RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools

Haochen Shi[1*] **Huazhe Xu**[2*] **Samuel Clarke**[1] **Yunzhu Li**[1,3] **Jiajun Wu**[1]
[1]Stanford University [2]Tsinghua University, IIS [3]UIUC *Equal contribution
https://hshi74.github.io/robocook

Abstract: Humans excel in complex long-horizon soft body manipulation tasks via flexible tool use: bread baking requires a knife to slice the dough and a rolling pin to flatten it. Often regarded as a hallmark of human cognition, tool use in autonomous robots remains limited due to challenges in understanding tool-object interactions. Here we develop an intelligent robotic system, RoboCook, which perceives, models, and manipulates elasto-plastic objects with various tools. RoboCook uses point cloud scene representations, models tool-object interactions with Graph Neural Networks (GNNs), and combines tool classification with self-supervised policy learning to devise manipulation plans. We demonstrate that from just 20 minutes of real-world interaction data per tool, a general-purpose robot arm can learn complex long-horizon soft object manipulation tasks, such as making dumplings and alphabet letter cookies. Extensive evaluations show that RoboCook substantially outperforms state-of-the-art approaches, exhibits robustness against severe external disturbances, and demonstrates adaptability to different materials.

Keywords: Deformable Object Manipulation, Long-horizon Planning, Model Learning, Tool Usage

1 Introduction

Think about all the steps and tools a robot would need to use to make a dumpling from a lump of dough. This scenario contains three fundamental research problems in robotics: deformable object manipulation [1, 2, 3, 4], long-horizon planning [5, 6, 7, 8], and tool usage [9, 10, 11, 12]. The task poses significant challenges to the robot, because it involves decisions at both discrete (e.g., which tool to use) and continuous levels (e.g., motion planning conditioned on the selected tool).

To address these challenges, we propose RoboCook, a framework that perceives, models, and manipulates elasto-plastic objects for long-horizon tasks like making dumplings and alphabet letter cookies. RoboCook introduces three technical innovations. First, we apply a data-driven approach with a Graph Neural Network (GNN) [13, 14, 15] to learn highly complex interactions between the soft object and various tools purely from visual observations. Second, we combine a PointNet-based [16, 17] tool classification module with learned dynamics models to determine the most appropriate tool to use at the current task stage. Third, we use a self-supervised policy trained with synthetic data generated by our learned dynamics model for gripping, rolling, and pressing to improve performance and speed, and hand-coded policies for other skills.

We carry out comprehensive evaluations to show RoboCook's effectiveness, robustness, and generalizability. Figure 1 shows a typical successful trial of making a dumpling. To showcase robustness, we apply external perturbations during real-time execution, and RoboCook still succeeds in making a dumpling. To demonstrate generalizability, we test RoboCook to make alphabet letter cookies and RoboCook outperforms four strong baselines by a significant margin. RoboCook can also generalize to various materials by making a particular shape on different materials without retraining.

7th Conference on Robot Learning (CoRL 2023), Atlanta, USA.

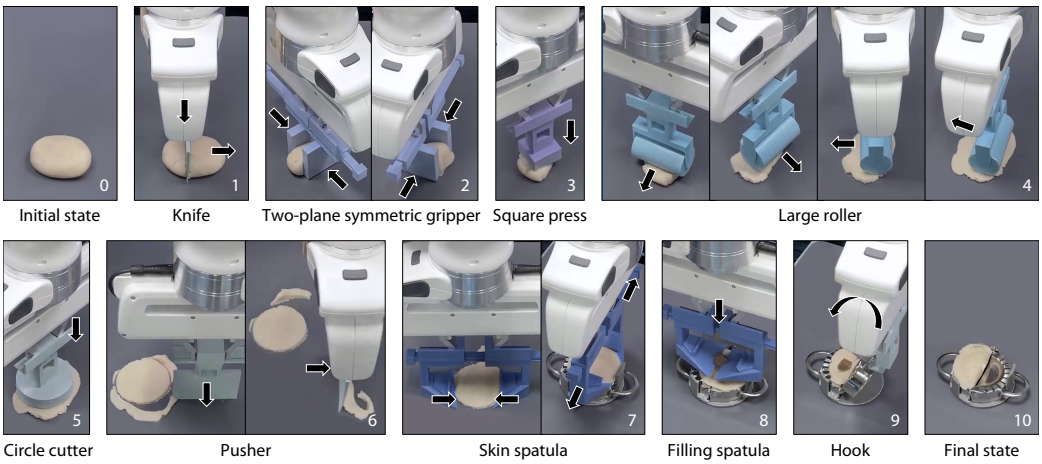

Figure 1: **Making dumplings.** RoboCook makes a dumpling from a piece of dough in nine steps: The robot (1) cuts the dough to an appropriate volume, (2) pinches the dough and regularizes the shape, (3) presses to flatten the dough, (4) rolls to flatten the dough further, (5) cuts a circular dumpling skin, (6) removes the excess dough, (7) picks and places the skin onto the mold, (8) adds the filling, and (9) closes and opens the mold. The black arrows denote the moving direction.

2 Related Work

Long-horizon deformable object manipulation. Real-world deformable object manipulation is a challenging task [18, 19, 20]. The innate high DoFs, partial observability, and non-linear local interactions make deformable objects hard to represent, model, and manipulate [21, 22, 23]. Conventional approaches in deformable object manipulation choose model-based methods [24, 25, 26, 27] or adaptive methods [28, 29, 30], yielding complex system identification problems. For example, Li et al. [27] proposes a differentiable simulator as the surrogate of real-world elasto-plastic objects. However, these simulators are based on approximate modeling techniques, resulting in a noticeable sim-to-real gap that hinders their application to real-world scenarios . There have been efforts toward learning to manipulate from expert demonstrations [31, 32, 33]. This paradigm is used in manipulating liquid [11], sand [29], and dough [34]. Despite these successes, obtaining the demonstration is expensive and sometimes prohibitive. Another recent trend is to learn a dynamics model from high-dimensional sensory data directly for downstream manipulation tasks [18, 35, 36, 37, 15, 38]. While these approaches show impressive results, they only consider short-horizon tasks using just one tool. In contrast, long-horizon tasks like dumpling-making require a reactive planner to understand the long-term physical effect of different tools to make the most effective discrete (e.g., which tool to use) and continuous (e.g., the action parameters) action decisions.

Tool usage. Tool usage is widely studied in cognitive science and robotics research [39, 40]. To study the process of human evolution, many researchers endorse tool usage as a main benchmark to evaluate the intelligence of living primates with respect to that of extinct hominids [41, 42, 43]. To equip robots with the same capability of tool usage as humans, prior works focus on teaching the robot the representation and semantics of tools or objects that potentially function like tools for downstream policy learning [11, 12]. To perform complicated long-horizon tasks composed of several subtasks, prior works also use human demonstrations to learn a hierarchical policy network [44]. In this work, we will use real-world robot random-play data, which is cheaper to acquire than human demonstration data. We train the robot on these self-exploratory trials to understand the physical interactions between different tools and deformable objects.

3 Method

The RoboCook framework has three major components: perception, dynamics, and closed-loop control. We first use an effective sampling scheme and intuitive tool representations for particle-

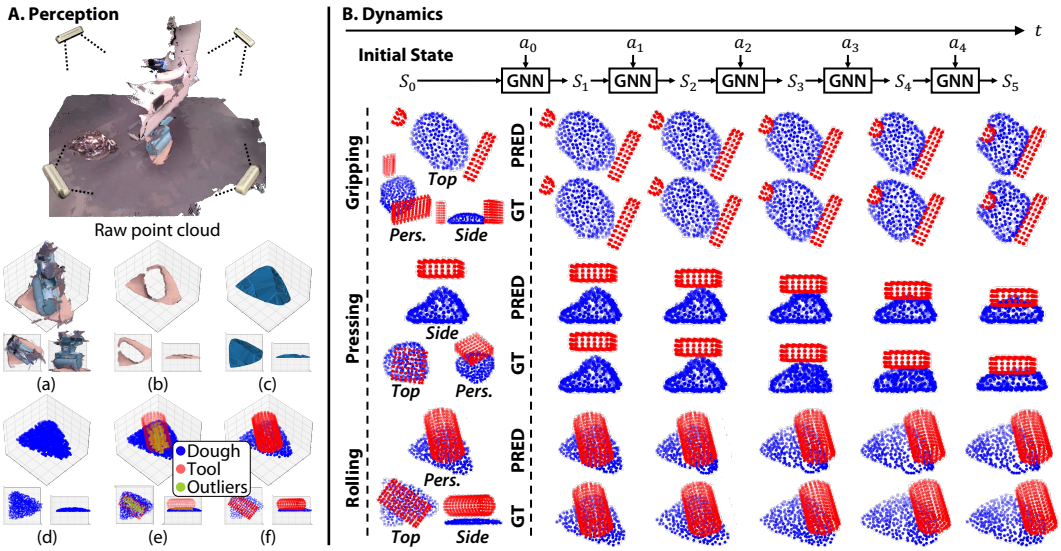

Figure 2: **Perception and dynamics of RoboCook**. **(A)** The input to the perception module is a point cloud of the robot's workspace captured by four RGB-D cameras. From the raw point cloud, we (a) crop the region of interest, (b) extract the dough point cloud, (c) reconstruct a watertight mesh (d) use the Signed Distance Function (SDF) to sample points inside the mesh, (e) remove points within the tools' SDF, and (f) sample 300 surface points. **(B)** We process videos of each tool manipulating the dough into a particle trajectory dataset to train our GNN-based dynamics model. The model can accurately predict long-horizon state changes of the dough in gripping, pressing, and rolling tasks. On the left are the initial states' perspective, top, and side views, and on the right is a comparison of model predictions and ground truth states.

based scene representation. Second, we train Graph Neural Networks (GNNs) as the dynamics model from the processed video dataset to accurately predict dough states during manipulation. Last, a tool classifier selects the best tool for each substage in a long-horizon task and a self-supervised policy network performs closed-loop control.

3.1 Perception

The perception module aims to sample particles sparsely and uniformly for the downstream dynamics model. This task is challenging because of dough occlusions from the robot and tool and self-occlusions from the irregular and concave shape of the dough.

We merge point clouds from four calibrated RGB-D cameras and perform color segmentation to extract the dough point cloud. Then we apply either Poisson surface reconstruction [45] or alpha-shape surface reconstruction [46] to reconstruct a watertight surface of the dough, depending on occlusion levels. In heavy occlusion cases, alpha-shape surface reconstruction is usually worse at capturing concavities than Poisson surface reconstruction. However, we use it to secure a complete and singular mesh. With few or no occlusions, we combine Poisson surface reconstruction and the MeshFix algorithm [47] to generate a watertight mesh. We use the watertight mesh's Signed Distance Function (SDF) to sample points inside it randomly. To compensate for details lost during surface reconstruction, we apply a voxel-grid filter to reconstruct concavities by removing the points above the original dough point cloud. We also remove the noisy points penetrating the tool using the mesh's SDF. We compute the tool mesh's SDF from the robot's end-effector pose (recorded during data collection) and the ground-truth tool mesh. We then perform alpha-shape surface reconstruction and use Poisson disk sampling [48] to sample 300 points uniformly on the surface, capturing more details of the dough with a fixed particle number.

For different tools, we uniformly sample particles on the surface of their ground truth mesh to reflect their geometric features. The complete scene representation for the downstream dynamics model

concatenates dough and tool particles. Section 6.2.1 and 6.2.2 of supplementary materials describe more data collection and preprocessing details.

3.2 Dynamics Model

Our GNN-based dynamics model predicts future states of dough based on the current state and actions of the tools with only 20 minutes of real-world data per tool. The rigid and non-rigid motions are each predicted by a multi-layer perceptron (MLP) and added together as the predicted motion by the GNN.

The graph of sampled particles at each time step is represented as $s_t = (\mathcal{O}_t, \mathcal{E}_t)$ with $\mathcal{O}_t$ as vertices, and $\mathcal{E}_t$ as edges. For each particle, $\mathbf{o}_{i,t} = \left(\mathbf{x}_{i,t}, \mathbf{c}^o_{i,t}\right)$, where $\mathbf{x}_{i,t}$ is the particle position i at time t, and $\mathbf{c}^o_{i,t}$ is the particle's attributes at time t, including the particle normal and group information (i.e., belongs to the dough or the tool). The edge between a pair of particles is denoted as $\boldsymbol{e}_k = (u_k, v_k)$, where $1 \leq u_k, v_k \leq |\mathcal{O}_t|$ are the receiver particle index and sender particle index respectively, and k is the edge index. Section 6.3.1 of supplementary materials provides more details on graph building.

Following the previous work [18], we use a weighted function of Chamfer Distance (CD) [49] and Earth Mover's Distance (EMD) [50] as the loss function to train the dynamics model:

$$\mathcal{L}(\mathcal{O}_t, \widehat{\mathcal{O}}_t) = w_1 \cdot \mathcal{L}_{\text{CD}}(\mathcal{O}_t, \widehat{\mathcal{O}}_t) + w_2 \cdot \mathcal{L}_{\text{EMD}}(\mathcal{O}_t, \widehat{\mathcal{O}}_t), \tag{1}$$

where $\mathcal{O}_t$ is the real observation, and $\widehat{\mathcal{O}}_t$ is the predicted state. We select $w_1 = 0.5$ and $w_2 = 0.5$ based on the experiment results. To be more specific, the CD between $\mathcal{O}_t, \widehat{\mathcal{O}}_t \subseteq \mathbb{R}^3$ is calculated by

$$\mathcal{L}_{\text{CD}}(\mathcal{O}_t, \widehat{\mathcal{O}}_t) = \sum_{\mathbf{x} \in \mathcal{O}_t} \min_{\mathbf{y} \in \widehat{\mathcal{O}}_t} \|\mathbf{x} - \mathbf{y}\|_2^2 + \sum_{\mathbf{y} \in \widehat{\mathcal{O}}_t} \min_{\mathbf{x} \in \mathcal{O}_t} \|\mathbf{x} - \mathbf{y}\|_2^2. \tag{2}$$

The EMD matches distributions of point clouds by finding a bijection $\mu : \mathcal{O}_t \to \widehat{\mathcal{O}}_t$ such that

$$\mathcal{L}_{\text{EMD}}(\mathcal{O}_t, \widehat{\mathcal{O}}_t) = \min_{\mu : \mathcal{O}_t \to \widehat{\mathcal{O}}_t} \sum_{\mathbf{x} \in \mathcal{O}_t} \|\mathbf{x} - \mu(\mathbf{x})\|_2. \tag{3}$$

We train the model to predict multiple time steps forward to regularize training and stabilize long-horizon future predictions. The training loss is calculated as a sum of distances between predictions and ground-truth states:

$$\mathcal{L}_{\text{train}} = \sum_{i=0}^{s} \mathcal{L}(\mathcal{O}_{t+i}, \widehat{\mathcal{O}}_{t+i}), \tag{4}$$

where the dynamics model takes $\widehat{\mathcal{O}}_{t+i-1}$ as the input to predict $\widehat{\mathcal{O}}_{t+i}$ when $i > 0$. The model performs better in inference time by predicting a slightly longer horizon during training time. Empirically, we discover that $s = 2$ is good enough for inference-time prediction accuracy, and a larger s does not give us much gain on the accuracy. Note that although we use $s = 2$ during training, we predict 15 steps during inference time, which is a long-horizon prediction. This highlights the inductive bias of our GNN-based dynamics model - we only need to train on short-horizon predictions to generalize to a much longer-horizon prediction during inference. Another motivation to keep s small is to increase the training speed. More implementation details on model training can be found in Section 6.3.2 of supplementary materials.

3.3 Closed-Loop Control

To solve long-horizon manipulation tasks involving multiple tools, the robot must determine (1) the best tool to use at each stage and (2) the optimal trajectory for the selected tool.

To answer the first question, we implement a tool classification module based on PointNet++ [17] that takes a concatenation of the input and output states of the dough as input and outputs probabilities of the 15 tools achieving the target state. We extract the features of the input and output point clouds separately with a PointNet++ classification layer. Then we feed the concatenated features to an MLP with a SoftMax activation layer to output probabilities.

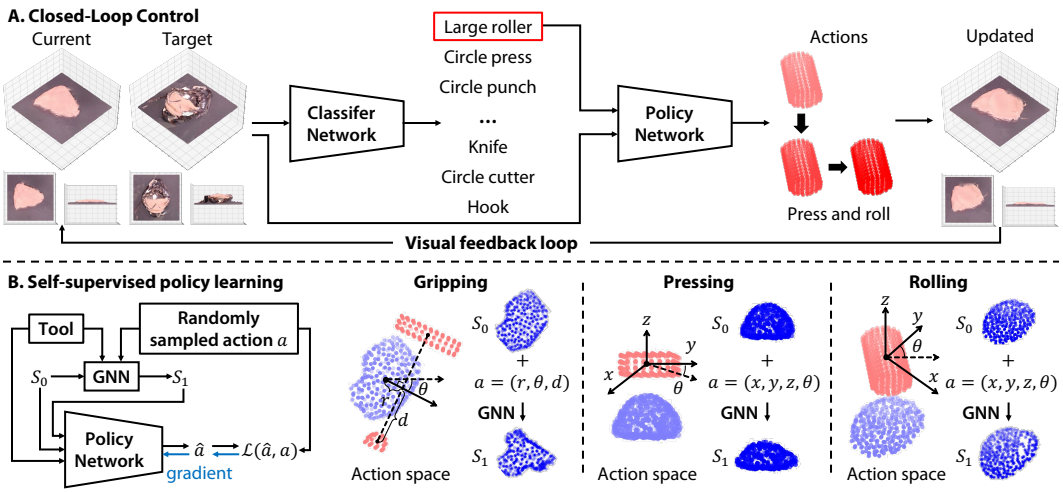

Figure 3: **Planning of RoboCook. (A)** PointNet-based classifier network identifies appropriate tools based on the current observation and the target dough configuration. The self-supervised policy network takes the tool class, the current observation, and the target dough configuration as inputs and outputs the manipulation actions. The framework closes the control loop with visual feedback. **(B)** We show the policy network architecture, the parametrized action spaces of gripping, pressing, and rolling, and how we generate the synthetic datasets to train the policy network.

The three tools with the highest probabilities are selected as candidates, and their optimal actions are planned. The robot selects the tool yielding the final state closest to the target and executes the planned actions. We let the module output three tool candidates instead of only one to increase our pipeline's error tolerance to the tool classifier.

To address the second question, we specify and parameterize the action space of the 15 tools based on human priors. Then we classify the tools into a few categories based on $|\mathcal{A}|$, the dimension of their action space. Section 6.4.1 of supplementary materials provides more details on the specification and justification of tool action spaces.

For grippers, rollers, presses, and punches, we collect data in the real world by random exploration in the bounded action space and learn their interaction model with the dough using a GNN. Then we generate a synthetic dataset with the dynamics model, train a self-supervised policy network to obtain the optimal policy, and execute closed-loop control with visual feedback. We use straightforward planners for other tools based on human priors. More implementation details can be found in Section 6.4.1 of supplementary materials.

To generate the synthetic dataset, we reuse the initial dough states acquired during data collection and randomly sample actions in the parameterized action space of each tool. As shown in Figure 3(A), the dough states before and after applying actions are the input, and the action parameters are the labels. We design a self-supervised policy network based on PointNet++. Given the continuous solution space of the action parameters, we formulate a multi-bin classification problem inspired by previous works on 3D bounding box estimation [51, 52]. First, we concatenate input and output point clouds, labeling them 0 or 1 for distinction. Next, we extract features using a PointNet++ layer. Finally, we feed the feature vector into separate classification and regression heads, both being MLPs with identical structures. The learned policy drastically improves the planning efficiency, and the complete pipeline only takes around 10 seconds of planning time to make a dumpling. More implementation details are described in Section 6.4.2 of supplementary materials.

During closed-loop control, cameras capture the workspace point cloud, which the perception module processes to obtain a clean and sparse point cloud of the dough. Our method takes the current observation and final target as input and returns the best tool and optimal actions, as shown in Figure 3(B). Then the robot picks up the selected tool, manipulates the dough, lifts its hand to avoid occluding the camera views, and plans the next move with visual feedback.

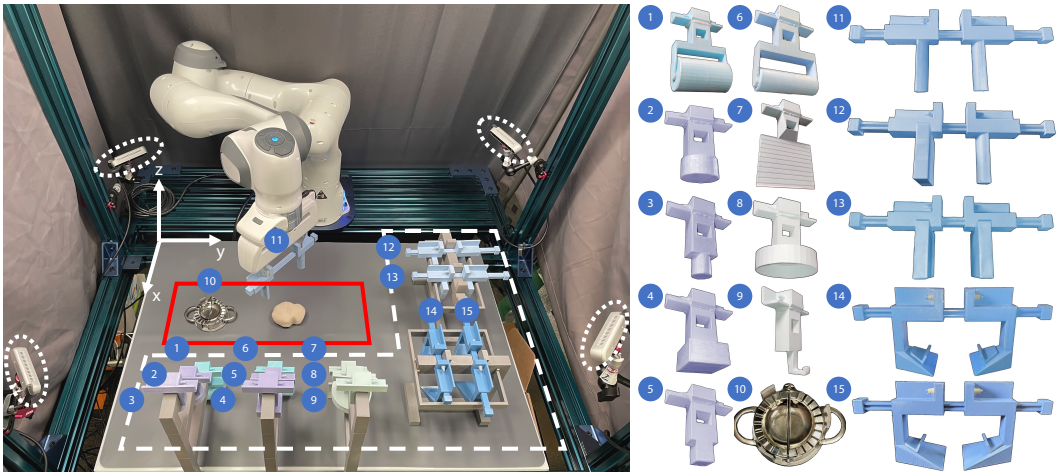

Figure 4: **RoboCook hardware and setup.** Left: Robot's tabletop workspace with xyz coordinates at top-left. Dashed white circles: four RGB-D cameras mounted at four corners of the table. Red square: dough location and manipulation area. Dashed white square: tool racks. Right: 15 tools: (1) large roller, (2) circle press, (3) circle punch, (4) square press, (5) square punch, (6) small roller, (7) knife/pusher, (8) circle cutter, (9) hook, (10) dumpling mold, (11) two-rod symmetric gripper, (12) asymmetric gripper, (13) two-plane symmetric gripper, (14) skin spatula, (15) filling spatula. Tools are 3D-printed, representing common dough manipulation tools. Section 6.5.2 of supplementary materials discusses the design principles of these 3D-printed tools.

4 Experiments

In this study, we design and implement a hardware setup for soft body manipulation tasks, allowing easy selection and interchange of 15 tools, as shown in Figure 4. We demonstrate the RoboCook framework's effectiveness in long-horizon soft object manipulation tasks that require the use of multiple tools, such as making a dumpling from a randomly shaped dough and shaping alphabet letter cookies to compose the word 'RoboCook.' Additionally, RoboCook can complete these tasks under extensive human interference, including significant changes to the shape and volume, as shown in the second supplementary video, demonstrating its robustness to external perturbations. We discuss the experiment setup in Section 6.5 of supplementary materials.

4.1 Making Dumplings

The dumpling-making task is to manipulate a piece of dough and the given filling into a dumpling shape. The main challenge lies in choosing appropriate tools and planning effective action trajectories. We consider the task successful if the dumpling skin is thin enough and completely covers the filling. Dumpling-making is a highly challenging task—even a single error might break the entire pipeline. Our method reliably accomplishes the task as shown in Figure 1. We show a comparison with manipulation results of human subjects in Section 6.6 of supplementary materials.

RoboCook also demonstrates robustness against external perturbations during real-time execution. At each stage of dumpling making, a human disturbs the robot by deforming the dough, adding additional dough, or even replacing the dough with a completely different piece to deviate from the trajectory of the robot's original plan. For example, at the rolling stage, the human folds the flattened dough into a bulky shape. The robot decides to roll again to get a flatter dumpling skin. In a more challenging case, the human replaces the round dumpling skin with a completely different dough in a highly irregular shape. After this perturbation, the robot puts down the roller, picks up the knife to start again from the beginning, and successfully makes a dumpling. The complete process is shown in the second supplementary video. We also show a quantitative evaluation of the tool classification network in Section 6.7 of supplementary materials.

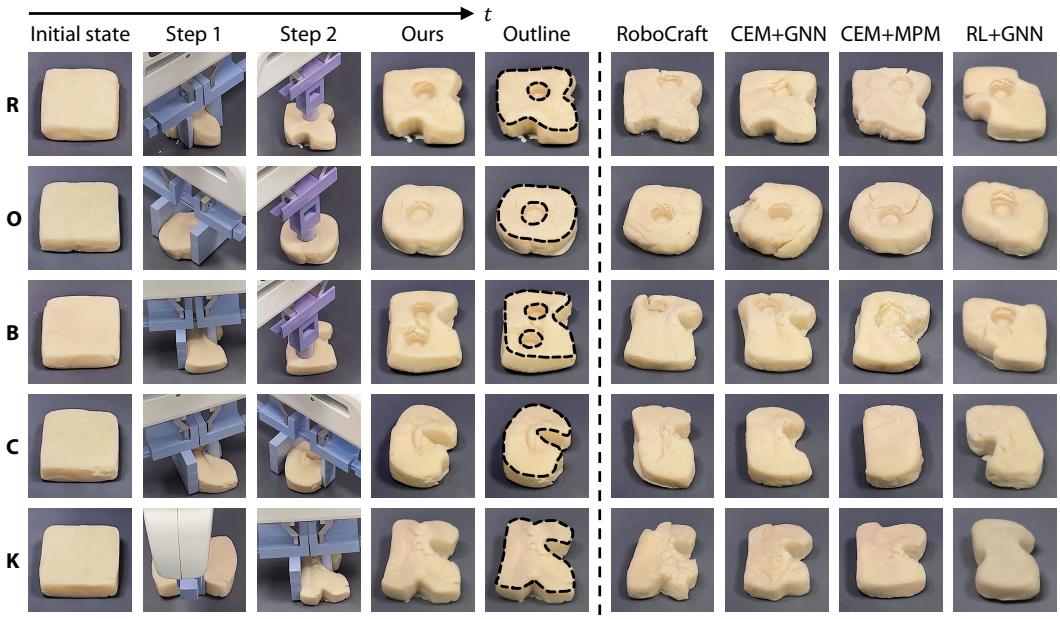

Figure 5: **Making alphabetical letter cookies.** We list R, O, B, C, and K shaping steps in Columns 1 to 4. Column 5 manually highlights the contour of the alphabetical letters. Columns 6 through 9 compare our self-supervised learned policy with three model-based planning baselines and one RL baseline. Our method can shape the dough closer to the target than all four baseline methods.

Methods	CD ↓	EMD ↓	CD of normal ↓	Human evaluation ↑	Planning time (s) ↓
Ours	**0.0062** $\pm$ **0.0007**	**0.0042** $\pm$ **0.0006**	**0.1933** $\pm$ **0.0345**	**0.90** $\pm$ **0.11**	**9.3** $\pm$ **1.5**
RoboCraft	0.0066 $\pm$ 0.0005	0.0044 $\pm$ 0.0006	0.2011 $\pm$ 0.0329	0.54 $\pm$ 0.43	613.7 $\pm$ 202.7
CEM+GNN	0.0066 $\pm$ 0.0007	0.0045 $\pm$ 0.0008	0.2043 $\pm$ 0.0431	0.52 $\pm$ 0.41	756.0 $\pm$ 234.5
CEM+MPM	0.0070 $\pm$ 0.0007	0.0046 $\pm$ 0.0006	0.1965 $\pm$ 0.0265	0.48 $\pm$ 0.35	1486.7 $\pm$ 512.8
RL+GNN	0.0077 $\pm$ 0.0007	0.0064 $\pm$ 0.0009	0.2041 $\pm$ 0.0414	0.17 $\pm$ 0.09	1867.9 $\pm$ 190.3

Table 1: **Quantitative evaluations.** We use CD and EMD between the point clouds and the CD between the surface normals to evaluate the results. We further profile how long these methods take to plan actions. Our method outperforms all baseline methods in these metrics by a large margin.

4.2 Making Alphabet Letter Cookies

The RoboCook framework demonstrates effectiveness and robustness in highly complicated manipulation tasks such as dumpling-making. This section explores its generalization ability in tasks requiring precise actions such as shaping alphabet letter cookies [18] without additional training.

Figure 5 shows that the RoboCook framework can accurately shape letters R, O, B, C, and K to compose the word 'RoboCook.' For R, the robot uses the two-rod symmetric gripper for cavities and the circle punch for the hole. For O, the tool classifier selects a two-plane symmetric gripper to pinch the square into a circle and the circle punch for the hole. Our method identifies the asymmetric gripper as suitable for B and locates accurate positions for the circle punch to press twice. Shaping C is more challenging due to the large distance between the initial and target shapes, but our method successfully plans gripping positions using closed-loop visual feedback. The robot combines the two-rod symmetric and asymmetric gripper for K to create cavities.

We compare our method with four strong baselines: (1) limited-memory BFGS [53] with GNN-based dynamics model (RoboCraft) [18]; (2) cross-entropy method (CEM) [54] with GNN-based dynamics model; (3) CEM with a Material Point Method (MPM) simulator [55]; and (4) Reinforcement Learning with GNN-based dynamics model. Qualitative results are shown in Figure 5. We include a detailed analysis of our results in Section 6.10.

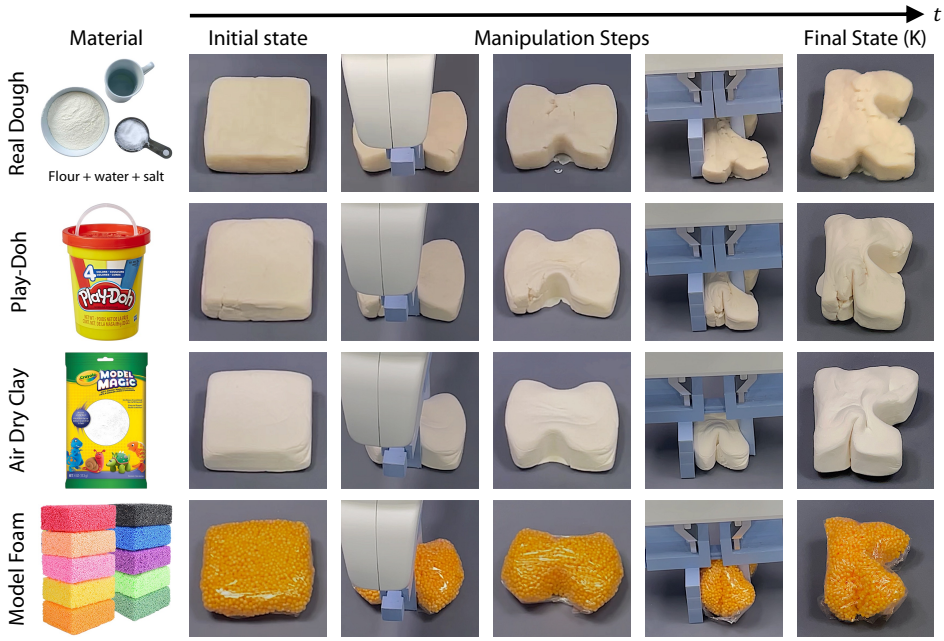

Figure 6: **Generalizing to different materials**. We showcase the dynamics model's capability to generalize to various materials by shaping a K without retraining.

Table 1 evaluates the results using Chamfer Distance (CD) [49], Earth Mover's Distance (EMD) [50] and CD between top surface normals. However, we recognize a discrepancy between how these metrics measure the results and how humans perceive them - these metrics are prone to local noises while humans are good at capturing the holistic features of the dough. Therefore, we show the prediction accuracy of 100 human subjects on recognizing the letters for these five methods in Table 1. Another highlight of our method is its speed. Since the policy network only needs one forward prediction to output the actions for each tool, it is significantly faster than other methods that rely on forwarding the dynamics models many times. Section 6.8 and 6.9 of supplementary materials discuss more details about the human study and the baseline implementations.

We show that the dynamics model can generalize to various materials by shaping a K without retraining in Figure 6. We test on Play-Doh, Air Dry Clay, and Model Foam, each displaying notably different dynamics compared to our original dough made of flour, water, and salt. The dynamics model is trained solely on interaction data with the real dough.

5 Conclusion and Limitations

RoboCook demonstrates its effectiveness, robustness, and generalizability in elasto-plastic object manipulation with a general-purpose robotic arm and everyday tools. The main contributions of RoboCook include (1) tool-aware GNNs to model long-horizon soft body dynamics accurately and efficiently, (2) a tool selection module combined with dynamics models to learn tool functions through self-exploratory trials, and (3) a self-supervised policy learning framework to improve the performance and speed significantly. RoboCook pioneers solutions for tool usage and long-horizon elasto-plastic object manipulation in building a generic cooking robot.

One limitation of RoboCook is the occasional failure of dough sticking to the tool. A solution is to design an automatic error correction system. RoboCook also relies on human priors of tool action spaces to simplify planning. But these simplifications do not constrain generalization as they can be easily specified for new tools. Section 6.4.1 provides more justifications for this. Another limitation is that humans define the subgoals. Higher-level temporal abstraction and task-level planning are required to get rid of them. Finally, RoboCook requires additional topology estimation to apply to cables and cloths [56], which is beyond the focus of this work.

Acknowledgments

This work was in part supported by AFOSR YIP FA9550-23-1-0127, ONR MURI N00014-22-1-2740, the Toyota Research Institute (TRI), the Stanford Institute for Human-Centered AI (HAI), JPMC, and Analog Devices.

References

[1] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 734–743. PMLR, 2018.

[2] X. Lin, Y. Wang, J. Olkin, and D. Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 432–448. PMLR, 2021.

[3] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. Li, et al. Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine*, 29(3):67–77, 2022.

[4] H. Yin, A. Varava, and D. Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 6(54):eabd8803, 2021.

[5] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint. Long-horizon multi-robot rearrangement planning for construction assembly. *IEEE Transactions on Robotics*, 2022.

[6] S. Nair and C. Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*.

[7] S. Pirk, K. Hausman, A. Toshev, and M. Khansari. Modeling long-horizon tasks as sequential interaction landscapes. In *Conference on Robot Learning*, pages 471–484. PMLR, 2021.

[8] A. Simeonov, Y. Du, B. Kim, F. Hogan, J. Tenenbaum, P. Agrawal, and A. Rodriguez. A long horizon planning framework for manipulating rigid pointcloud objects. In *Conference on Robot Learning*, pages 1582–1601. PMLR, 2021.

[9] A. Billard and D. Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446): eaat8414, 2019.

[10] N. Yamanobe, W. Wan, I. G. Ramirez-Alpizar, D. Petit, T. Tsuji, S. Akizuki, M. Hashimoto, K. Nagata, and K. Harada. A brief review of affordance in robotic manipulation research. *Advanced Robotics*, 31(19-20):1086–1101, 2017.

[11] D. Seita, Y. Wang, S. J. Shetty, E. Y. Li, Z. Erickson, and D. Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. In *6th Annual Conference on Robot Learning*, 2022.

[12] A. Xie, F. Ebert, S. Levine, and C. Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019. doi:10.15607/RSS.2019.XV.001.

[13] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[14] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2018.

[15] 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.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[17] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[18] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu. RoboCraft: Learning to See, Simulate, and Shape Elasto-Plastic Objects with Graph Networks. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022. doi:10.15607/RSS.2022.XVIII.008.

[19] C. Matl and R. 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. IEEE, 2021.

[20] X. Lin, C. Qi, Y. Zhang, Z. Huang, K. Fragkiadaki, Y. Li, C. Gan, and D. Held. Planning with spatial-temporal abstraction from point clouds for deformable object manipulation. In *6th Annual Conference on Robot Learning*, 2022.

[21] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint. Learning multi-object dynamics with compositional neural radiance fields. In *Conference on Robot Learning*, pages 1755–1768. PMLR, 2023.

[22] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song. Iterative Residual Policy for Goal-Conditioned Dynamic Manipulation of Deformable Objects. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022. doi:10.15607/RSS.2022.XVIII.016.

[23] H. Ha and S. Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robot Learning*, pages 24–33. PMLR, 2022.

[24] Z. Huang, Y. Hu, T. Du, S. Zhou, H. Su, J. B. Tenenbaum, and C. Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. In *International Conference on Learning Representations*, 2020.

[25] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. In *International Conference on Learning Representations*, 2021.

[26] A.-M. Cretu, P. Payeur, and E. M. Petriu. Soft object deformation monitoring and learning for model-based robotic hand manipulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):740–753, 2011.

[27] S. Li, Z. Huang, T. Du, H. Su, J. B. Tenenbaum, and C. Gan. Contact points discovery for soft-body manipulations with differentiable physics. In *International Conference on Learning Representations*, 2021.

[28] D. Navarro-Alarcon, H. M. Yip, Z. Wang, Y.-H. Liu, F. Zhong, T. Zhang, and P. Li. Automatic 3-d manipulation of soft objects by robotic arms with an adaptive deformation model. *IEEE Transactions on Robotics*, 32(2):429–441, 2016.

[29] A. Cherubini, V. Ortenzi, A. Cosgun, R. Lee, and P. Corke. Model-free vision-based shaping of deformable plastic materials. *The International Journal of Robotics Research*, 39(14):1739–1759, 2020.

[30] K. Yoshimoto, M. Higashimori, K. Tadakuma, and M. Kaneko. Active outline shaping of a rheological object based on plastic deformation distribution. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1386–1391. IEEE, 2011.

[31] B. Balaguer and S. Carpin. Combining imitation and reinforcement learning to fold deformable planar objects. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1405–1412. IEEE, 2011.

[32] F. Nadon, A. J. Valencia, and P. Payeur. Multi-modal sensing and robotic manipulation of non-rigid objects: A survey. *Robotics*, 7(4):74, 2018.

[33] B. Jia, Z. Pan, Z. Hu, J. Pan, and D. Manocha. Cloth manipulation using random-forest-based imitation learning. *IEEE Robotics and Automation Letters*, 4(2):2086–2093, 2019.

[34] N. Figueroa, A. L. P. Ureche, and A. Billard. Learning complex sequential tasks from demonstration: A pizza dough rolling case study. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 611–612. Ieee, 2016.

[35] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[36] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[37] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR, 2018.

[38] T. Silver, R. Chitnis, A. Curtis, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11962–11971, 2021.

[39] C. P. Van Schaik and G. R. Pradhan. A model for tool-use traditions in primates: implications for the coevolution of culture and cognition. *Journal of Human Evolution*, 44(6):645–664, 2003.

[40] R. Holladay, T. Lozano-Pérez, and A. Rodriguez. Force-and-motion constrained planning for tool use. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7409–7416. IEEE, 2019.

[41] S. T. Parker and K. R. Gibson. Object manipulation, tool use and sensorimotor intelligence as feeding adaptations in cebus monkeys and great apes. *Journal of Human Evolution*, 6(7): 623–641, 1977.

[42] T. Ingold. Tool-use, sociality and intelligence. *Tools, language and cognition in human evolution*, 429(45):449–72, 1993.

[43] T. Matsuzawa. Primate foundations of human intelligence: a view of tool use in nonhuman primates and fossil hominids. In *Primate origins of human cognition and behavior*, pages 3–25. Springer, 2008.

[44] J. Liang, B. Wen, K. Bekris, and A. Boularias. Learning sensorimotor primitives of sequential manipulation tasks from visual demonstrations. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8591–8597. IEEE, 2022.

[45] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[46] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, 13(1):43–72, 1994.

[47] M. Attene. A lightweight approach to repairing digitized polygon meshes. *The visual computer*, 26(11):1393–1406, 2010.

[48] C. Yuksel. Sample elimination for generating poisson disk sample sets. *Computer Graphics Forum*, 34(2):25–32, 2015.

[49] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[50] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[51] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[52] A. Kundu, Y. Li, and J. M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3559–3568, 2018.

[53] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[54] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

[55] S. G. Bardenhagen and E. M. Kober. The generalized interpolation material point method. *Computer Modeling in Engineering and Sciences*, 5(6):477–496, 2004.

[56] Z. Huang, X. Lin, and D. Held. Mesh-based Dynamics with Occlusion Reasoning for Cloth Manipulation. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022. doi:10.15607/RSS.2022.XVIII.011.

6 Supplementary Materials

6.1 Inputs and Outputs of RoboCook's Respective Modules . . . 13
6.2 Perception . . . 14
6.2.1 Data Collection . . . 14
6.2.2 Data Preprocessing . . . 14
6.3 Dynamics Model . . . 14
6.3.1 Graph Building . . . 14
6.3.2 Model Training . . . 14
6.3.3 Building Synthetic Datasets . . . 14
6.4 Closed-loop Control . . . 15
6.4.1 Action Space . . . 15
6.4.2 Multi-bin Classification . . . 16
6.4.3 Subgoal Definitions . . . 16
6.5 Experiment Setup . . . 16
6.5.1 Robot and Sensors . . . 16
6.5.2 Tool Design . . . 16
6.5.3 Tool-Switching Setup . . . 17
6.6 Comparison with Human Subjects on Dumpling-making . . . 17
6.7 Tool Classification . . . 18
6.8 Human Evaluation of Alphabet Letters . . . 19
6.9 Baseline Implementation Details . . . 19
6.10 Analysis of Comparison with Baselines . . . 19

6.1 Inputs and Outputs of RoboCook's Respective Modules

Perception module: At each time step, the input is a raw observation point cloud merged from four RGBD cameras and the franka panda end-effector pose. The output is a clean and sparse surface point cloud representing the dough concatenated with a point cloud sampled on the tool mesh surface representing the action. We compute the pose of the tool point cloud from the end effector pose and the tool geometry.

GNN dynamics: Our GNN predicts S_{t+1} from S_t and a_{t+1} at each prediction step. The inference-time input is the initial state S_0 and $\{a_t, t = 1, 2, \ldots N\}$. The output is S_N. We select $N = 15$ for the gripping, pressing, and rolling tasks.

Tool predictor: The input is a point cloud of the current dough configuration and a point cloud of the final target (e.g., the dumpling point cloud). The output is the next tool to achieve the final target configuration.

Policy network: The input is a point cloud of the current dough configuration and the point cloud of the subgoal associated with the selected tool. We have a one-to-one mapping between subgoals and tools. For example, a rolling pin corresponds to a dough flattened by the rolling pin. The output is the action in the defined action space of the selected tool.

6.2 Perception

6.2.1 Data Collection

To train the GNNs, we collect around 20 minutes of data for each tool using a behavior policy that randomly samples parameters within a predefined action space. In each episode, the robot applies multiple action sequences to the soft object, and after the current episode, a human manually resets the environment. Humans reshape the dough into a bulky but not necessarily cubic shape after every five grips, three rolls, and three presses.

The data collected for each tool are as follows: (1) Asymmetric gripper / two-rod symmetric gripper / two-plane symmetric gripper: 60 episodes with five sequences per episode; (2) Circle press / square press / circle punch / square punch: 90 episodes with three sequences per episode; (3) Large roller / small roller: 80 episodes with three sequences per episode.

We collect point clouds before and after executing each sequence to train the tool classifier. However, we augment the training data by including any pair of these point clouds, not just consecutive pairs. For tools that don't require a GNN-based dynamics model, we execute a pre-coded dumpling-making pipeline ten times and add point clouds captured before and after using each tool in the pipeline to our dataset. Note that most tool selection data is a direct reuse of the dynamics data collected during the training of the dynamics model. This approach efficiently collects real-world tool selection data without needing extra exploration. We record the data collection process in the fourth supplementary video.

6.2.2 Data Preprocessing

When building the dataset to train the dynamics model, aside from the sampling process of the perception module at each time frame, we also want to leverage the continuity of the video data. Therefore, we introduce simple geometric heuristics into the physical environment for better frame consistency. First, if the operating tool is not in contact with the convex hull of the object point cloud, we use the same sampled particles from the previous frame. This also applies when the tool moves away from the object. Additionally, we subsample the original videos to ensure that each video in the dataset has the same number of frames (16 frames in practice).

6.3 Dynamics Model

6.3.1 Graph Building

When building the graph, the edges between the dough particles are constructed by finding the nearest neighbors of each particle within a fixed radius (in practice, 0.1 cm). The edges between the tool and dough particles are computed slightly differently. Instead of simply connecting to all the neighbors within the threshold, we limit the number of undirected edges between tool particles and the dough particles to at most four per tool particle to cut off the redundant edges in the graph neural network. Since all the node and edge features in the GNN are encoded in each particle's local neighborhood, our GNN is naturally translation-invariant and therefore can accurately predict the movement of the dough regardless of its absolute location in the world frame.

6.3.2 Model Training

We train the model with temporal abstraction to enhance performance and inference speed. For example, when $t = 0$, we train the model to predict the state of the dough at $t = 3$ directly instead of $t = 1$. This shortens the horizon, eases the task, and improves our model's inference speed by decreasing the number of forward passes needed for a full action sequence.

6.3.3 Building Synthetic Datasets

Since the synthetic dataset generated by the dynamics model is cheap, we can collect as much synthetic data as desired. Empirically, for example, we sample 128 random actions in the action

space of gripping for each of the 300 different initial states, so there are $128 * 300 = 38,400$ pairs of input and output point clouds to train the policy network. The random walk to generate the synthetic dataset starts from initial dough states acquired during dynamics data collection, which are not all block-shaped states and cover various shapes.

6.4 Closed-loop Control

6.4.1 Action Space

We classify the tools into a few categories based on $|\mathcal{A}|$, the dimension of their corresponding action space. We visualize action spaces for gripping, pressing, and rolling in Figure 3 (B).

A) Nine tools that have an action space with $|\mathcal{A}| \geq 3$:

   1) Asymmetric gripper / two-rod symmetric gripper / two-plane symmetric gripper: $\{r, \theta, d\}$, where r is the distance between the midpoint of the line segment connecting the centers of mass of the gripper's two fingers and the center of the target object, θ is the robot gripper's rotation about the (vertical) axis, and d is the minimal distance between the gripper's two fingers during this pinch.
   2) Large roller / small roller / square press / square punch: $\{x, y, z, \theta\}$, where $\{x, y, z\}$ is the bounded location indicating the center of the action, and θ is the robot gripper's rotation about the vertical axis. In the case of rollers, the rolling distance is fixed and therefore not included in the action space.
   3) Circle press / circle punch: $\{x, y, z\}$, where $\{x, y, z\}$ is the bounded location indicating the center of the action. The robot gripper's rotation is unnecessary because the tool's bottom surface is a circle.

B) Five tools that have an action space with $|\mathcal{A}| = 2$:

   1) Knife / circle cutter / pusher / skin spatula / filling spatula: $\{x, y\}$, where $\{x, y\}$ is the bounded location indicating the center of the action on the plane. θ and z are fixed for these tools to simplify the action space.

C) The action of the hook is precoded.

In category B, for all tools except the knife, we leverage the prior that the center of the dough is always the optimal solution in the action space and directly compute the center from the processed point cloud. In the case of the knife, we use the y coordinate of the center of the dough as the solution for y (the xyz coordinate system is illustrated in Figure 4). For x, we first compute the volume of the target dough and then perform a binary search with the center of the dough as the starting point to find the cut position that results in the closest volume to the target volume.

In category C, the hook is precoded first to hook the handle of the dumpling mold, then close the mold, press the mold handle to turn the dough into a dumpling shape, and finally open the mold by hooking and lifting the handle.

The guiding principle in designing action spaces involves starting with the end-effector's 6-DoF action space and eliminating redundant DoFs. For instance, rotations along the x and y axes are typically not required to generate a meaningful action. Hence, we opt to exclude them from the action space of the 14 tools. For grippers, we transform the Cartesian coordinate system into a polar coordinate system to simplify the search process for action parameters since corner cases in the bounded Cartesian space are usually suboptimal. Following this, we introduce tool-specific DoFs, which are determined by the tool's geometric properties. For example, in the case of grippers, we incorporate an additional parameter, d, to represent the width between the gripper's two fingers.

Our method can potentially generalize to various challenging dough manipulation tasks besides dumpling-making, such as making alphabet letter cookies (as shown in the paper), pizza, and noodles. A successful transfer requires the ground truth meshes of new tools and data from interacting with them. We only need 20 minutes of real-world interaction data per tool, demonstrating the ease

of retraining for new tasks and tools. Although we incorporate human prior knowledge to simplify the action space for tools, it does not constrain the generalization capability since we can easily specify the action space for new tools. One limitation is that hand-designed action spaces may not be desirable in general manipulation settings. Our insight is that for most tasks, especially tasks where the robot uses a tool, there is much redundant and undesired space in the full 6-DoF action space. In most cases, humans follow certain rules and a constrained action space when using a specific tool. A future direction is to design a planner to automatically prune the action space conditioned on the tool and the task.

6.4.2 Multi-bin Classification

We formulate the self-supervised policy training as a multi-bin classification problem inspired by previous works on 3D bounding box estimation [51, 52]. The total loss for the multi-bin classification is

$$\mathcal{L} = \sum_{i=1}^{|\mathcal{A}|} \left(\mathcal{L}_{\text{conf}}^{\mathcal{A}_i} + w \cdot \mathcal{L}_{\text{loc}}^{\mathcal{A}_i} \right), \tag{5}$$

where the confidence loss $\mathcal{L}_{\text{conf}}^{\mathcal{A}_i}$ is the softmax loss of the confidences of each bin for each action parameter $\mathcal{A}_i$, and the localization loss $\mathcal{L}_{\text{loc}}^{\mathcal{A}_i}$ is the loss that tries to minimize the difference between the estimated parameter and the ground truth parameter. For orientation estimation, we use negative cosine loss as the localization loss and force it to minimize the difference between the ground truth and all the bins that cover that value. We use the smooth L1 loss as the localization loss for action parameters not representing an orientation. During inference time, for each parameter, the bin with maximum confidence is selected, and the final output is computed by adding the estimated delta of that bin to the center of the same bin.

To establish the bins, we first bound our action space into a reasonable range (A_{min}, A_{max}). Second, we define the number of bins N as a hyperparameter and select $N = 8$ for translations and $N = 32$ for rotations. Third, we divide the action space into $N + 1$ bins with size $2 * (A_{max} - A_{min})/N$. The center of the first bin is at A_{min}, the center of the last bin is at A_{max}, and each bin overlaps with neighboring bins. This approach is similar to [51].

6.4.3 Subgoal Definitions

As mentioned in 6.2.1, during data collection, we execute a hand-coded dumpling-making pipeline ten times and add point clouds captured before and after using each tool in the pipeline to our dataset as expert demonstrations. The point clouds recorded after using each tool in one of these trajectories are selected as subgoals.

6.5 Experiment Setup

6.5.1 Robot and Sensors

We use the 7-DoF Franka Emika Panda robot arm and its parallel jaw gripper as the base robot. Four calibrated Intel RealSense D415 RGB-D cameras are fixed on vertical metal bars around the robot tabletop, as shown in Figure 4. The cameras capture 1280×720 RGB-D images at 30 Hz. We also design a set of 3D-printed tools based on real-world dough manipulation tools.

6.5.2 Tool Design

We design and 3D-print 14 tools: large roller, small roller, circle press, circle punch, square press, square punch, knife / pusher, circle cutter, two-rod symmetric gripper, asymmetric gripper, two-plane symmetric gripper, skin spatula, filling spatula, and hook. The dumpling mold is the same as real-world ones. In Figure 7, we compare our 3D-printed tools and their real-world prototypes, which are common kitchen tools for dough manipulation. The design principle of these 3D-printed tools is to mimic real-world ones as closely as possible.

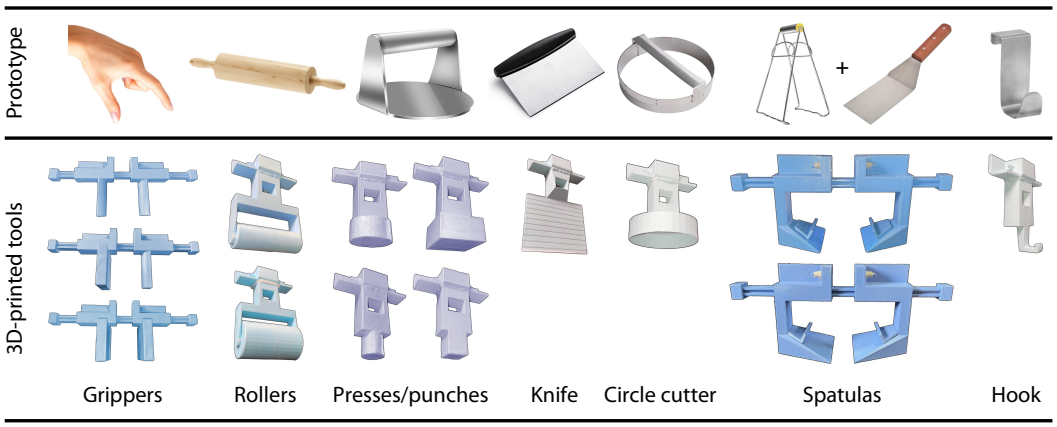

Figure 7: **Prototypes of 3D-printed tools**. We show a comparison between our 3D-printed tools and their real-world prototypes which are common kitchen tools for dough manipulation. The design principle of these 3D-printed tools is to mimic real-world ones as closely as possible. We use 3D-printed tools instead of real-world ones to allow the robot arm to acquire and manipulate the tools more easily.

The roller is composed of a holder and a rolling pin so that the rolling pin can rotate freely while the holder remains static. We designed both large and small rollers to accommodate different needs. We also have a set of punches and presses with square and circle shapes. The knife is a thin planar tool that can cut through objects. Similarly, the circle cutter can cut an object into a circular shape. Among the grippers, the two-rod symmetric gripper consists of two cylindrical extrusions, the asymmetric gripper consists of a cylindrical and planar part, and the two-plane symmetric gripper consists of two planar parts. The two extruding rods on each gripper insert into the corresponding holes of the two fingers of Franka's gripper, allowing them to adhere to and move along with the fingers. A linear shaft connects the two parts of each gripper, constraining their movement to a single axis. The skin and filling spatulas have a similar design, except that their two extrusions are each spatula, so they can pick up and put down the soft object without deforming it. The hook and the dumpling mold are tools used together to mold the dough into a dumpling shape.

6.5.3 Tool-Switching Setup

The tool-switching setup is an engineering innovation we implement in this project. We adopt two designs so that the robot can pick up, use, and put down the tools without any help from humans: (1) The connector on the top of each tool attaches firmly to the Franka's gripper when it closes its fingers and also unlocks easily when the gripper reopens. (2) The tool racks on the bottom and right side of the robot table hold all the 3D-printed tools in their upright poses so that the robot can easily pick up and put down the tools. Additionally, we calibrate the tools' world coordinates so that the robot knows where to find each tool. The supplementary videos of making dumplings show examples of how the robot switches tools.

6.6 Comparison with Human Subjects on Dumpling-making

We invited five human subjects to make dumplings with the same tools to highlight the complexity of dumpling-making. Each subject participated in two experiments: choosing tools independently and following a given tool sequence and subgoals. For a fair comparison, human subjects were not allowed to directly touch the dough with their hands or apply each tool more than five times. Before the experiments, we introduced each tool and gave them sufficient time to get familiar with the dough's dynamics and devise their plan. We compare their best attempt among three trials to our method for each experiment. Figure 9 shows that human performance is notably worse than our method without subgoals. Performance improves with the tool sequence and subgoals but remains comparable to or worse than our method. The fifth supplementary video records the entire process.

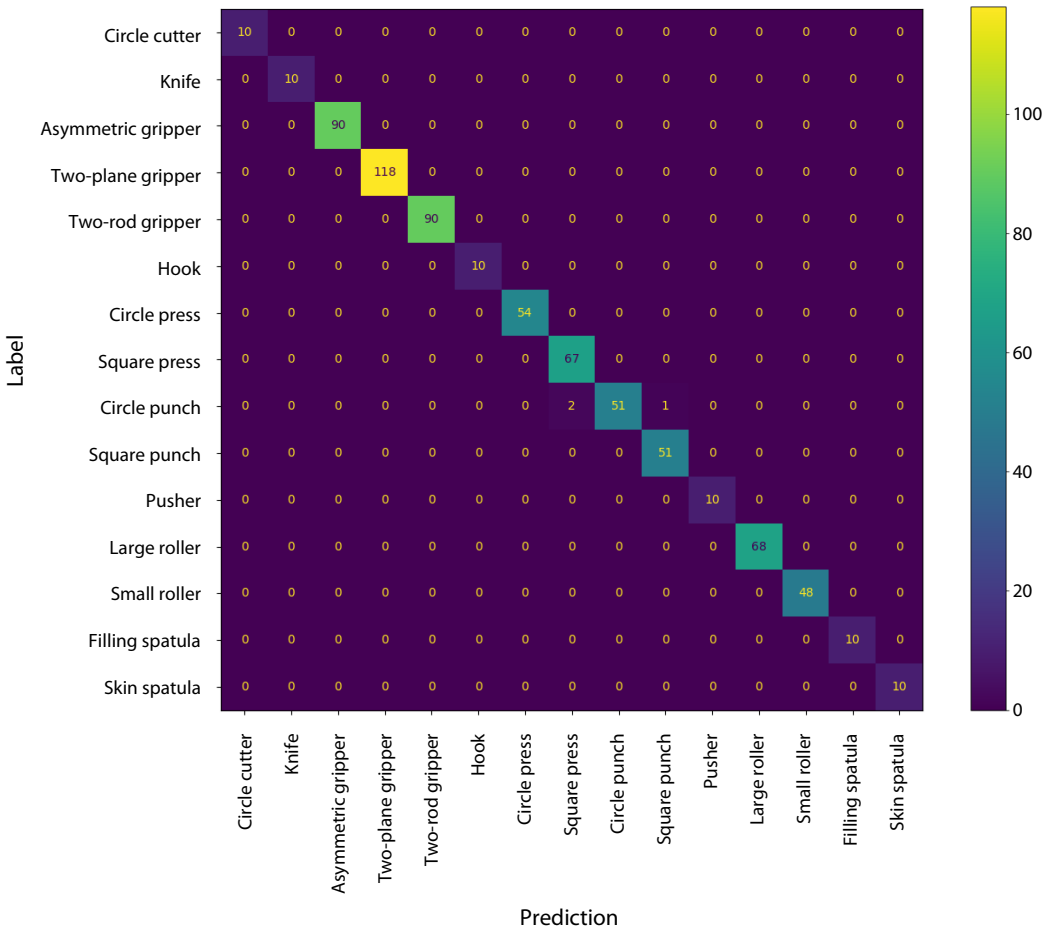

Figure 8: **Confusion matrix of the tool classifier predictions**. We show the confusion matrix of the tool classifier predictions on the test set, which is split from the training data. The tool classifier achieves an accuracy very close to 1.

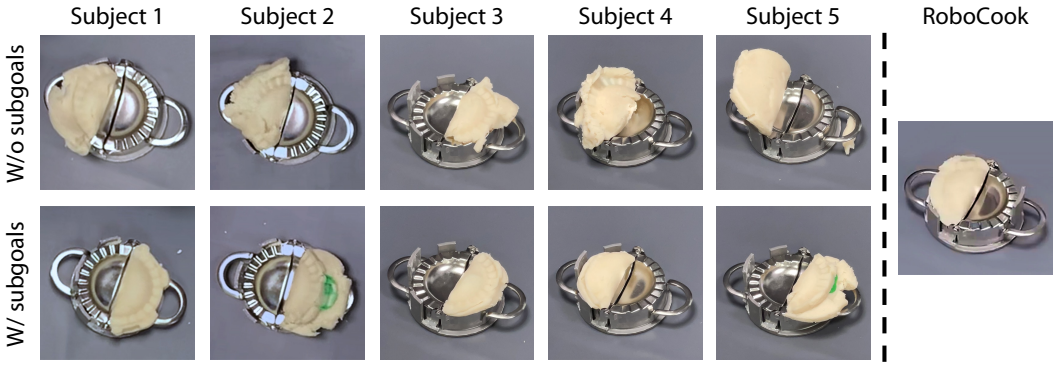

Figure 9: **Comparison with human subjects**. We show a comparison with the manipulation results of human subjects. In the first row, Human subjects devise their manipulation plan and choose tools independently. In the second row, human subjects follow a given tool sequence and subgoals.

6.7 Tool Classification

The training of our tool classifier is supervised learning with a cross-entropy loss as in standard classification architectures. We split a test set from the training data of the tool classifier and show

the confusion matrix of the tool classifier predictions in Figure 8. The instance accuracy is 0.996. We compared PointNet-based and ResNet-based architectures for the tool classification network. PointNet-base architecture generalizes better due to its ability to encode depth information. Empirically, it demonstrates greater robustness to changes in lighting, dough color, and dough transformations.

6.8 Human Evaluation of Alphabet Letters

We recognize a discrepancy between how metrics such as Chamfer Distance measure the results and how humans perceive them - these metrics are prone to local noises while humans are good at capturing the holistic features of the dough. Therefore, we invite 100 human subjects to evaluate the results. The human survey asks the question: "What alphabet letter is the robot trying to shape in the given image?" If we put all 20 images (four methods $\times$ five letters) in Question 1, there could be a predictive bias from seeing more than one image of the same letter. Therefore, we shuffle the order of 20 images and split them into four groups. Each group contains one image for each letter but from different methods. After averaging over five letters, we show the human perceived accuracy and human ranking of the performance of these four methods in Table 1.

6.9 Baseline Implementation Details

Baselines RoboCraft, CEM+GNN, and RL+GNN use the same framework as RoboCook but replace the policy network with gradient descent (RoboCraft), CEM, and RL. In other words, GNNs in these baselines are the same as GNNs we trained in RoboCook. They also use the same tool predictor as in RoboCook to handle this multi-tool task. They are meant to compare the policy network in RoboCook with alternatives.

The RL+GNN baseline utilizes a model-based Soft Actor-Critic (SAC) with a learned GNN-based dynamics model as the world model. The action space aligns with other planning methods, and the state space comprises the point cloud position. The reward function is derived from the change in Chamfer Distance after each grip. Training involves a discount factor of 0.99, a learning rate of 0.0003 with the Adam optimizer, 2-layer MLPs with 256 hidden units, and ReLU activation for both policy and critic models. We initially collect 250 steps of warm-up data. The replay buffer size is 1e6, and the target smooth coefficient is 0.005. We trained the RL baseline for 2500 steps online for each action prediction to fit into our closed-loop control module. The inputs and outputs of the RL baseline are the same as our policy network. The input is a point cloud of the current dough configuration and the point cloud of the subgoal. The output is the action in the defined action space of the selected tool. The results shown in Figure 5 and Table 1 indicate that the RL baseline is noticeably worse than our method.

6.10 Analysis of Comparison with Baselines

Our methods outperform baselines RoboCraft, CEM+GNN, and RL+GNN by a large margin, using the same GNNs and tool predictors but different planning methods. One reason is that our policy network sees a complete coverage of the entire action space from the large synthetic datasets generated by our dynamics model offline. Other planning methods explore the action space online and therefore have a smaller coverage than ours. By training a goal-conditioned policy network offline, we also make the online action synthesis much more efficient than baseline methods.

Another insight is that our PointNet-based policy network is better at capturing higher-level shape changes of the point cloud, such as concavities. For example, by comparing the current and target dough configuration, the policy network knows that the gripper should probably pinch the concave locations in the target dough configuration to deform the current dough into the target shape.

Our method also outperforms the baseline CEM+MPM since an MPM simulator suffers from a sim2real gap and relies on careful system identification to bridge the gap. Thus, the MPM simulator underperforms the GNN, which is directly trained on real-world data.