initial values and scales are set such that when the parameters in the optimization space are $\vec{\mu} = 0$, the actual values in the physics simulator $\vec{p}$ are sensible defaults (see initial value column). Note for center of gravity, $(0, 0)$ refers to the geometric center of the object.

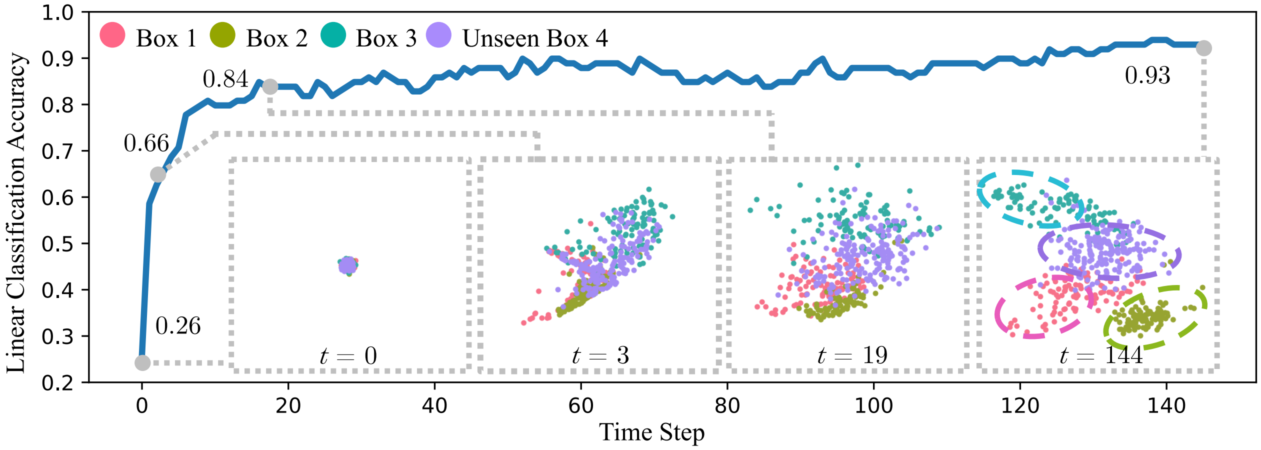


Fig. 6: **Analysis of learned physics parameters.** We assess our state estimator across 145-step trajectories and record the estimated physics parameters at each step. PCA visualizations at four distinct timesteps show that the physics parameters gradually form clusters by box type. We also employ a linear classifier trained on these parameters to accurately predict box types to demonstrate these clusters' linear separability. The classifier's improving accuracy across timesteps underscores the state estimator's proficiency in extracting and integrating box-specific information from the tactile observation history.

where $dinov2(p)$ represents the multi-view interpolated DinoV2 feature at the 3D point represented by p, and again p_{init} is the position of the point in the first frame (not necessarily immediately prior frame) of tracking.

3) **Motion regularization.**

$$\mathcal{L}_{reg}(R, T, S) = w_{reg}^{T}\|R\|_2 + w_{reg}^{T}\|T\|_2 + w_{reg}^{S}\|S\|_2.$$

Motion regularization prevents tracked points from exhibiting high frequency jitter when the objects they are tracking do not move.

4) **Mask consistency.** We introduce a mask consistency loss. Intuitively, this loss tries to ensure that each pixel within a 2D mask for an object from a particular camera view should have a tracked point for that object that is close to that pixel when projected into that view.
Let the set of all views be V and the set of object masks in a particular view v be $M(v)$. Then the total number of masks points N is $N = \sum_{v \in V}\sum_{obj \in M(v)} |obj|$.
Concretely, this can be written as:

$$\mathcal{L}_{mask}(\vec{p}) = \frac{1}{N}\sum_{v\in V}\sum_{obj\in M(v)}\sum_{pix\in obj}\min_{p\in\vec{p}^{obj}}\|pix - proj(p, v)\|$$

where $proj(p, v)$ is the 2D projection of 3D point p into the image space of viewpoint v.

Hyperparameter	Box pushing	Dense packing
Optimizer	Adam	Adam
LR schedule	Reduce on plateau	Reduce on plateau
Grad steps	200	200
Learning rate (T)	0.04	0.01
Learning rate (R)	0.04	0.1
Learning rate (S)	0.04	0.01
Use scale term	No	Yes
w_{depth}	1	1
w_{align}	1	1
w_{reg}^{T}	1e-3	3e3
w_{reg}^{R}	1e-3	1e2
w_{reg}^{S}	N/A	3e3
w_{mask}	100	15

TABLE VI: Loss weights for the tracking module.

The overall objective is computed by weighting and combining these terms:

$$\mathcal{L}_{tracking} = w_{depth}\mathcal{L}_{depth} + w_{align}\mathcal{L}_{align} + \\ + w_{reg}\mathcal{L}_{reg} + w_{mask}\mathcal{L}_{mask}$$

The weights for each term as well as optimizer parameters are enumerated in Table VI. The transformed points with the best loss after the total number of gradient steps is complete is output as the result.

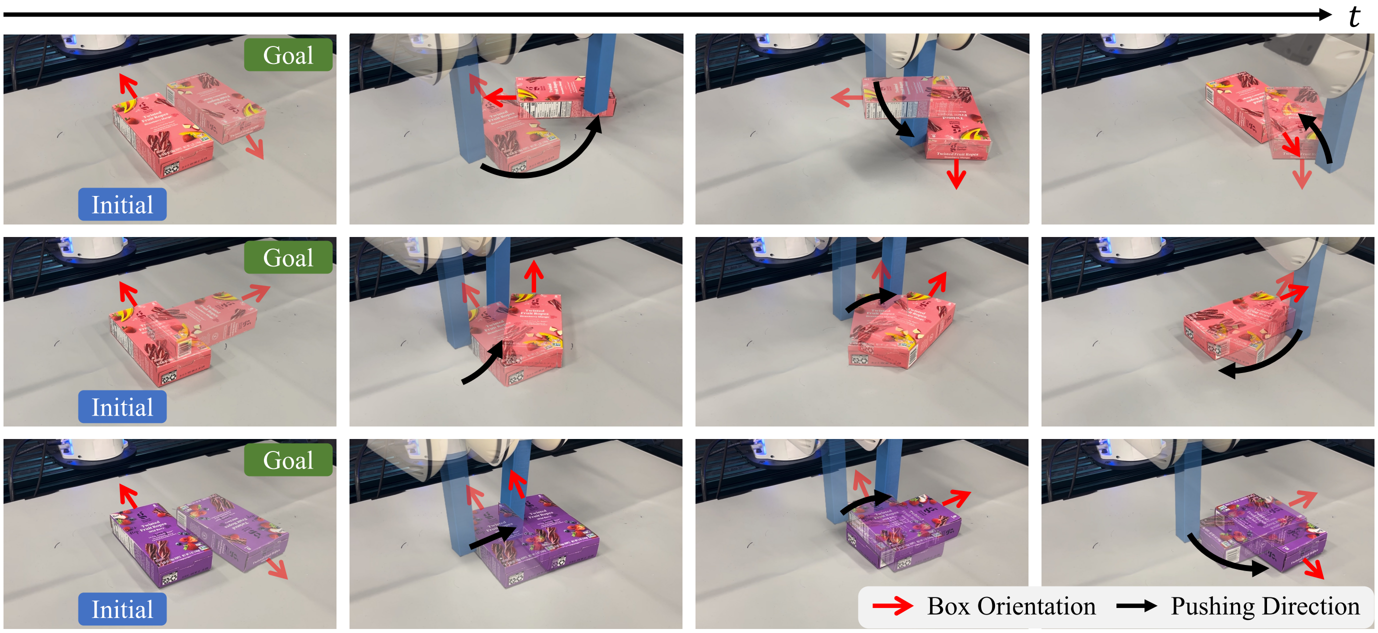


Fig. 7: **Non-prehensile box pushing.** We demonstrate our robot can push a box with unknown mass distribution from a starting pose to a target pose. Note that our box pushing is non-prehensile because the in-hand object is not fixed. We show that our method can generalize to unseen initial and target box poses in the first two rows and also previously unseen box configurations in the third row. A green arrow indicates the box's orientation, so boxes in rows 1 and 3 are flipped vertically.

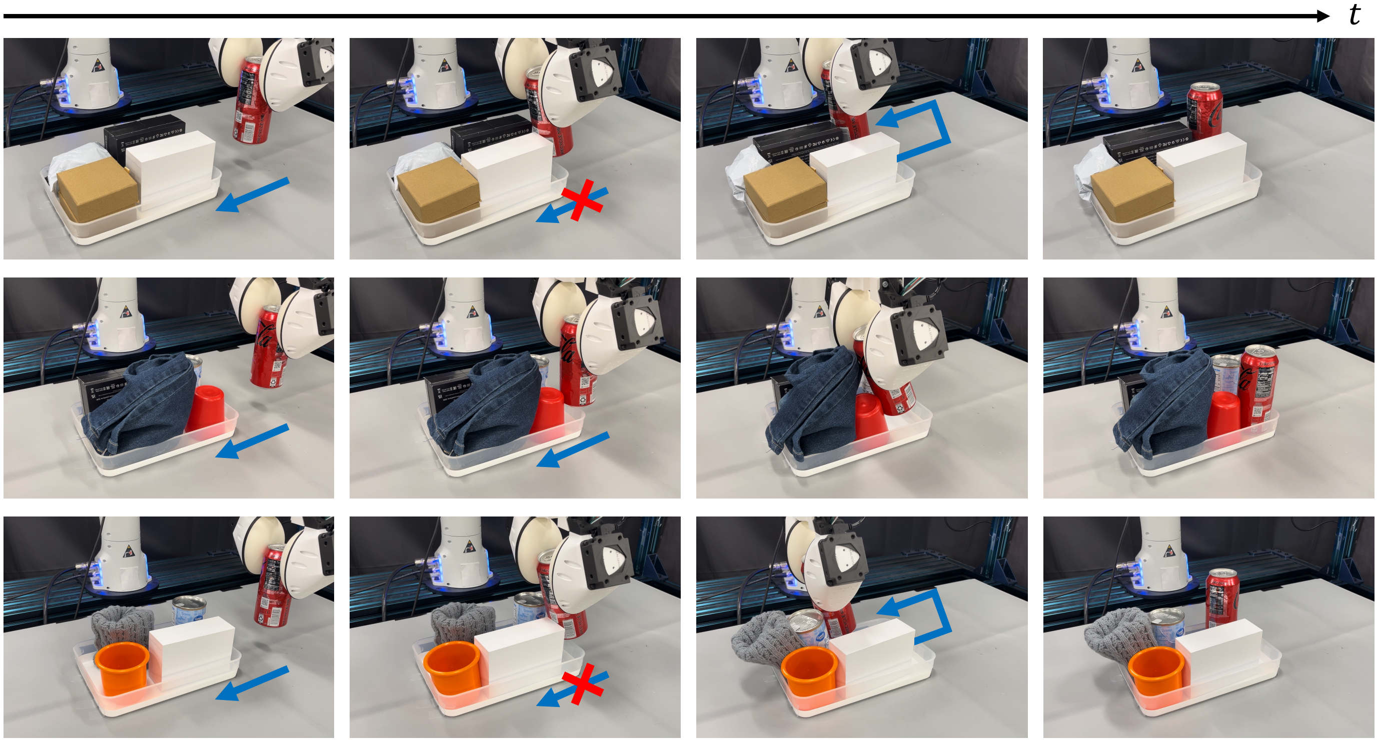


Fig. 8: **Dense packing with diverse object sets.** In the *Dense Packing* task, we demonstrate that RoboPack effectively identifies feasible insertion rows in a tray, minimizing excessive force on the robot to prevent hardware damage. The first row presents a set of objects from data collection, while subsequent rows illustrate our method's capability to adapt to objects with various visual appearances and different levels of deformability.