6 Supplementary Materials

6.1 Perception

6.1.1 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 resets the dough's shape into another random form.

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 the majority of 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.

6.1.2 Data Preprocessing

When building the dataset to train the dynamics model, aside from optimizing the sample quality 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).

6.2 Dynamics Model

6.2.1 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.

6.2.2 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 Closed-loop Control

6.3.1 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.

6.3.2 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 [50, 51]. 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.

6.4 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.

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.

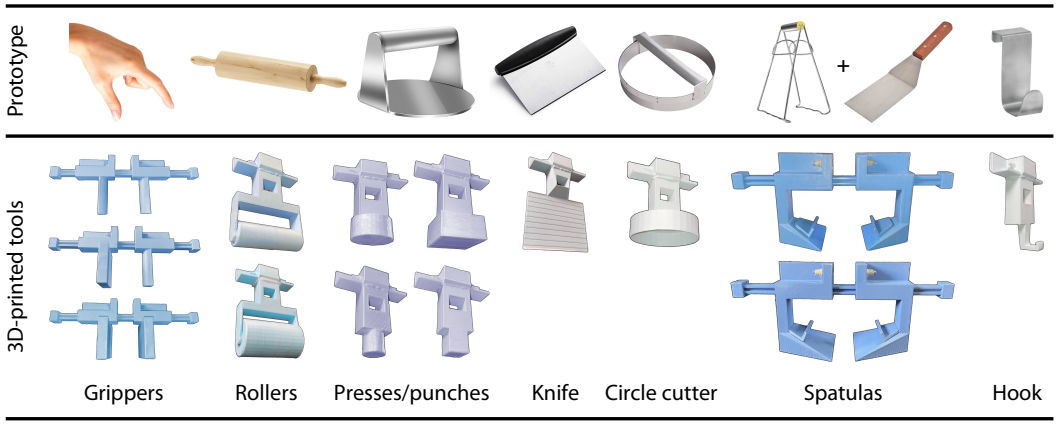

Figure 7: **Prototypes of 3D-printed tools**. We show a comparison between our 3D-printed tools and their real-world prototypes which are common kitchen tools for dough manipulation. The design principle of these 3D-printed tools is to mimic real-world ones as closely as possible. We use 3D-printed tools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

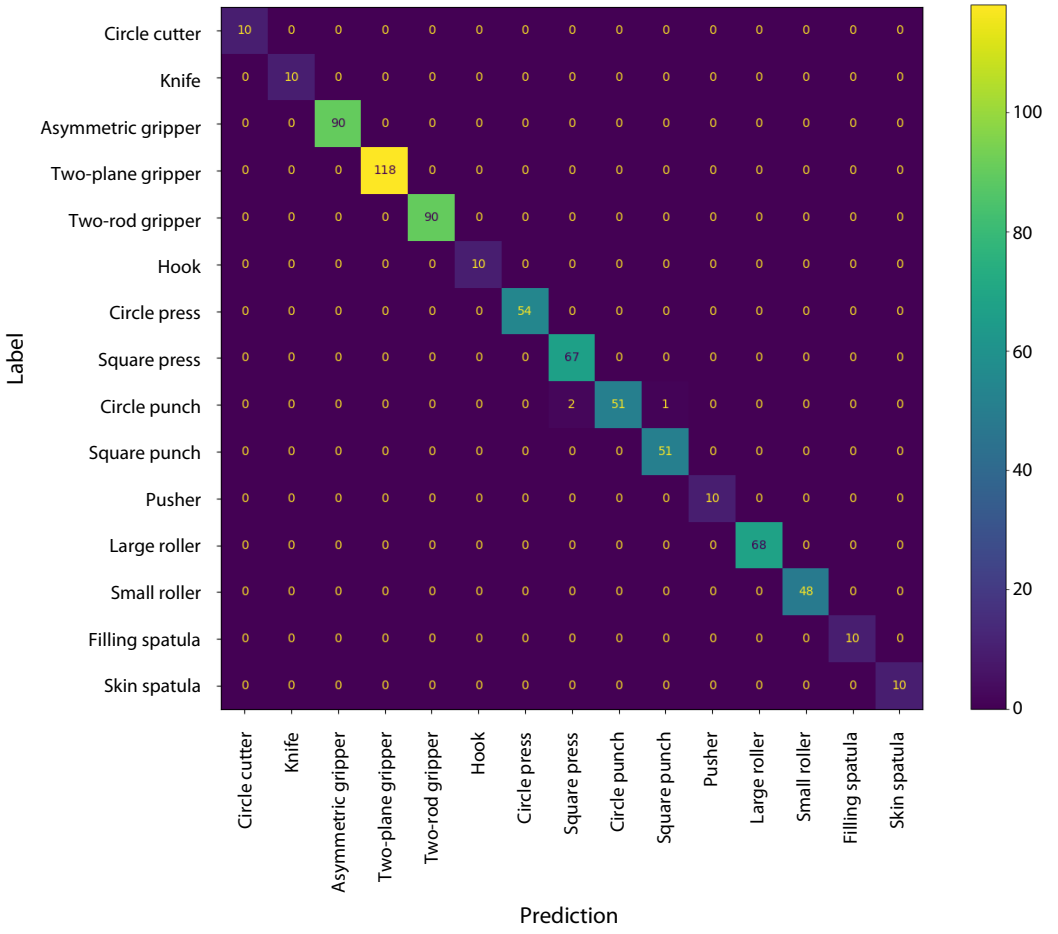

Figure 8: **Confusion matrix of the tool classifier predictions**. We show the confusion matrix of the tool classifier predictions on the test set, which is split from the training data. The tool classifier achieves an accuracy very close to 1.

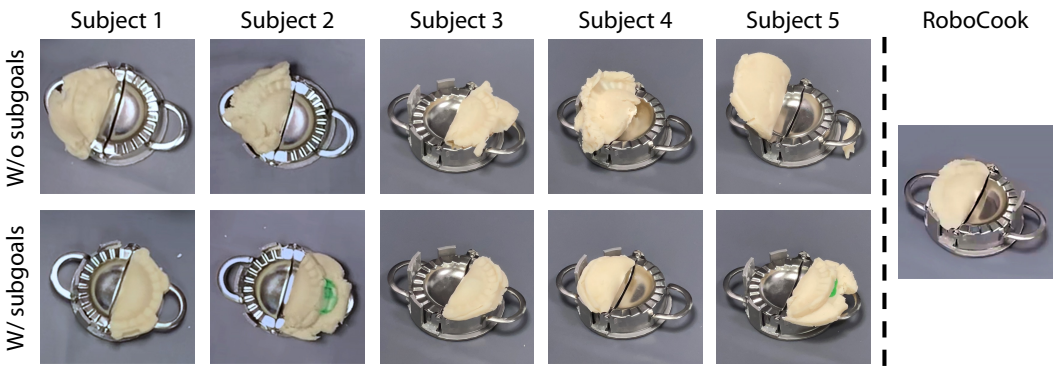

Figure 9: **Comparison with human subjects**. We show a comparison with the manipulation results of human subjects. In the first row, Human subjects devise their manipulation plan and choose tools independently. In the second row, human subjects follow a given tool sequence and subgoals.

6.5 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.

6.7 Tool Classification

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 Reinforcement Learning Baseline

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. The results shown in Figure 5 and Table 1 indicate that the RL baseline is noticeably worse compared to our method.