Learning Sequential Acquisition Policies for Robot-Assisted Feeding

Please refer to our website for videos, code, and supplementary material, as well as the 'Additional Experiments' page for supplementary ablations and a comparison to additional baselines. In this section, we provide an overview of the main design choices behind **VAPORS** and a thorough overview at implementation and experimental details.

A Simulator Details

A.1 Simulator Design

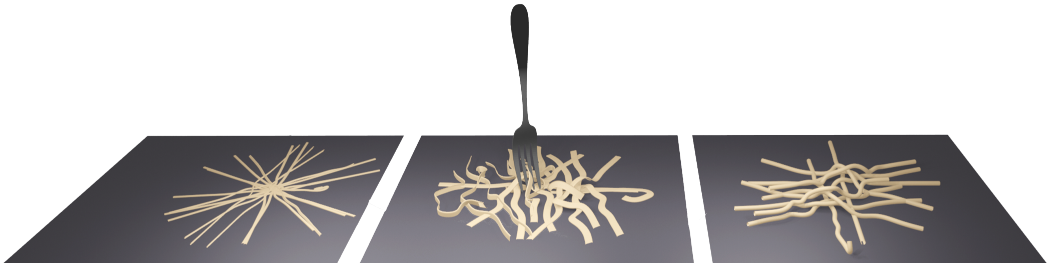

Figure 8: **Blender Food Simulation Environment:** We implement a custom food manipulation simulator in Blender 2.92 with an Open AI gym-style environment. The simulator supports softbody objects, such as noodles in different shape variations, as well as rigid, granular piles of items. We implement cutlery with arbitrary utensil meshes such as forks and spoons, and implement actions using the *keyframing* feature of Blender to control the position and orientation of a tool across frames.

We use Blender 2.92, a physics and rendering engine, to develop a custom feeding environment supporting deformable items, rigid items, and cutlery interactions. To instantiate deformable items like noodles (Fig. 8), we represent each item as a group of particles simulated with soft body physics. We treat granular piles of food such as jelly beans as separate rigid bodies. Additionally, we provide support for mesh-based utensils including a fork, spoon, and pusher tool, where we programatically keyframe the position and orientation of the tool across simulation frames to implement actions.

A.2 Reward Design

In this section, we describe the implementation of the reward function given in Eq. (3). For a set of known food item states in simulation $s_t = \{(x_i, y_i, z_i)\}_{i \in (1,\ldots,N)}$, `PICKUP` measures the quantity of food items picked up out of N total items. We detect a picked up food item in simulation by thresholding the z position of all items before and after an action, relative to plate height. Analogous to task progress metrics in cloth smoothing work [51, 52], we use `COVERAGE` to measure of spread of items on the plate. We compute this via the area of the convex hull of $\{(x_i, y_i)\}_{i \in (1,\ldots,N)}$, depicted in Fig. 2, via the Scipy Python library.

B Details of Learning-Based Methods

B.1 Latent Dynamics Training Details

We implement the latent plate dynamics model using the recurrent state space model from [37], with 64×64 input images and 30-dimensional diagonal Gaussian latent variables. This is a multi-headed deep recurrent network comprised of a learned encoder, transition model, and reward model. We supervise each head of the network with the following objectives:

- For the *encoder* $q(z_t | M_{\leq t}, a_{\leq t-1})$, we use an auxiliary decoder head that upsamples latent variables z_t to predicted images $\hat{M}_t$ and take the mean-squared error between $(\hat{M}_t, M_t)$ as a standard reconstruction objective. This encourages the learned latent representations to preserve the notion of food spread captured in segmented image observations.
- We supervise the *transition function* $p(z_\tau | z_{\tau-1}, h^k_{\tau-1})$ head using the KL-divergence for multi-step predictions as defined in [37].

- Finally, for the *reward model* given by $p(r_t|z_t)$, we take the mean-squared error between predicted rewards and ground truth rewards $(\hat{r}_t, r_t)$. This objective promotes accurately decoding rewards of future states to inform planning at test-time.

B.2 Planning with Learned Dynamics Model

Once trained, we use an MPC-style loop to sample and plan actions that maximize predicted rewards under the learned reward model.

At time τ, we can enumerate all K^T future candidate action sequences for the small library of primitives K, where T is the planning horizon. Conditioned on a history of observations $M_{1:t}$ and actions $a_{1:t-1}$, we imagine the future latent states $z_{\tau:\tau+T+1}$ under each action sequence $h^k_{\tau:\tau+T}$:

$$z_{t:t+T+1} \sim q(z_\tau | M_{1:t}, a_{1:t-1}) \prod_{i=\tau+1}^{\tau+T+1} p(z_i | z_{i-1}, h^k_{i-1}), \tag{4}$$

where $q(z_t|M_{\leq t}, a_{<t-1})$ is the learned encoder and $p(z_\tau|z_{\tau-1}, h^k_{\tau-1})$ is the learned transition model. Next, we predict decoded rewards according to the reward model $p(r_t|z_t)$ for each candidate sequence:

$$R = \sum_{i=\tau+1}^{i+T+1} \mathbb{E}\left[p(r_i|z_i)\right]. \tag{5}$$

Next, we select the sequence of actions $(\hat{h}^k_\tau, \hat{h}^k_{\tau+1}, \ldots, \hat{h}^k_T)$ which maximizes predicted cumulative reward R. The final step of the MPC planning loop is we take $\pi_H(M_{\leq t}, a_{\leq t-1}) = \hat{h}^k_\tau$, which is simply the first primitive in the predicted sequence. After executing this action, we replan with π_H, thus obtaining a second action and so on until $\tau = T$ (Algorithm 1).

Algorithm 1 Planning with VAPORS

```
for $\tau \in \{1, \ldots, T\}$ do
    $I_t, D_t \leftarrow$ Get current RGBD image observation
    $\hat{M}_t = f_{\text{seg}}(I_t)$ // Infer segmentation mask
    $\hat{h}^k_\tau = \pi_H(\hat{M}_{1:t}, a_{1:t-1})$ // Select high-level action
    Execute $\pi_L(\hat{M}_t, \hat{h}^k_\tau)$
```

B.3 Food Segmentation Training Details

Self-Supervised Dataset Generation. To circumvent the painstaking process of pixel-level segmentation annotation for real food images, we design a self-supervised annotation procedure. First, we record a grayscale RGB image of an empty plate, $I_{\text{empty}} \in \mathbb{R}_+^{W \times H}$. Next, we manually place food items on the plate at random without changing the position of the plate, yielding a new grayscale observation I_t. Let $I_{\text{diff}} = |I_{\text{t}} - I_{\text{empty}}|$, the framewise absolute difference between the full and empty plate. We initialize the ground truth segmentation mask M_t corresponding to I_t as a 2D array of zeros, and then assign $M_t[I_{\text{diff}} > \texttt{THRESH}] = 1$. In practice, we find that `THRESH` = 20 reasonably separates the foreground from the background to detect food. With this procedure, we can scalably collect 280 paired RGB food images and segmentation masks in real within an hour and a half of data collection. This includes plate resets, food placement, image capture, and offline background subtraction post-processing.

Augmentation. We augment this dataset 8X by randomizing the linear contrast, gamma contrast, Gaussian blur amount, saturation, additive Gaussian noise, translation, and rotation of each RGB image, applying only the affine component of these same transformations to the associated segmentation masks.

Training Objective. We train f_{seg}, implemented as a fully convolutional FPN (Feature Pyramid Network) using Dice loss:

$$\mathcal{L}_{\text{dice}} = 1 - \frac{2 \times \texttt{TP}}{2 \times \texttt{TP} + \texttt{FN} + \texttt{FP}} \tag{6}$$

This objective encourages high overlap between predicted and ground truth masks, as `TP, FN, FP` denote the number of pixel-level true positives, false negatives, and false positives in a prediction $\hat{M}_t$ compared to ground truth M_t.

C Experimental Details

C.1 Noodle Acquisition Hardware Setup

Using a Franka Panda 7DoF robot, we aim to clear a plate of cooked noodles within a horizon of $T = 10$ actions. We fit the end-effector with a custom 3D-printed mount consisting of a RealSense D435 camera and a fork. To enable autonomous twirling and scooping capabilities, we extend the fork's range of motion via two servo motors (Dynamixel XC330-M288-T). We control the robot with a Cartesian impedance controller, where the programmable servos are integrated in the forward kinematics chain for positional control of the fork tip. The action space consists of either *group* (rearrangement) or *twirl* (acquisition) actions, instantiated according to the learned segmentation and pose estimation models detailed in Section 4.2.

A group action consolidates a sparsely distributed plate by sensing the furthest and densest points, $(\hat{x}_f, \hat{y}_f, \hat{z}_f)$ and $(\hat{x}_d, \hat{y}_d, \hat{z}_d)$, and executing a planar push from the furthest to densest point. In a twirl action, we infer the densest point and appropriate insertion angle $\hat{\gamma}$, roughly orthogonal to the grain of majority of the noodles. We use positional control to insert the fork into the densest noodle pile, and execute a fixed twirling motion by making two rotations at 6 radians per second. Finally, the fork scoops upward until nearly horizontal ($\beta = 80°$) and the robot brings the acquired noodles to a neutral position in the workspace.

For all trials, we use a non-slip plastic dinner plate, and mimick a bite successfully taken by a user after twirling by autonomously untwirling onto a discard plate.

C.2 Bimanual Scooping Hardware Setup

We assume access to two Franka Panda robots, equipped with a pusher tool and a metal spoon, respectively, and an external RealSense D435 camera for perception. With this setup, we aim to acquire granular items on a plate using either *group* (rearrangement) or *scoop* (acquisition) actions, with a total action budget of $T = 8$ actions. In particular, we evaluate our system on the task of scooping jelly beans, but **VAPORS** is agnostic to the exact choice of food. Following the experimental setup of Grannen et al. [7], the spoon is mounted at an angle to the robot end-effector ($\beta = 30°$). The pusher is a concave 3D-printed tool intended to push piles of items into the spoon and maintain contact during lifting so as to prevent spillage.

Grouping actions are unimanual and use the pusher tool to push the sensed furthest item to the densest region on the tray. In a scoop action, we sense the densest pile and execute a parameterized motion in which the pusher and spoon move towards each other synchronously at a fixed $\gamma = 180°$. Once they arms are within a fixed threshold apart, the spoon scoops by tilting to $\beta = 80°$ and lifting to a neutral workspace position.

We conduct all trials on a standard cooking tray due to the enlarged manipulation workspace for two arms. To simulate a user's bite between actions, we manually discard the spoon contents after a scoop action.

C.3 Implementation Details

For each task, we use the following training procedures. We train π_H on simulated segmentation observations of size 64×64 for $2,250$ update steps, where we collect 1 episode every 150 update steps. We instantiate the reward as per Eq. (3) with $\alpha = 0.66$, and train each model using the Adam optimizer with with a learning rate of 10^{-3} , $\epsilon = 10^{-4}$, and gradient clipping norm of 1000 with batch size $B = 32$, based on the training procedure from [37]. Each model takes approximately 1 hour to train on an Nvidia RTX A4000 GPU. To instantiate π_L, we train f_{seg} and f_{ori} from real data. For segmentation, we collect 280 paired examples of images and binary segmentation masks using the self-supervised annotation process from Section 4.2, where we use cooked noodles of randomized

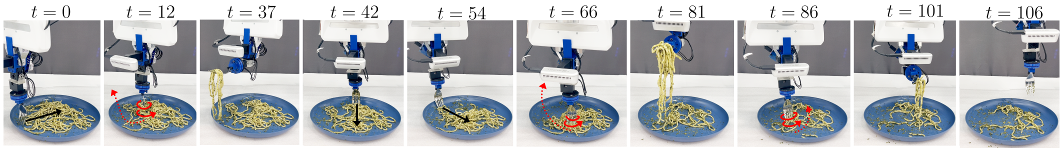

Figure 9: **Noodle Acquisition Rollout:** We visualize 6 actions performed by **VAPORS** on the task of clearing an initially half-full plate of Tier 3 noodles. As the distribution of noodles on the plate becomes sparse ($t = 0, 42, 54$), **VAPORS** employs grouping strategies (black) to push noodles in close proximity. Once consolidated, **VAPORS** employs twirling ($t = 12, 66, 86$), as shown in red, for efficient plate clearance, where t denotes the clock time in seconds.

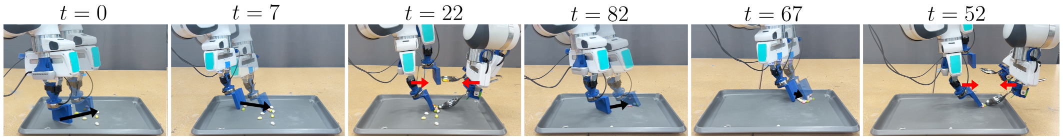

Figure 10: **Bimanual Scooping Rollout:** Using a bimanual setup with two Franka Emika Panda robots, **VAPORS** performs 6 actions consisting of grouping (black arrows) and scooping (red arrows) to acquire jelly beans on a tray. By grouping when the tray is sparse and acquiring when a bite-sized clump forms, **VAPORS** demonstrates efficient acquisition. The annotated timestamps denote clock time in seconds.

shape and sauce variations as well as jelly beans of randomized colors. We augment each dataset 10X and train for 50 epochs, which takes approximately 3 hours on an NVIDIA GeForce RTX 2080 GPU. In order to instantiate the twirl primitive for noodle acquisition, we additionally train f_{ori} to predict fork tine orientation γ from 280 manually annotated crops of noodles as per Section 4.2, augmented 8X. The train time for f_{ori} is approximately 1 hour on an NVIDIA GeForce RTX 2080 GPU. For deployment, we use an Intel NUC 7 for inference and robot control via a ROS 2-based control stack.

C.4 Additional Experimental Results

In this section, we supplement the experimental findings from Section 5 with additional results.

Plate Clearance: Fig. 9 and Fig. 10 visualize two rollouts of **VAPORS** on plate clearance. We note that visually, **VAPORS** tends to favor grouping as the plates become sparser and otherwise acquires when there is a reasonably sized bite available.

VAPORS Failure Mode Categorization: In addition to evaluating the percentage of the plate cleared, we observe the occurrence of a few failure modes, as depicted in Fig. 11. A *misplanned* action (A) can occur due to a perception error, such as accidentally perceiving sauce, a garnish, a vegetable, or plate specularity for noodles and erroneously grouping or twirling in that region.

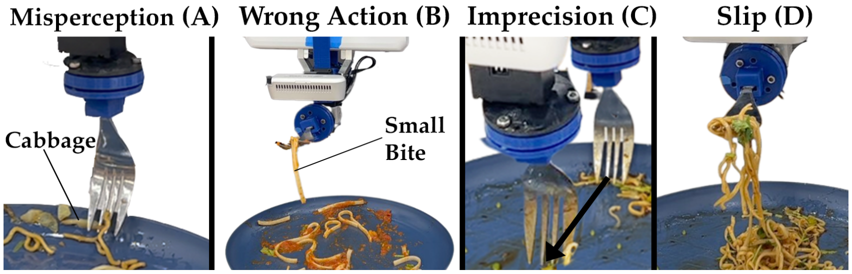

Figure 11: **VAPORS Failure Modes:** We illustrate the 4 most commonly observed failure modes with **VAPORS** on noodle acquisition. Misperception (A) occurs when π_L erroneously senses vegetables, sauce, or plate glare as a noodle due to false positives with f_{ori}, leading to a misplanned action such as grouping in that region. Occasionally, π_H may acquire when rearrangement is more appropriate, leading to a low-volume bite (B). In terms of action execution, food acquisition requires care so as to not miss food (C), as seen in the grouping motion which fails to group singular noodle strands due to system imprecision. Finally, slippage (D) can happen during acquisition with highly adversarial items such as those coated in sauce.

Alternatively, this can happen when (B) the robot twirls when grouping is more appropriate or vice versa. A *mis-executed* action failure occurs when (C) the fork fails to group or acquire due to system imprecision or (D) the noodles slip during acquisition due to sauce. In Table 1, we also report the per-action failure rate, computed as the total number of failures over the total number of actions (60 = 6 trials × $T = 10$).

Qualitative User Study: In the Likert survey administered to gauge user preferences across methods, we report in Fig. 6 the statistical findings which are significant. In Table 2, we indicate the specific margin of significance for each of the criteria, obtained via 1-way ANOVA testing.

Table 2: **1-Way ANOVA Statistically-Significant Findings** (p-value < 0.05)

Criterion	**Method 1**	**Method 2**	p-value
Efficiency	Heuristic	VAPORS	0.0004
Efficiency	Acquire Only	VAPORS	0.0010
Bite Size	Acquire Only	VAPORS	0.0318
Humanlike	Heuristic	VAPORS	0.0003
Humanlike	Acquire Only	VAPORS	0.0044
Practicality	Heuristic	VAPORS	0.0012
Practicality	Acquire Only	VAPORS	0.0025
Reuse	Heuristic	VAPORS	0.0000
Reuse	Acquire Only	VAPORS	0.0008
Trust	Heuristic	VAPORS	0.0124
Trust	Acquire Only	VAPORS	0.0198
Generalizability	Heuristic	VAPORS	0.0478

Table 3: **Noodle Acquisition**

Criterion	**Method 1**	**Method 2**	p-value
Efficiency	Heuristic	Acquire Only	0.0029
Practicality	Heuristic	Acquire Only	0.0091
Reuse	Heuristic	Acquire Only	0.0093
Efficiency	Heuristic	Acquire Only	0.0029
Efficiency	Acquire Only	VAPORS	0.0000
Bite Size	Heuristic	VAPORS	0.0029
Bite Size	Acquire Only	VAPORS	0.0002
Humanlike	Acquire Only	VAPORS	0.0094
Practicality	Heuristic	Acquire Only	0.0091
Practicality	Acquire Only	VAPORS	0.0001
Reuse	Heuristic	Acquire Only	0.0093
Reuse	Acquire Only	VAPORS	0.0001
Trust	Acquire Only	VAPORS	0.0438
Generalizability	Acquire Only	VAPORS	0.0481

Table 4: **Bimanual Scooping**

In addition to the user study outlined in Section 5, we administered a second part of the study, in randomized order to the first, in which users were asked to pick a preferred method for feeding in side-by-side comparisons of jelly bean acquisition trials. To control for the initial state of the jelly beans, we purposely arrange 16 beans into a 4×4 grid initially, and conduct two trials per method which are randomly selected for the comparisons. Although we would like to include an analogous side-by-side comparisons survey for noodle acquisition for completeness, we find in practice that controlling for the initial state of noodles is nontrivial due to their highly deformable nature and vast set of feasible initial configurations. This makes it difficult to present users with unbiased comparisons across methods.

Thus, for bimanual scooping, we presented all permutations of pairs of the three methods, for a total of six comparisons overall. Empirically, we find that **VAPORS** is the preferred method by a large margin compared to both baselines (Fig. 12).

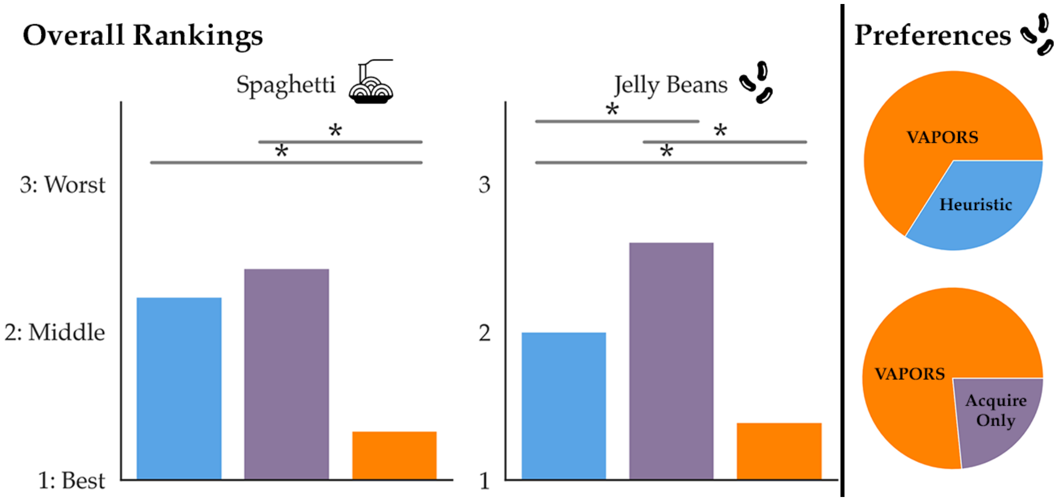

Figure 12: **Overall Ratings** Left: After observing all methods perform acquisition across 10 trials, we ask users to rank all three methods from most to least preferable. We find the **VAPORS** is most consistently ranked the best by a statistically significant margin ($p < 0.05$, denoted '*') compared to the baselines. Right: For jelly bean acquisition, we control for the initial state of the plate by arranging the beans in a 4×4 grid, and ask users to select their preferred method across 6 side by side acquisition videos of different methods. **VAPORS** is the preferred method by a large margin compared to Heuristic and Acquire-Only.