Learning to Perceive and Manipulate Diverse Food Materials Through Interaction

Amrita Singh Sawhney

CMU-RI-TR-21-06
May 2021

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of M.S. Robotics.

Abstract

The home kitchen environment presents many challenges for an autonomous cooking robot, such as the deformability of food items, the wide range of material properties of food, and the complex interaction dynamics involved in food manipulation tasks. Material properties are important when interacting with non-rigid body objects, because they have a large influence on how the skills should be performed. To successfully execute skills such as food cutting, the robot should be able to infer material properties of and adapt its behaviors to different food materials. In this thesis we discuss experimental approaches to robot learning in the context of food manipulation skills. First, we present an approach to encode food item material property information through unsupervised robotic interactions with a variety of food classes, and show how these learned embeddings can be used for material property-related classification and regression tasks. Second, we focus on enabling the robot to learn a range of food cutting behaviors from human demonstrations and optimize these behaviors using reinforcement learning, specifically looking at different ways of defining rewards for robotic cutting tasks. Our results indicate that the robot can autonomously learn about food item material properties through interaction, and also that it can successfully learn and optimize its cutting behaviors for different food types using a reinforcement learning framework.

Acknowledgments

I'd first like to thank my advisor, Professor Oliver Kroemer, for being a wonderful mentor who has supported, challenged, and helped me grow in so many ways since I first dove into the exciting world of robotics two years ago. Next, I would like to thank my lab colleagues for fostering such a welcoming, collaborative, intellectually stimulating, and inclusive research environment. Particularly, thanks to Mohit Sharma, Kevin Zhang, Steven Lee, Jacky Liang, Tim Lee, and Tetsuya Narita for always making time to teach me new concepts, help me formulate problems, and provide insightful input and research guidance. I'd also like to thank Professor David Held for his thoughtful feedback during my research over the past year. Finally, I'd like to thank my family and friends, who have supported and encouraged me throughout graduate school in more ways than I can express. Special thanks to my incredible dad, who has not only been my biggest supporter, intellectual mentor, and role model, but has also provided me with a constant supply of Seattle coffee beans to fuel my experiments and thesis writing over the past year. I would not be where I am today without any of these people and I am extremely grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A future with robots working autonomously in home kitchens to prepare and cook food for people is an exciting vision. Working towards realizing this vision of an at-home chef robot requires a synergy between robotics research in manipulation, controls, perception, and human-robot interaction. The kitchen is a very challenging environment for autonomous robots, largely because of its unstructured and dynamically changing nature. Because of this complex environment, the cooking domain presents an interesting area for research into how robots can be deployed in the real world in a way that is helpful to people in their daily lives. Imagine, for example, an elderly person who is unable to cook meals because of the physical demands of cooking. An at-home cooking robot could provide them with a level of self-sufficiency, enabling them to enjoy home-cooked healthy meals without the need for physical exertion or leaving their home.

Certain companies, such as Moley Robotics[1],Miso Robotics[2], and Dexai Robotics[3], have realized the potential impact of having robots in the kitchen and have developed various products aimed at this domain. However, there are significant challenges yet to be overcome, since many of the existing products on the market are targeted at very specific applications or are meant for

[1]`https://moley.com/`
[2]`https://misorobotics.com/`
[3]`https://www.dexai.com/`

more industrial-type applications rather than in the home kitchen setting.

From a robotic manipulation standpoint, working with food items presents challenges such as the large variability in food material properties including stiffness, hardness, elasticity, plasticity, ductility, liquid content, shear strength, and surface friction. Additionally, there is a great deal of variability both across different food items and also within each food type. For example, fruits and vegetables that have varying degrees of ripeness, are cooked to varying degrees, or prepared with different cooking methods exhibit different material properties. Because of the range of material properties in food items and the complex dynamics involved, food cutting is a challenging action to simulate. Therefore, the majority of the work presented in this thesis is done on a real robot, a 7-DOF Franka Emika Panda serial manipulator, rather than in simulation. This also provides a great opportunity to better understand the practical challenges associated with deploying cooking robots in the real world.

The research presented in this thesis strives to take a step towards realizing the vision of a home cooking robot that is able to acquire new skills and adapt to diverse materials. Characterizing and adapting to different materials is a critical skill needed for robots to successfully perform many food manipulation tasks. In this thesis we explore a few different areas related to robot learning in the food manipulation domain. The specific focus of this research is two-fold: first, we want the robot to autonomously reason about different food items through playful exploration and learn how to represent them. Second, we want to robot to learn and adapt food cutting skills to different food materials through interaction.

In Chapter 3 we present an approach to learning food item representations through autonomous robotic exploration that can be useful for estimating object material property information. Our key contributions here are the collection of a varied multimodal food interaction dataset, and an approach to learning embedding representations from this dataset. In Chapter 4, we focus on experimental approaches to applying learning methods to robotic food cutting. We look at teaching the robot a range of different food cutting behaviors from human demonstrations, parameterizing these skills using Dynamic Movement Primitives, and fine-tuning the skill parameters using reinforcement learning (RL) methods. We then explore adding human input to the RL framework and evaluate the robot's ability to tune its cutting behaviors to the human's preferences. Our results show promising potential for teaching robots real-world, complex food manipulation skills and enabling them to adapt these skills to diverse food materials. Finally, we discuss conclusions from this research in Chapter 5 and identify some key open challenges in the domain of robotic food manipulation in Chapter 6.

Chapter 2

Related Works

The work presented in this thesis spans across a range of topics within robotics research, including imitation learning from human demonstrations, learning-based approaches to robotic food cutting, human-in-the-loop reinforcement learning, deformable object manipulation, and interactive perception. Many prior works have explored these topics for a variety of different applications. This chapter seeks to contextualize the work presented in this thesis and how it fits into existing research in the robotics community.

Teaching robots skills through human kinesthetic demonstrations has been used by many researchers as an effective way of teaching complex behaviors that are difficult to hard-code or teach by other methods such as teleoperation [1, 2, 6, 14]. Like these previous works, our approach starts off with human kinesthetic demonstrations of various cutting behaviors by guiding the robot arm through the movements in low stiffness mode. However, unlike prior works focused on learning skills solely from demonstrations, we use the human demonstrations for policy initialization and then allow the robot improve the human-taught policies with a reinforcement learning framework. Therefore, our work explores the combined problems of imitation learning and reinforcement learning in the context of robotic food cutting skills.

Once kinesthetic trajectories are collected from human demonstrations, we parameterize the different cutting behaviors using the raw trajectory data. Dynamic Movement Primitives (DMPs) provide a flexible framework for generating smooth trajectories, allowing us to parameterize skills by learning the DMP parameters from human demonstrations of these skills. More details regarding the DMP formulation we use is in Section 4.1. Many previous works have used DMPs for generating trajectories for a wide range robotics skills, such as placing, tossing, soccer playing, table tennis playing, and also robot cutting [3, 27, 38]. Zhang et al [38] use the DMP formulation proposed by Kroemer et al [20] for food cutting DMPs, which is the same formulation we use in this thesis. While [38] focuses on a planar back and forth cutting motion and hand-selects skill parameters for different foods, our work expands on this to incorporate a wider range of cutting motions and seeks to have the robot learn optimal skill parameters through interaction. In our approach, the robot fine-tunes cutting DMP policies for a range of food material types by exploring around the human demonstrated policies and iteratively updating them based on a defined reward. To our knowledge, this has not been explored in-depth in previous work in the food manipulation domain.

Prior works have also taken different approaches to learning robust robotic food cutting policies. Papers such as [21, 26] take a Model Predictive Control (MPC) approach to the problem. These works learn a recurrent neural network dynamics model for the contact dynamics of the task from data, and then use it with a model predictive controller to execute planar back and forth movement cuts [21, 26]. In contrast, our work uses a model-free RL framework to optimize a range of cutting policies for diverse food types through interaction, while leveraging human demonstrations to allow for sample-efficient learning. We combine learning from demonstration and reinforcement learning to reduce sample complexity associated with many model-free RL approaches that seek to learn policies from scratch. We also explore optimization of different cutting skills with the same generalized reward function, while these prior works largely focus on a single back and forth cutting motion.

The next area of our work explores the area of human-in-the-loop (HIL) reinforcement learning applied to robotic food cutting to allow the human to provide input to guide policy learning. Many previous works have looked at human-in-the-loop reward learning as a method of allowing a human expert to specify rewards to the agent in place of hard-coded reward functions. This is especially useful for complex real-world problems, where reward functions are often difficult to specify. [17] explicitly optimizes a loss function that incorporates the human critic's ratings, an approach evaluated in simulation game engines. In contrast, in our work we explicitly model the reward using noisy human inputs while minimizing human queries, and we evaluate this in real-robot experiments rather than simulation. Another human-in-the-loop approach presented in [12] is to use a Bayes optimal control algorithm to directly convert human feedback into a policy update, which they evaluate in simulation using a simulated human teacher. Our approach differs because we model the human rewards using a Gaussian process and use the learned reward model to optimize the policy. Other papers, such as [7, 19] develop a framework for combining expert advice to model the human's reward function, where the agent chooses actions that try to maximize reward. These works evaluate this approach on sequential decision making tasks in simulation on games. While we also use the idea of learning a reward model from human inputs in our work, we focus on applying human-in-the-loop reward learning methods to real world robotic tasks and perform online experimental evaluations using a Franka robot. Finally, Daniel et al propose an approach to learn a reward model from human rewards while minimizing queries to the human, and simultaneously optimizing a policy [9]. Our work closely follows the methods proposed by Daniel et al, but we seek to apply their methods to a different task of robotic food cutting, which to our knowledge is a new application of this technique and presents new challenges due to the nature of the task. We also evaluate batch vs. iterative updates to the reward model during the learning process.

The other main research thread discussed in this thesis is around learning deformable food representations through autonomous interaction and play. This is an area closely related to robotic food cutting, since reasoning about food material properties is critical to enabling the robot to successfully manipulate different foods. While many works focus on learning deformable object material properties and dynamics models in simulation [24, 35, 37], our approach involves real robot interactive data collection with food items to avoid the difficulties of simulating food items and robot interactions with them. Other studies, such as [5, 16, 22, 38] use multimodal sensory data to glean object material information and inform various deformable object robotic manipulation tasks. Our approach, while also aiming to characterize object material property information, only requires overhead images as input at test time since we use the multimodal sensor data as a contrastive metric during training of our embedding networks.

In the interactive perception domain, works such as [4, 18, 32, 33, 34] have shown the use of vision, haptic feedback, audio, and proprioception to identify the location of objects or classify them. Our work focuses on learning representations encoding material properties for a wide range deformable food items, rather than rigid body objects, through data collected while the robot playfully and repeatedly interacts with the foods. Another key difference in our approach compared to interactive perception works is that the multimodal sensor data is only used during embedding training. At test time, only an image of the slice is needed as input to the learned embeddings networks. To our knowledge this has not been explored thoroughly in prior works. Additionally, many researchers have looked into using deep neural networks to learn lower dimensional object representations that can be used in manipulation tasks. Related works in this area, such as [11] have looked at designing features and using supervised learning to map these features to physical food material properties. Our work is related in that we also seek to learn features that encode object material properties, however, we take a self-supervised approach. In our method, the robot plays with the food and we then use the data collected to train embedding networks using self-supervision that output embeddings encoding food material properties. Similarly, [15, 31] train embeddings networks with visual data to represent food slice thicknesses and fruit ripeness over time. Our approach, on the other hand, seeks to learn food object embeddings through a multimodal sensory data approach incorporating vision, proprioception, and audio during embedding training.

Chapter 3

Learning Food Representations

Material in this chapter comes from the paper "Playing with Food: Learning Food Item Representations through Interactive Exploration" with co-authors Amrita Sawhney, Steven Lee, and Kevin Zhang [29].

Humans implicitly use their knowledge of food material properties when they are interacting with food in the kitchen, but it is difficult to explicitly specify features. For example, how would one describe the material properties of a kiwi? Mozzarella? Spam? Uncooked vs. cooked steak? Yet, when a person cuts any of these items, they implicitly know to handle them differently due to their prior experience handling different types of foods.

In this chapter, we explore how a robot can autonomously reason about food items through interaction, taking inspiration from how young children playfully interact with food. In Section 3.1.1 we first discuss collecting a multimodal food interaction dataset consisting of vision, audio, proprioceptive, and force data acquired through autonomous robot interactions with a variety of food items. We then, in Section 3.1.2, present a way to use this multimodal dataset to learn continuous embeddings that encode material property information such as hardness, juiciness, and whether or not the food is cooked. We think the embeddings learned with our approach can be useful in planning for more material-aware skills in the food manipulation domain.

To learn these embeddings, we use a network architecture that takes in visual input (i.e. an overhead Kinect image of a food slice), and outputs continuous embedding vector representations. The networks are trained using the self-supervised method of triplet loss, where similar and dissimilar samples are grouped based on features extracted from the multimodal sensor data. Therefore, we can learn complex representations of the different food items autonomously without needing to have a human label all the samples. The goal of using the audio, visual, and proprioceptive data as similarity metrics to train the networks is to encode material property information in the learned embeddings. Finally, in Section 3.2.2 we evaluate the utility of the learned embedding representations in various classification and regression tasks related to identifying food material properties and compare their performance to baselines that use only visual and only audio data. Our results show that regressors and classifiers trained using our learned embeddings outperform similar baseline networks on certain tasks, indicating that the embedding network encodes additional material property information using the sensor data. This approach is also advantageous because the robot does not need to collect additional interactive sensor data with the object at test time, since the learned networks only require visual input.

A link to our publicly available dataset is located here[1].

3.1 Methods - Learning Food Representations

3.1.1 Collecting Real-World Multimodal Food Dataset and Experimental Setup

In this section of the thesis we discuss our methods for collecting a multimodal dataset through autonomous robot "playful" interactions with a wide range of food items.

[1]`https://tinyurl.com/playing-with-food-dataset`

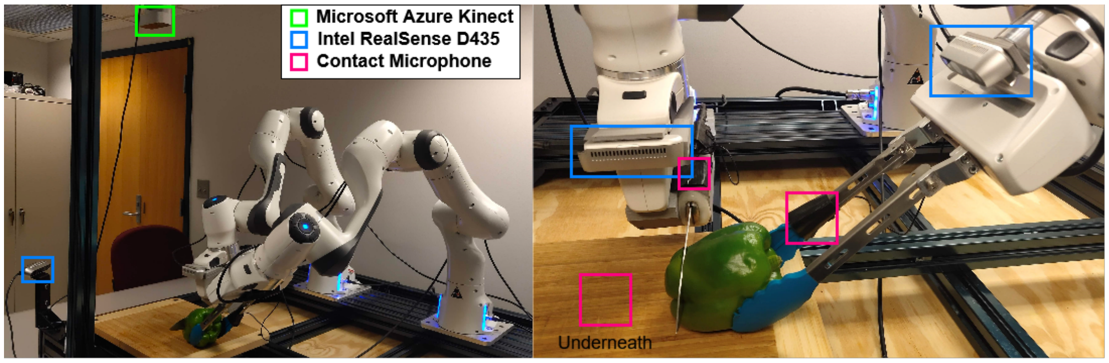


Figure 3.1: Our cutting experimental setup, consisting of two Franka arms with knife and tong attachments, cutting board, cameras (overhead, side-view, and wrist-mounted), and contact microphones.

Experimental Setup

We collect data using two different experimental setups - one for robotic cutting where the robot cuts the whole food items into 10 different types of slices, and a second for food playing, where the robot plays and interacts with the food slices that were cut. We collect multi-modal sensor data during both the cutting and playing phases using our Franka robot control framework [40].

Robot Cutting:

Our cutting experimental setup uses two Franka Emika Panda arms with a cutting board mounted in between, shown in Figure 3.1. One arm grasps a knife for cutting in its end effector, and the other arm has 8" kitchen tongs mounted to its end effector. We mount four cameras to this setup (an overhead Microsoft Azure Kinect Camera, a side-view Intel Realsense D435, another D435 mounted above the wrist of the robot holding the knife, and a third D435 mounted on the wrist of the robot holding the tongs. Sample images from each camera are shown in Figure 3.2). We also mount three contact microphones in different locations: one underneath the center of the cutting board, another on the knife attachment, and the last on the tongs.

Robot Playing:

Our playing setup has a single Franka arm and a cutting board in front of it (shown in Figure

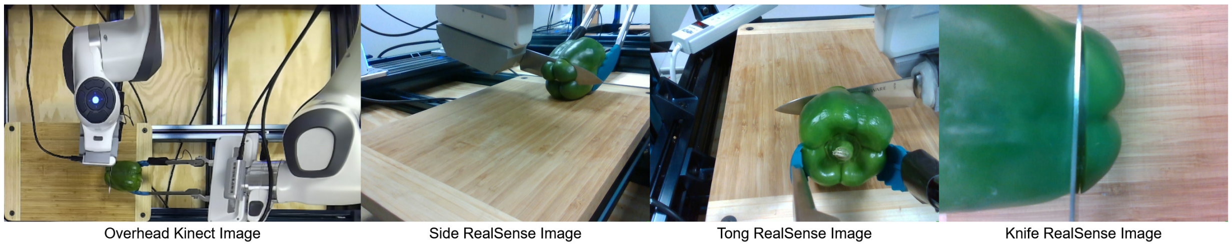


Figure 3.2: Images from the 4 cameras spread around our cutting experimental setup.

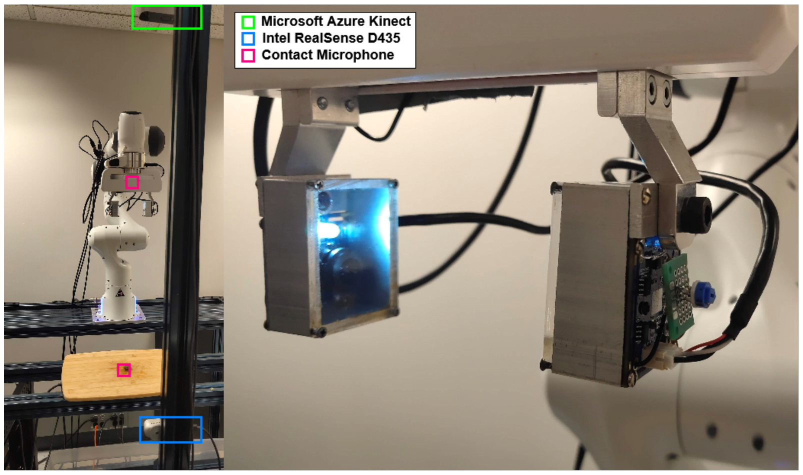


Figure 3.3: Our playing experimental setup consisting of a Franka arm, cutting board, cameras (overhead, side-view, and FingerVision), and contact microphones.

3.3). We mount an overhead Microsoft Azure Kinect Camera and a front Intel Realsense D435 that faces the robot. We also attach a fisheye 1080P USB Camera3 to each fingertip as in FingerVision [36] with a clear acrylic cover over the cameras, which allow us to get images of the object being grasped between the fingertips. Images from all of the cameras are shown in Figure 3.4. We also mount contact microphones in this setup - one underneath the center of the cutting board and the other mounted on the back of the Franka Panda hand. The piezoelectric contact microphones are used to capture vibro-tactile feedback through the cutting board, fingers, and tools, and the audio from the contact microphones are captured using a Behringer UMC404HD Audio Interface [2] and synchronized with ROS using sounddevice ros [39].

[2]https://www.amazon.com/BEHRINGER-Audio-Interface-4-Channel-UMC404HD/dp/B00QHURLHM

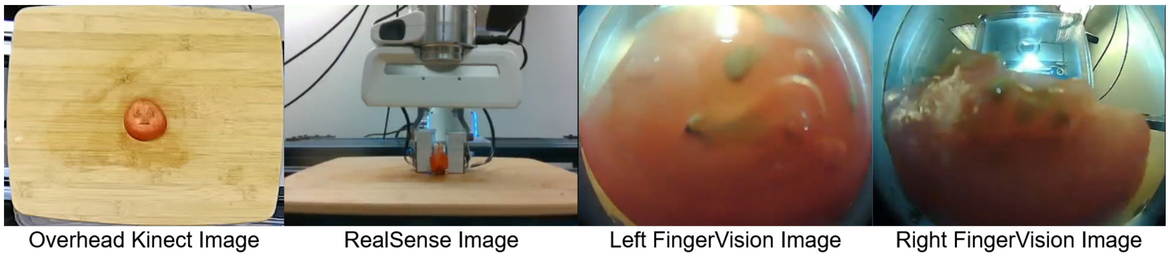


Figure 3.4: Examples of Kinect, Realsense, and FingerVision images of a cut tomato.

Data Collection

We first teach the robot cutting skills via kinesthetic demonstrations, by fitting parameterized Dynamic Movement Primitive (DMP) trajectories to the raw trajectory data collected during the demonstrations (more detail on this will be discussed in Section 4.1). The robot performs 10 different cuts to generate slices of each of 21 different food types: apples, bananas, bell peppers, bread, carrots, celery, cheddar, cooked steak, cucumbers, jalapenos, kiwis, lemons, mozzarella, onions, oranges, pears, potatoes, raw steak, spam, strawberries, and tomatoes. The slice types are labeled 1 to 14 and generated using varying skills parameters such as thickness (3mm, 5mm, 10mm, 20mm, and 40mm slice thicknesses), starting knife angles (+/- 30 degrees), and slice orientations. There are more than 10 slice types because food items are shaped differently and not all cuts can be executed on every food item, especially the angled cuts. While the robot is cutting the food items, we collect audio, image, force, and proprioceptive data. Figure 3.5 shows the slices resulting from the robot's cuts. Once the robot has cut the different slices types, we move them to the playing setup and start collecting playing data. Playing data collection involves five trials per slice type. The goal of running multiple trials on each slice is to capture variations in the object's response to the robot's actions due to different initial conditions at the start of each trial, such as orientation, position, and in some cases shape due to deformation of the slice over time. During each trial, we first take an RGBD image with the overhead Kinect to get the center of the object. Next, the robot closes its gripper and pushes downward on the object until 10N of force is reached. We refer to this as the "push down" action and the robot's forces and end effector positions are recorded during this action. Next, the robot resets its position, and performs the next action, which is grasping. It grasps the object and lifts it to a specified height of 15cm, and finally opens its gripper to release and drop the food item onto the cutting board (referred to below as the "release" action).

During all three actions, we record Realsense videos and audio from the two contact microphones. We also record FingerVision camera videos during the grasp and release actions, and record gripper width post-grasp. Finally, we save Overhead Kinect RGBD images before and after all three actions are performed in each trial. Our full dataset is available for download here[3]. The data is located in the appropriately named folders and are sorted first by food type, then slice type, and finally trial number.

Data Processing

Before training the embeddings networks, we first extract features from the raw sensor data collected during the food interactions. For the audio data, we transform the raw cutting and playing audio (collected during the push-down, grasp, and release actions) into Mel-frequency cepstrum coefficient (MFCC) features [23] using Librosa [25], which have been shown in [38] to differentiate between different material types. We then perform principal component analysis (PCA) on the raw MFCC features to get lower-dimensional representations of the cutting ($\mathcal{A}_{cut}$) and playing ($\mathcal{A}_{play}$) audio features, reducing the dimensionality of the raw features by about half.

For the proprioceptive features, we analyze the robot poses and forces collected during the push down and grasp actions, and extract three features: first, the final z position (z_f) of the robot's end-effector during the push-down action once 10N of downward force is reached, second the change in z position between the point of first contact and z_f (referred to as Δz and gives a metric related to the object's stiffness), and lastly the final gripper width (w_g) during the grasping action once 60N of force has been reached. These three features are combined into an array to

[3]`https://tinyurl.com/playing-with-food-dataset`

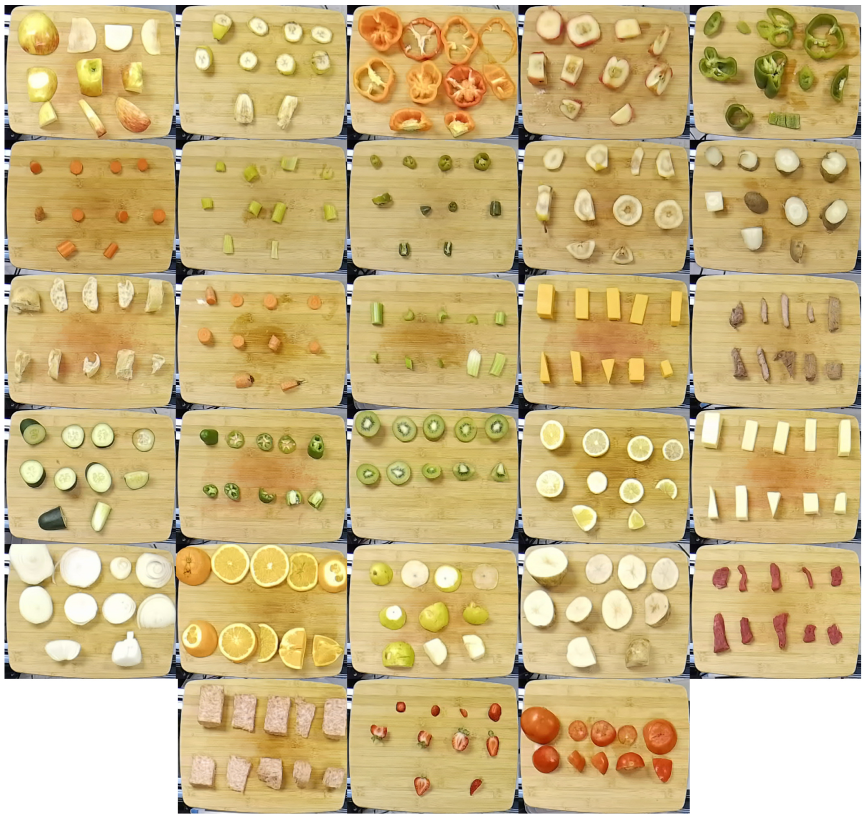


Figure 3.5: Food item slices. From left to right, top to bottom: apple, banana, bell pepper, boiled apple, boiled bell pepper, boiled carrot, boiled celery, boiled jalapeno, boiled pear, boiled potato, bread, carrot, celery, cheddar, cooked steak, cucumber, jalapeno, kiwi, lemon, mozzarella, onion, orange, pear, potato, raw steak, spam, strawberry, and tomato.

form $\mathcal{P}$. We also have high level labels for each data sample based on food class ($\mathcal{F}$) and slice type ($\mathcal{S}$) (based on the slice action performed during the cutting phase).

3.1.2 Learning Embeddings from Multimodal Data

Our approach to learning embeddings for the food items, shown in Figure 3.6, is to train convolutional neural networks that take an overhead RGB image of a food slice as input and output a lower dimensional continuous embedding using self-supervision. Our architecture consists of ResNet34 [13], which is pre-trained on ImageNet [10] and has the last fully connected layer removed. We add an additional three hidden layers, with ReLU activation for the first two, to reduce the dimensionality of the embeddings.

We use a triplet loss [30] to train the network, so similarities across food types are captured in the embeddings because the loss function attempts to push similar samples closer together in embedding space and dissimilar samples farther apart.

The features mentioned in Section 3.1.1 (the food class labels ($\mathcal{F}$), slice type labels ($\mathcal{S}$), playing audio features ($\mathcal{A}_{play}$), cutting audio features ($\mathcal{A}_{cut}$), proprioceptive features ($\mathcal{P}$), and combined audio and proprioceptive features ($\mathcal{A}_{play}+\mathcal{P}$)) are used as similarity metrics to identify similar and dissimilar samples. During training, triplets are randomly formed using these positive and negative identifiers.

With this approach, the sensor data is incorporated in the triplet loss during training and is not needed at test time. This enables the learned embeddings to be used online during robot manipulation tasks without the need for access to the multimodal sensor data information that was present during training. Once the embeddings are trained, we evaluate them in various classification and regression tasks to understand if they can be useful in characterizing material property information. To perform these evaluation, we train multiple 3-layer multi-layer perceptron classifiers and regressors (in a supervised manner) for a variety of tasks, using the learned embeddings as inputs. These results are presented in Section 3.2.2. When combining multiple modalities, we concatenate the embeddings output from each separate network.

Auxiliary Study with Additional Cooked vs. Uncooked Food Data

We also collect an additional dataset with boiled foods to study our method's ability to detect whether or not a food is cooked using interactive perception. Details of this auxiliary study on a boiled vs. unboiled food dataset are in Section 3.2.2.

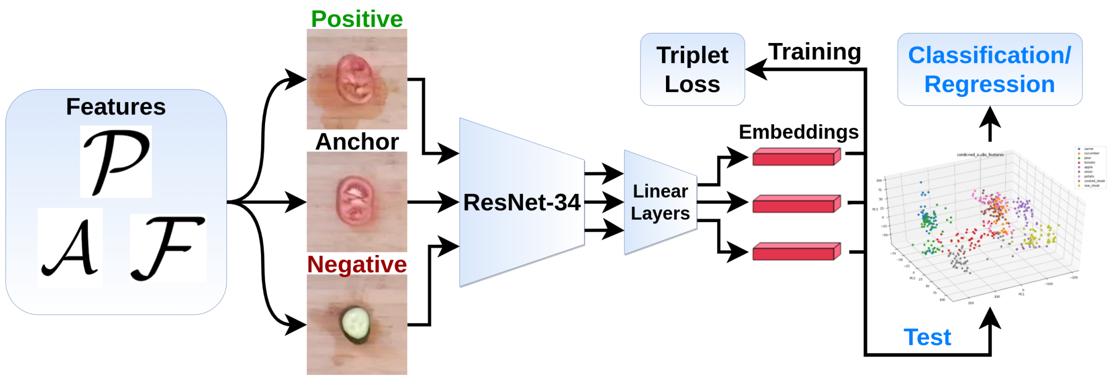


Figure 3.6: An overview of our approach to learning embeddings using sensor data. The features (sensor modalities) defined in Section 3.1.1 are used to form triplets to learn embeddings in an self-supervised manner, which are then used for supervised classification and regression tasks (blue text).

3.2 Results - Learning Food Representations

3.2.1 Collecting Real-World Multimodal Food Dataset

Figure 3.7 shows examples of the MFCC features from the grasp audio data for tomato, mozzarella, celery, and carrot samples plotted as log power spectrograms. This plot allows us to qualitatively visualize high-level differences in the audio features between samples. Next, Figure 3.8 shows similarity matrices for grasp audio and the combined playing audio features, with darker blue indicating high correlation and lighter green/yellow indicating less correlation between samples. The checkerboard pattern indicates that there are subsets of food types that are correlated to other food types. For example, raw/cooked steak appear to be correlated, as do lemon/orange, while spam/carrot and carrot/bread are relatively uncorrelated.

Figure 3.9 shows the push-down audio and combined playing audio MFCC features visualized in PCA space for the boiled vs. unboiled auxiliary dataset. We can see there are distinct clusters by food type in the feature space. The boiled vs. unboiled classes also show separation in feature space, indicating distinct differences in the audio data between the cooked and uncooked foods.

Finally, Figure 3.10 shows mean and standard deviations of the peak z-axis cutting forces by food type. We note that some food that were highly complaint and deformable registered very high cutting forces because the robot's knife was in contact with the cutting board while cutting these foods (which generates high z-axis forces). Because of the varying levels of contact with the cutting board for each sample, this force data is quite noisy.

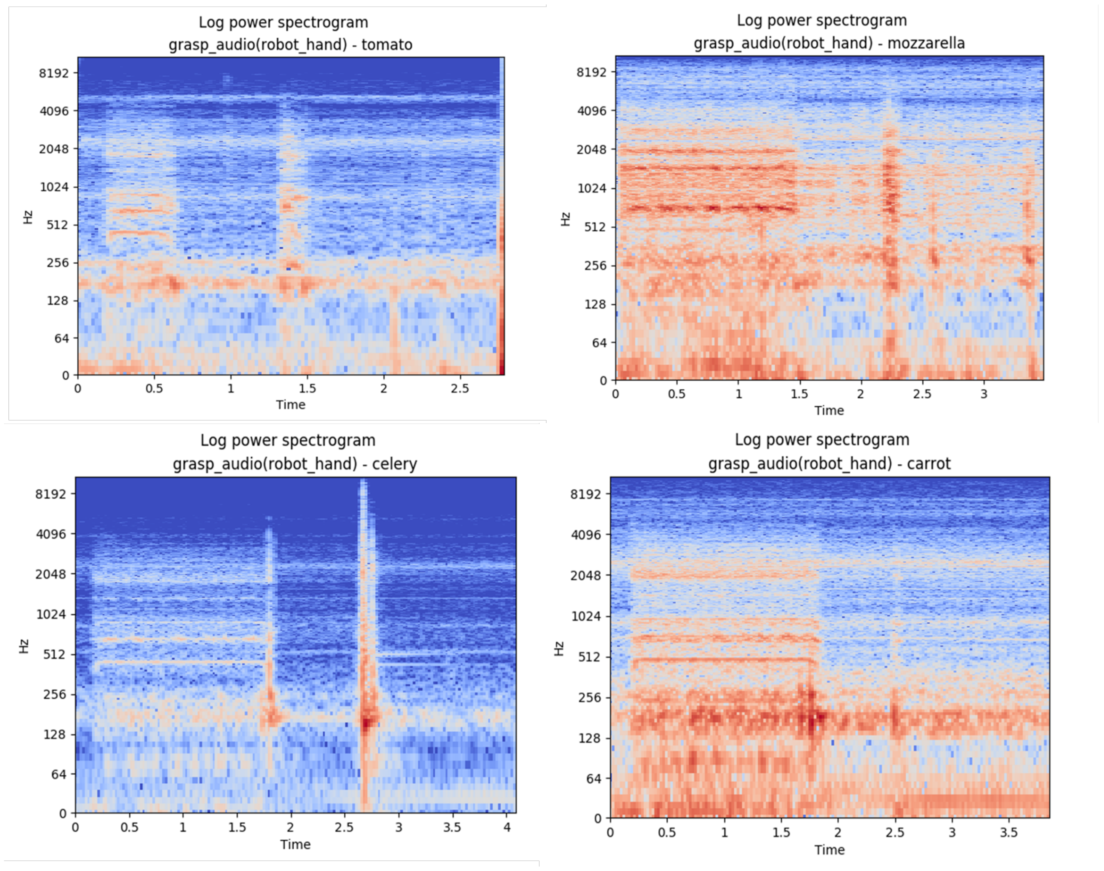
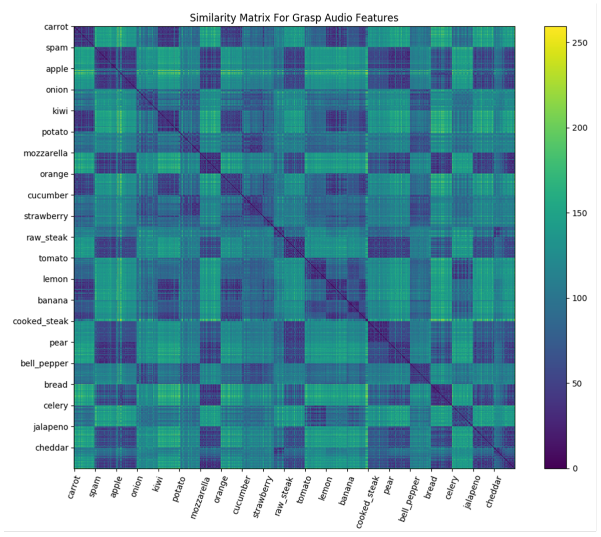


Figure 3.7: Examples of spectrogram of audio MFCC features from contact microphone data collected during play grasp interactions.



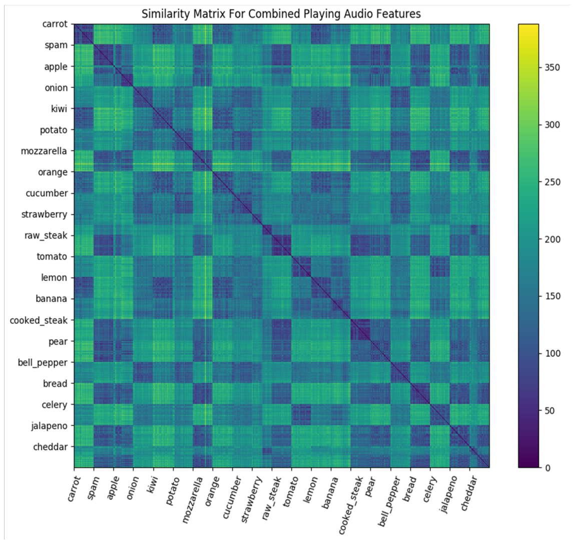


Figure 3.8: Similarity matrices for grasp and combined playing audio features.

3.2.2 Learning Embeddings from Multimodal Data

9-class Dataset Results

We first collect a 9-class playing dataset with no robot cutting (i.e. the human performed the cuts to create the slices) and evaluate our embedding learning approach on this dataset. The food classes in this dataset are: apple, carrot, cooked steak, cucumber, onion, pear, potato, raw steak, and tomato. Table 3.1 shows classification results for food type and hardness (0,1,2) and regression results for predicting slice width using embeddings trained with $\mathcal{F}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{A}$+$\mathcal{P}$, and a visual-only baseline with ResNet. While the ResNet baseline outperforms the embeddings for food type classification, the combined audio and proprioceptive embeddings ($\mathcal{A}$+$\mathcal{P}$) perform the best on the hardness and slice width prediction tasks.

21-Class Dataset Results

We next expand on the previous dataset and collect a 21-class dataset with both cutting and playing data, where food class label ($\mathcal{F}$), slice type label ($\mathcal{S}$), playing audio features ($\mathcal{A}_{play}$), cutting audio features ($\mathcal{A}_{cut}$), proprioceptive features ($\mathcal{P}$), and combined audio and

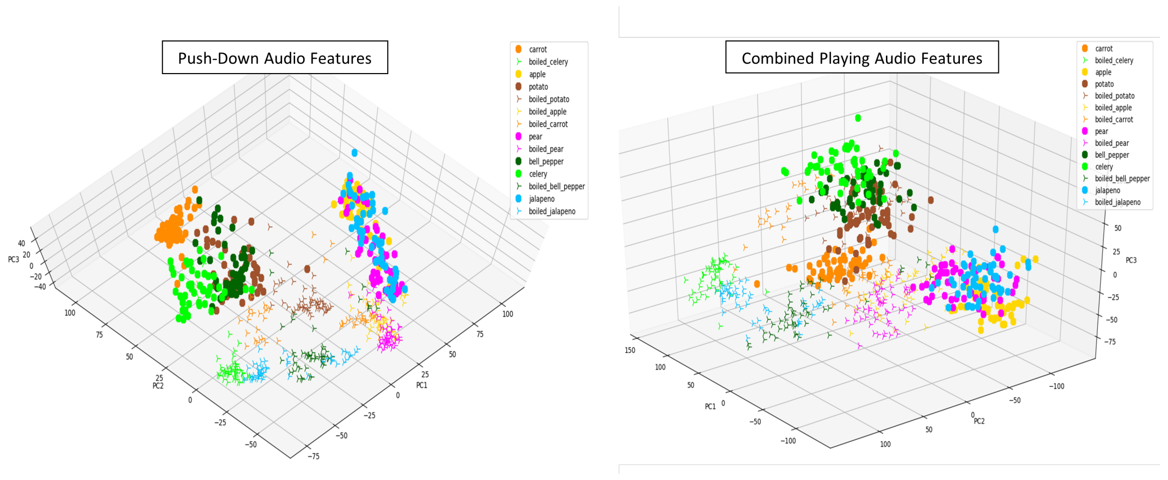


Figure 3.9: PCA of audio MFCC features for push-down audio and combined playing audio features.

Embeddings	**Food Type Accuracy (%)**	**Hardness Accuracy (%)**	**Slice Width RMSE (mm)**
$\mathcal{F}$	95.56	73.02	11.2
$\mathcal{A}$	80.00	57.75	11.4
$\mathcal{P}$	67.78	85.35	6.8
$\mathcal{A}$+$\mathcal{P}$	86.67	**87.60**	**6.6**
ResNet	**100.00**	17.21	17.9

Table 3.1: Results with a 9-class dataset on 3 evaluation tasks for different learned embedding networks.

proprioceptive features ($\mathcal{A}_{play}$+$\mathcal{P}$) are used as similarity metrics for forming triplets in the different embedding networks. Once the embedding networks were trained, we used the learned embeddings to train multi-layer perceptron classifiers and regressors (as discussed in Section 3.1.2) to predict various properties such as food type labels (21 classes), slice width prediction (i.e. gripper width post-grasp), hardness classification (three human-labeled classes describing food as hard, medium, or soft), juiciness classification, i.e. describing the liquid content of the food (three human-labeled classes describing food as juicy, medium juiciness, or dry), and finally slice type classification (14 classes based on the types of cuts the robot used to generate the slice).

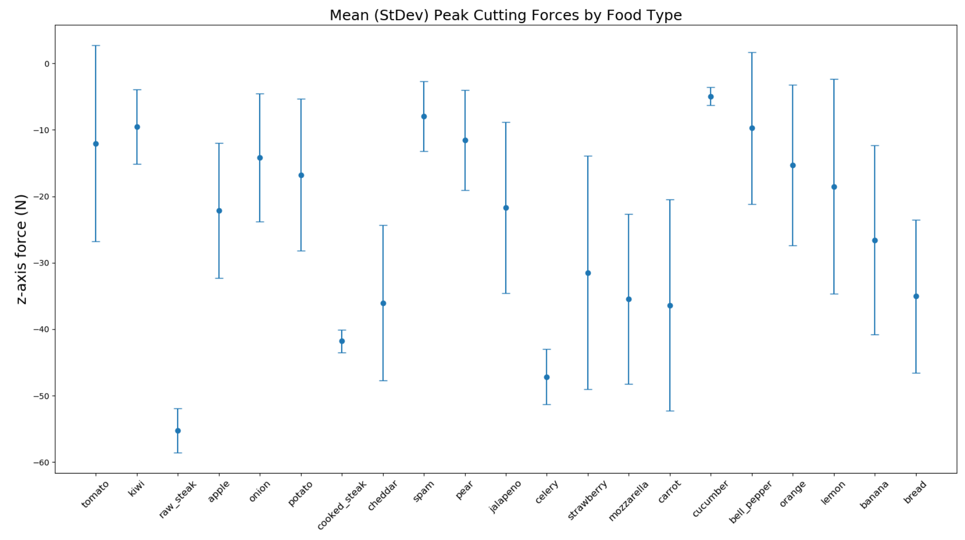


Figure 3.10: Mean and standard deviation of z-axis cutting forces by food type.

We compare our embedding approach with two baselines: one is a visual-only baseline that uses the pre-trained ResNet convolutional neural network, and the other is a three-layer MLP that takes the $\mathcal{A}_{play}$ data as input (rather than image input) and tries to directly classify based on audio input.

We assess how well our embedding approach can generalize to unseen food types by using a leave-one-out-classification approach for the hardness and juiciness classification tasks. For each of the tasks, we trained a classifier using the learned embeddings and left an entire food type out of the training set and tested the classification performance on the held out food type. We performed this leave-one-out classification 21 times, leaving each food type out of the training set and evaluating classification performance on the held-out food type. The results shown in Table 3.2 reflect the average classification accuracy across all 21 leave-one-out trials for hardness and juiciness classification.

Results with this approach on the 21-class dataset are shown in Table 3.2.

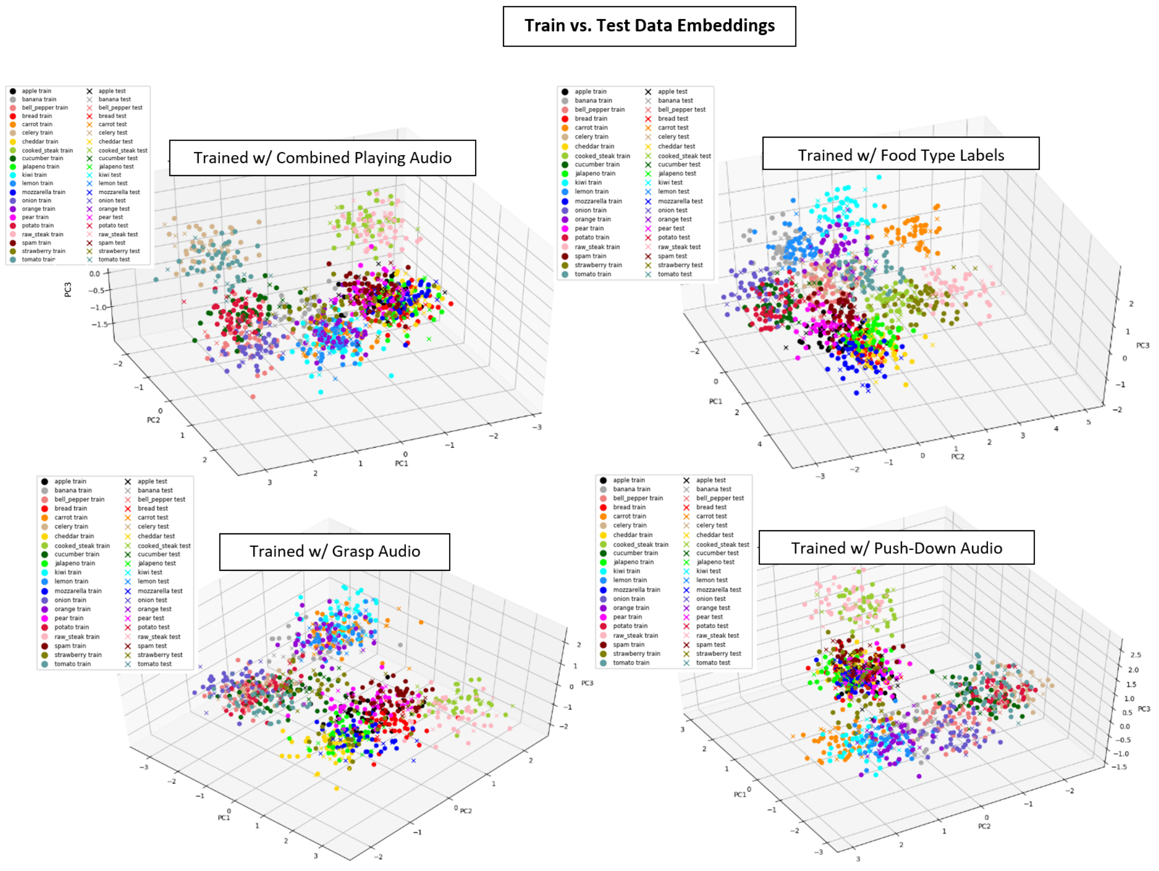


Figure 3.11: Train vs. test embeddings visualized in PCA space for some of the embedding networks (grasp audio, push-down audio, combined playing audio, food type). Train data is shown in solid circles, test data is shown in solid X's.

Auxiliary Study with Additional Cooked vs. Uncooked Food Data

As a supplement to the dataset discussed above, we collected an additional dataset consisting of various boiled foods to further explore whether our method can detect whether or not a food item is cooked through interactive sensor data. We added seven additional boiled food classes (apples, bell peppers, carrots, celery, jalapenos, pears, and potatoes) and boiled each food slice for 10 minutes. Due to difficulties with grasping the cooked foods, we did not have the robot perform the cuts to generate slices of these foods. Instead, we had a human cut slices of the boiled foods. We then ran through the playing data collection process described in Section 3.1.1.

Embeddings	Food Type Accuracy - 21 classes (%)	Hardness Accuracy - 3 classes (%)	Juiciness Accuracy - 3 class (%)	Slice Type Accuracy - 14 class (5)	Slice Width RMSE (mm)
$\mathcal{F}$	92.0	40.7	36.6	12.9	10.9
$\mathcal{S}$	17.1	37.0	34.9	**40.5**	11.8
$\mathcal{A}_{play}$	85.7	35.0	**46.0**	17.1	9.9
$\mathcal{A}_{cut}$	93.5	33.5	45.6	16.8	11.3
$\mathcal{P}$	49.5	**47.1**	37.0	20.0	**7.9**
$\mathcal{A}_{play}$+$\mathcal{P}$	83.8	36.4	40.2	21.4	9.5
ResNet	**98.9**	34.9	36.5	30.0	13.9
Classifier w/ $\mathcal{A}_{play}$ as input	84.4	40.8	34.0	30.1	34.4

Table 3.2: Baseline and multi-layer perceptron results for 21-class dataset on five evaluation tasks using different learned embedding networks that were trained on our full dataset.

To train the embeddings, we combined the seven boiled classes with their uncooked counterparts from the 21-class dataset to form a 14-class dataset. We then conduct a subset of the classification performance evaluations (all three tasks evaluated with leave-one-out approach) on the learned embeddings from this dataset, results of which are shown in Table 3.3. Additionally, Figure 3.12 shows the combined audio learned embeddings visualized in PCA space for this dataset.

Embeddings	Hardness Accuracy (%)	Juiciness Accuracy (%)	Cooked Accuracy (%)
$\mathcal{A}_{play}$	98.0	62.9	98.9
$\mathcal{P}$	63.0	68.4	60.6
$\mathcal{A}_{play}$+$\mathcal{P}$	**99.7**	**70.6**	**99.1**
ResNet	90.5	66.1	90.4
Classifier w/ $\mathcal{A}_{play}$ as input	82.1	67.4	88.8

Table 3.3: Results on three evaluation tasks for different learned embedding networks that were trained using the auxiliary cooked vs. uncooked dataset.

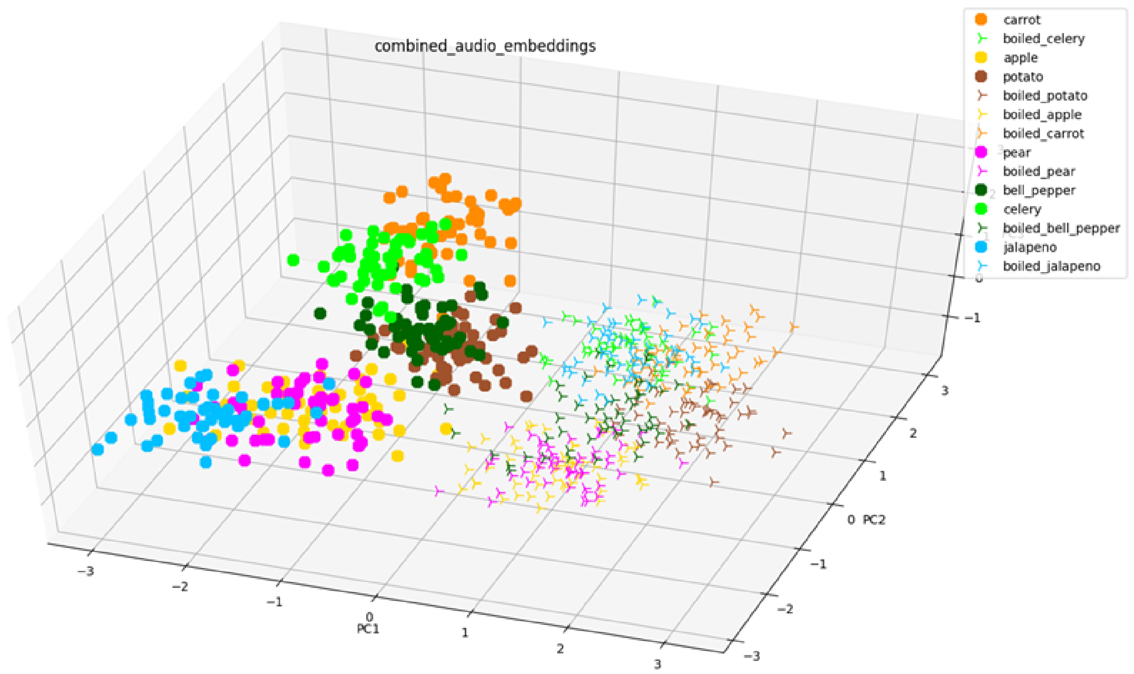


Figure 3.12: Combined playing audio learning embeddings from boiled vs. unboiled 14-class dataset visualized in PCA space.

3.3 Discussion - Learning Food Representations

For the 21-class dataset, we see that the visual ResNet baseline performs better than our embeddings in the food type classification task. This result makes sense, because ResNet is pre-trained on a huge dataset of labeled images in the ImageNet dataset. This prior data advantage makes ResNet more effective at classifying food type, which is a task that is relatively easy to do with just visual data. However, on the other four tasks (hardness, juiciness, slice type classification, and slice width prediction), our embeddings outperform the ResNet baseline. This is likely because these tasks require more than purely visual information since they involve understanding the material properties of the object, and the ImageNet dataset does not have information on the physical properties of the food items. Also, ResNet is trained to differentiate classes, rather than identifying similarities across classes. Therefore, it had more difficulty generalizing to unseen data in the leave-one-out classification task.

The results from our approach indicate that the embeddings are capturing material information via the different multimodal features, and encoding these in the learned representations without the need for explicit human labels during embedding learning. It seems that our approach generalizes better to unseen data compared to the supervised, purely visual baseline. The results also indicate that different types of embeddings (i.e. embeddings trained with different types of sensor data features) are useful for different tasks. For example, the audio embeddings provide information that helps to distinguish between food type and level of material juiciness, while the proprioceptive embeddings seem to be more effective at encoding hardness and slice width. This makes intuitive sense as the proprioceptive features are correlated to the object's stiffness characteristics.

Because no explicit material property labels were provided during embedding network training, the information encoded in the embeddings was learned in a self-supervised manner. We also note that some food types, such as tomatoes, were more difficult to classify when left out of the training dataset during the LOO evaluations than others, such as carrots. This could be attributed to the fact that our dataset is quite diverse but also relatively small, so there are not a lot of food classes that have very similar material properties within the dataset. Finally, we note that the slice type classification accuracy was poor across all methods, and this is likely due to the inherent difficulty in classifying slice type in this dataset. There was a lot of variability in shapes between the food items, so the slices generated by the robot's cuts were not very consistent and the slices resulting from the robot performing the same cutting actions varied greatly across food classes. Therefore, it is not an unexpected result that the only embeddings that performed relatively well on this task were the embeddings trained using slice type labels.

We also note the differences in classification performance between the 9-class and 21-class dataset - the classification performance decreased on the 21-class dataset compared to the 9-class dataset using the same embedding learning approach. This is likely because the additional food classes in the 21-class dataset introduced a much larger amount of variability and noise, making the classification tasks more difficult. The classification performance on the 21-class dataset could potentially be improved through a more controlled and structured data collection process. For example, the robot grasps objects from a fixed angle and starting location rather than trying to align the hand with the object first. However, the goal of this study was to allow for autonomous, unstructured playful interactions without injecting too much structure or prior knowledge of more targeted exploration behaviours, so this is an area for future work.

Looking at the auxiliary study with the boiled vs. unboiled foods, we observe that the playing audio embeddings are effective at distinguishing between cooked and uncooked foods, without the need for human labels. $\mathcal{A}_{play}$+$\mathcal{P}$ shows superior performance on all three leave-one-out classification tasks (hardness, juiciness, and cooked classification), indicating that our approach can generalize well to unseen data and is capturing similarities across food classes with similar material properties (e.g. apples and pears, potatoes and carrots, bell peppers and jalapenos). We also observe that on this dataset, combining the audio and proprioceptive embeddings showed better performance than each of these embeddings individually, which indicates that additional information is gained by combining the different embedding types. We see in Figures 3.9 and 3.12 that there is a clear separation between the cooked vs. uncooked food items in both the audio feature space and the learned embedding space. Within the cooked and uncooked clusters, certain food types tend to cluster close together, such as the uncooked pear and uncooked apple, which makes sense given the similarity of these two fruits. Further, they remain clustered close to one another after they are cooked, even though there is a shift in the feature space between cooked and uncooked. This behavior has interesting implications for using these embeddings in manipulation tasks to identify whether foods are cooked or not (or even to what degree they are cooked), as this would be a useful skill for a cooking robot to have. Also, this study of cooked vs. uncooked foods is a concrete example of when two foods might visually look the same but have drastically different material properties, so visual approaches alone would not be sufficient and our approach may become useful.

These results indicate that embeddings trained with different sensor modalities can encode food material property information that is not necessarily captured in a purely visual approach, and can therefore be selectively used (or combined) in different use cases.

In summary, our contributions from this section are: first, a unique multimodal food interaction dataset, and second, a method of using the multimodal sensor data to learn embedding representations that encode food material property information. We envision the learned embeddings may be useful in skill parameter selection for a variety of food manipulation skills, thereby working towards the goal of a material-aware robot that can adapt its actions based on the type of food it is handling.

Chapter 4

Learning Robotic Food Cutting

In Chapter 3 we discussed the idea of having the robot reason about diverse food materials through interaction and focused on how to represent food slices. We next visit the related research problem of having the robot learn to adapt manipulation behaviors to foods with varying material properties, a key element to generalizing robot behaviors in the kitchen environment. In this chapter, we focus specifically on teaching a robot food cutting skills and exploring its ability to adapt cutting policies to foods with different material properties using reinforcement learning. We break down the overall goal of learning robotic food cutting skills into a series of sub-problems. Figure 4.1 below shows a high level overview of our approach, which first involves learning cutting skills from human demonstrations and then fine-tuning these policies for a range of food material types using an RL framework. Within the latter, we explore defining rewards for policy optimization in two different ways: first with using a hand-defined analytical reward function, and second with human-specified rewards. We will elaborate on the different steps shown in Figure 4.1 in the sections in this chapter.

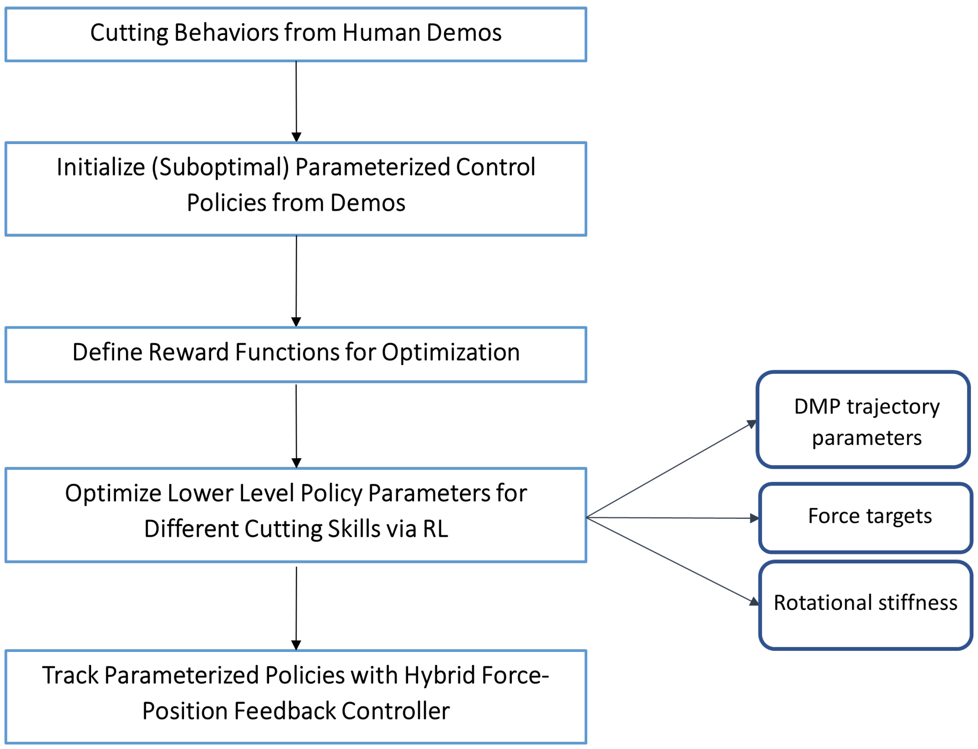

Cutting Behaviors from Human Demos
Initialize (Suboptimal) Parameterized Control Policies from Demos
Define Reward Functions for Optimization
Optimize Lower Level Policy Parameters for Different Cutting Skills via RL
Track Parameterized Policies with Hybrid Force-Position Feedback Controller
DMP trajectory parameters
Force targets
Rotational stiffness

Figure 4.1: High level overview of our approach to learning robot food cutting skills.

4.1 Learning Cutting Skills from Demonstrations

We first explore how to learn and parameterize different food cutting skills from human kinesthetic demonstrations. We use Dynamic Movement Primitives (DMPs) to fit smooth parameterized trajectories to the raw end effector pose trajectory data collected during the human skill demonstrations. For our application, we use the DMP formulation shown in Equations (4.1) and (4.2) used by Zhang et al [38]. α_z and β_z are spring and damper coefficients, ϕ_j is a scaling parameter that we set to 1 in our case, τ is a time constant, and x is the canonical system state that starts at a value of 1 and decays as a function of $\dot{x} = -\tau x$. It functions as a timer to synchronize multiple linear ODE systems [38]. The DMP weight parameters w_j in the forcing function shown in Equation (4.2) are learned via linear regression using the raw trajectory data (robot end effector x-y-z positions) as ground truth [38]. These parameters are weights on the Gaussian basis functions ψ_k that are summed to generate the trajectory, so they define the shape of the trajectory. ψ_0 is a Gaussian basis function that follows a minimum jerk trajectory that goes from 0 to 1. We use 6-8 Gaussian basis functions to represent each of the different cutting behaviors, which is found to be a sufficient number of basis functions to capture the complexity of the motions without being too many such that the learned DMP trajectory starts to overfit to the noise present in the demonstrations.

$$\ddot{y} = \alpha_z(\beta_z \tau^{-2}(y_0 - y) - \tau^{-1}\dot{y}) + \tau^{-2}\sum_{j=1}^{M} \phi_j f(x; w_j) \tag{4.1}$$

$$f(x; w_j) = \alpha_z \beta_z \left(\frac{\sum_{k=1}^{K} \psi_k(x) w_{jk} x}{\sum_{k=1}^{K} \psi_k(x)} + w_{i0}\psi_0(x) \right) \tag{4.2}$$

We demonstrate the following eight cutting behaviors to the robot and fit parameterized DMP policies to each skill:

1. Planar back and forth movement (referred to here as a normal cut)
2. Normal cut movement with a rotated left and right initial roll angle (referred to as angled left and angled right cuts)
3. Dice (fast straight up and down movement)
4. Pivoted chop (downward movement with rotational movement along the pitch axis)
5. Moving pivoted chop (previous movement with additional x movement during motion)
6. Scoring (backward x movement with downward force maintained in contact with the cutting board)
7. Between-finger cut (normal cut rotated 90 degrees for cutting the chunk of food grasped in between the robot's fingers)

We kinesthetically demonstrate 15-20 trajectories to the robot for each cutting skill type, and fit a parameterized DMP policy for each demonstration, where the parameters are referred to as the DMP weights and are unique to each skill.

Figure 4.2 shows raw x-y-z end effector trajectory data collected from multiple human demonstrations of the eight different cutting behaviors (Top left to right: normal cut, pivoted chop, scoring, angled right. Bottom left to right: moving pivoted chop, in hand cut, dice, angled right). Figure 4.3 show examples of overlaid raw trajectory data with fitted smooth DMP trajectories, using learned weights. Dashed lines show the fitted DMP trajectories, and solid lines show the raw trajectories.

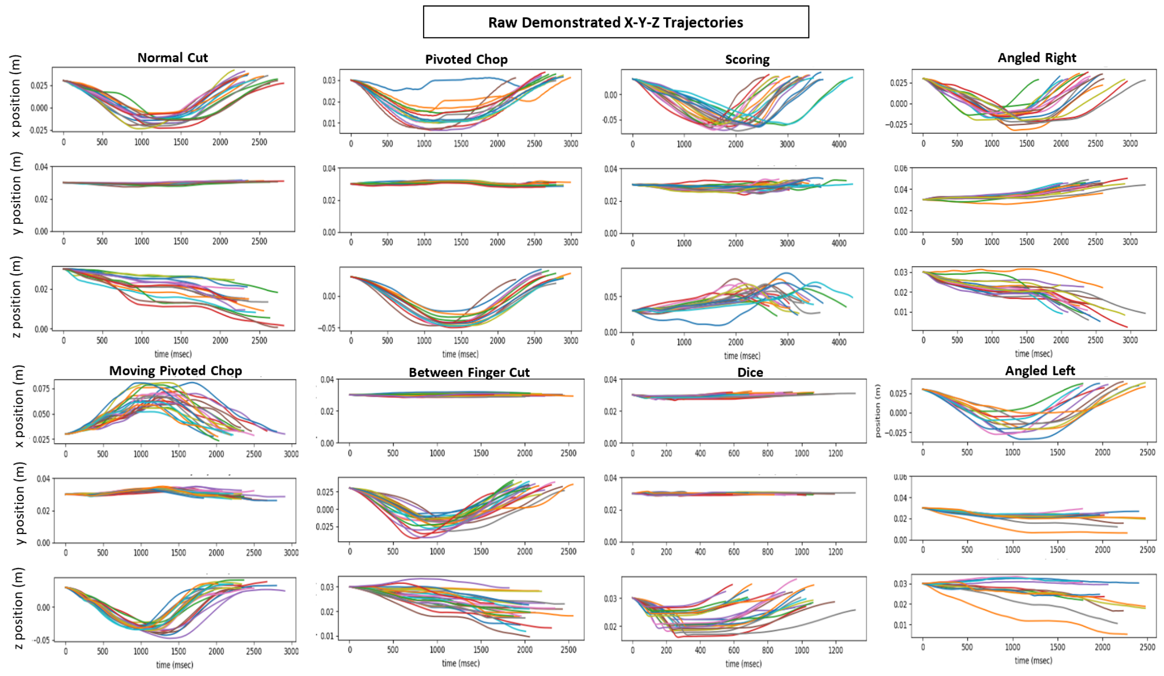


Figure 4.2: Raw end effector x-y-z trajectory data collected during human demonstrations of eight different cutting skills.

4.1.1 Dimensionality Reduction in DMP Weight Space

A research thread of interest was to better understand the feasibility of finding a generalized skill parameterization that covers all the eight different cutting behaviors. This generalized action space parameterization would remove the need to have separate parameterizations for each cutting skill and could then be sampled from and used to generate random cutting behaviors that

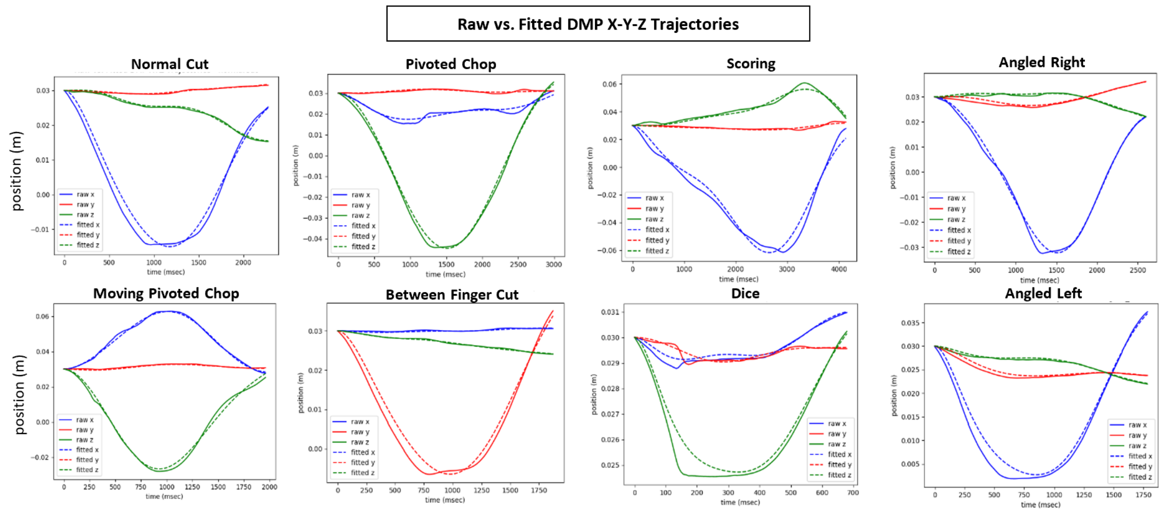


Figure 4.3: Examples of fitted DMP trajectories overlaid with raw trajectory data collected for the eight different cutting skills. Dashed lines shows the smooth fitted DMP trajectories, and solid lines show the raw trajectories.

lie within the spectrum of the seven demonstrated behaviors. This ability to randomly sample and generate cuts could be useful from an RL perspective, where the robot could theoretically explore the space of cuts by sampling in this space. To explore this idea, we look at the DMP weights learned from each demonstrations of the different cutting skills and perform a dimensionality reduction using PCA to look at the principal components of the combined weight data. Each axis dimension (x, y, and z) has an associated set of six weight parameters (corresponding to six Gaussian Basis functions in the DMP), and we do PCA separately for each axis and look at the resulting eigenvectors corresponding to each of the principal components for the weight data for each axis.

Figure 4.4 shows a visualization of PCA performed on each dimension (x, y, z) of the combined DMP weight parameters for the eight different cutting skills. In the x and z dimensions, the scoring and moving pivoted chop cutting skills lie clustered farther away from many of the other cutting skills in PCA space, while normal cut, angled slice left and right, and in hand cut cluster together. This makes sense because these skills are all similar motions, while scoring and moving pivoted chop are quite different cutting motions than the other skills in the set. The dice data is tightly clustered both in the x and z dimension, which also makes sense because the dice is a simple up and down movement with little to no other movement. Finally, there is a lot of variability and not many identifiable clusters in the y dimension, which makes sense because these cutting skills do not involve y-axis movement so the information captured by the DMP weight parameters in this dimension is largely noise from the demonstration data. Figure 4.5 shows the percent variance explained as a function of the number of principal components (1-6), and indicates that 99% variance explained is achieved with three principal components for each of the x, y, and z dimensions. Finally, Figure 4.6 shows the eigenvector weights associated with each of the six principal components rolled out as trajectories to visualize the interpretability of each of the different principal components in terms of the x-y-z motions they represent. The eigenvector trajectories (shown in solid lines) are overlaid with the mean x-y-z trajectory (shown in dashed lines) for the whole dataset to compare them visually relative to the mean. We can see that the first three eigenvectors show x and z back and forth movement with differing amplitudes relative to the mean, while the last three eigenvectors are slightly less interpretable and may be representing some of the more dynamic components present at the beginning and ends of the different cutting skills.

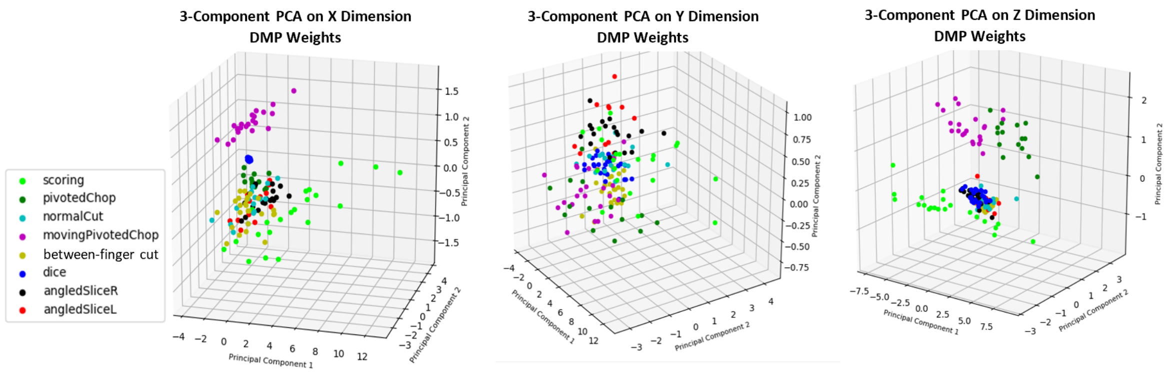


Figure 4.4: PCA performed on fitted DMP weight data for trajectories from all 8 cut types combined, separated by dimension (x, y, and z).

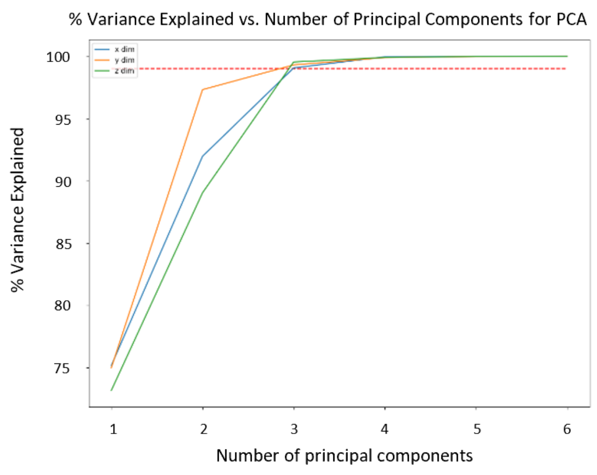


Figure 4.5: Percent variance explained vs. number of principal component for PCA done on all cut type data in the DMP weight space. Red dashed line indicates 99 percent variance explained.

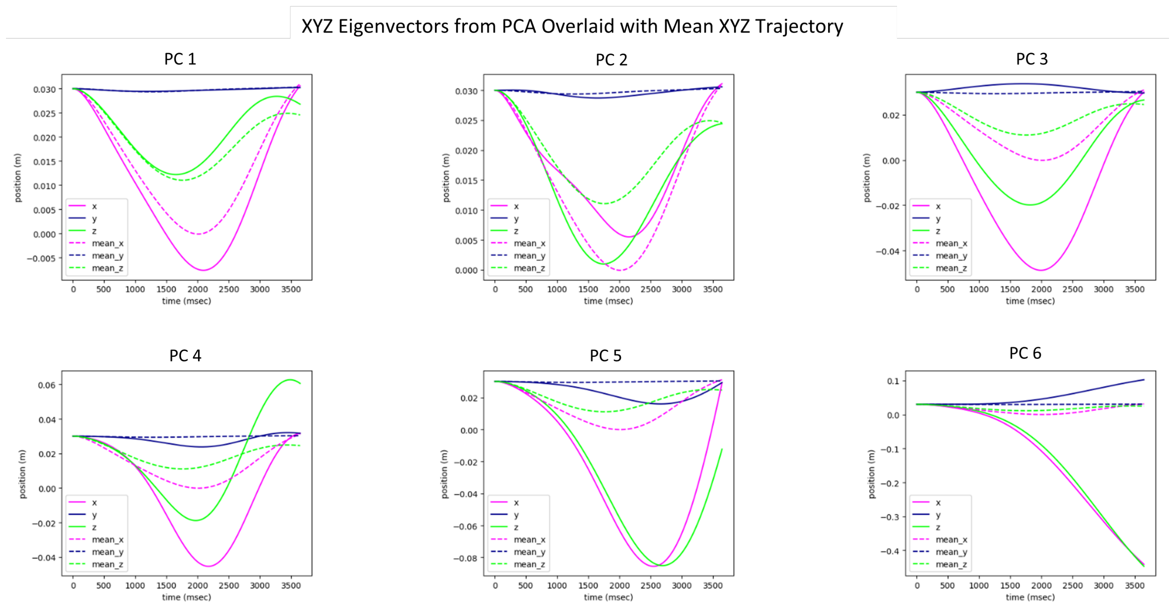


Figure 4.6: DMP trajectories computed using the eigenvectors corresponding to each of the 6 principal components (shown in solid lines) overlaid with the mean DMP trajectory for the combined dataset (shown in dashed lines).

Based on these results, we chose to separate the parameterizations of the different cutting skills so that each skill has its own set of DMP weight parameters to make sure the different x-y-z motions inherent to each of the skills are captured. Additionally, the PCA results in Figure 4.4 indicate that the pivoted chop (and moving pivoted chop) and scoring skills are fairly different than the rest of the skills, while normal cut and the rest of the skills are fairly similar. We therefore choose to focus on the normal cut, pivoted chop, and scoring skills for the reinforcement learning experiments discussed in Section 4.2, since these skills are representative of the range of cutting motions studied here.

4.1.2 Controller Implementations

Implementation to Reduce Unintended Noise In Demonstrations

Because most cutting behaviors consist of planar x-y-z movements, we modified our Franka arm's Cartesian impedance controller to add stiffness constraints in the local end effector frame (rather than global frame), such as high stiffness in the end effector frame's y-axis, i.e. across the knife's blade. See Figure 4.7 for visual.

Since the cutting skills we look at do not involve y-axis movements (only x and z), this enabled collection of better human demos in various starting orientations, such as the angled cuts, with less noise and unintended movements. To implement this, we take the general form of the Cartesian impedance controller for our Franka arm and do the following:

- Compute position and orientation error (e)
- Transform position and orientation error into the local end effector frame: $e_{local} = R^T \cdot e$
- Apply translational and rotational stiffnesses (grouped as K_P) to the error in the local end effector frame using $K_{P_{local}} = K_P \cdot e_{local}$

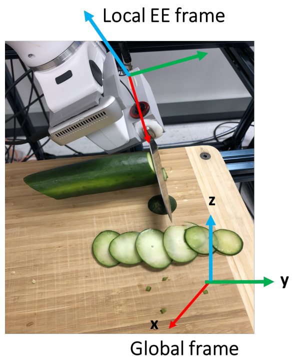


Figure 4.7: Global vs. local end effector reference frames.

- Transform the position and orientation back to the global reference frame using $e_{pose} = R \cdot K_{P_{local}}$
- Perform similar transformation of velocity error into local reference frame, apply translational and rotational damping coefficients (grouped as K_D) in local frame, and then transform back into global reference frame: $e_{velocity} = R \cdot K_D \cdot R^T \cdot Jdq$
- Finally we compute the control input torques using the transformed errors using the following impedance control law:

  $u = g(q) + J^T\left(e_{pose} + e_{velocity}\right)$

Implementation to Facilitate Flexible DMP Skill Execution on Robot

Calculating the DMP trajectory in the local end effector frame allows better generalization of learned cutting behaviors by enabling a movement learned from demonstration in one reference frame to be executed in another without needing to re-learn the DMP parameters when starting in a different initial orientation than the demonstrated trajectory. For example, because the angled cuts are a rotated version of the normal cut behavior (i.e. normal cut with a different initial orientation), we treat the angled cuts as normal cuts with different initial conditions and execute them by executing the normal cut DMP policy in the local end effector reference frame with an approximately +/- 20 degree starting roll angle.

To implement this, we define the rotation between the DMP cutting frame to the end effector frame (R_{dmp-EE}), and the initial orientation and position of the end effector, R_0 and p_0. Then, at each timestep, we compute the new x-y-z trajectory positions (p_{new}) as:

$p_{new} = R_0(R_{dmp-EE}(p - p_0)) + p_0,$

where p_{new} is a 1x3 array of x-y-z target positions at each timestep.

The new target positions are sent to the Cartesian impedance feedback controller, which commands the robot via a torque control interface at each timestep, in a 1 kHz control loop, to move the end effector to the desired positions.

Hybrid Force-Position Control for Cutting Skill Execution

Hybrid force-position control (HFPC) has been used in many contact-rich robotic manipulation tasks. Because the cutting tasks are contact-rich and largely planar motions, they naturally lend themselves to position control in some axes and force control in others. Once we have parameterized DMP trajectories for each of the cutting skills and estimated z-axis downward forces based on prior data, we use a hybrid force-position controller to track the desired trajectories when executing the skills on the Franka robot. The x-y-z axis controller combination is defined by a 1x3 vector S, where each element of S is either 0 or 1, with 0 indicating force control for a given axis and 1 indicating position control for a given axis.

Implementation of DMP Trajectory Generator and HFPC Interface on Franka Robot

Our Franka robot arm's control stack involves a set of real-time controllers (including the HFPC) implemented in C++ on a control PC, and a Python interface to the control PC using Protocol Buffers (Protobuf), as discussed in detail in [40]. To generate the position trajectory targets to send to the force-position-controller, we compute the DMP trajectories for the entire skill on the Python side, and then send them to the control PC via Protobuf messages at 100 Hz. For axes that use force control, we assume constant forces over the course of the trajectory and send over these force targets with the position trajectories in a message at each timestep of the trajectory.

4.2 Optimizing Cutting Skills Using Reinforcement Learning

In this section we look at formulating the RL problem of fine-tuning cutting skills learned from human demonstrations and adapting skill parameters to different food material types. We explore the question of how to define rewards for robot cutting tasks and experimentally evaluate policy learning first using a hand-defined generalized reward function, and finally simultaneous policy learning and reward model learning from human input.

4.2.1 Problem Formulation

Once we have parameterizations for each of the cutting skills, as discussed in Section 4.1 above, our goal is for the robot to optimize (i.e. fine-tune) control policies for different cutting behaviors and food material types. This capability is useful because it is somewhat challenging for a human to naturally guide the robot through cutting motions due to the complexity of the interactions between the knife, grasped food, and cutting board. Therefore the human demonstrations for each skill are not necessarily optimal and we want the robot to improve upon them. Additionally, we do not want to have to rely on having human demonstrations for cutting skills on every possible food material type, as this would be infeasible. Thus we want a method that allows the robot to start from some base policy initialized from a demonstration on just one food type, and then to fine-tune this policy through interaction (i.e. using reinforcement learning) with new food types that have a range of material properties.

We experimentally evaluate three different cutting behaviors (normal cut, pivoted chop, and scoring) across six different food types - three hard foods (potato, carrot, and celery) and three soft foods (mozzarella, tomato, banana). As discussed in Section 4.1.1, we chose these three cutting skills because they cover a range of knife skills that one might use on different foods in the kitchen, and the other cutting skills we demonstrated in Section 4.1 are largely variations of these three skills. The six foods were chosen because they represent a wide range of materials properties, even within the hard and soft groups. For example, tomato, banana, and mozzarella are all relatively soft, but tomatoes are very juicy and have different material properties as you move through their cross-section, while bananas are more uniform but easily mushed, and mozzarella is more “springy” but tears or shreds easily.

We formulate the skill learning problem as a hierarchical action space problem, where the high level actions are the x-y-z controller combinations (force vs. position) and the low level actions are the DMP position trajectory parameters, z-axis force targets, and pitch axis stiffnesses (all other stiffness are constant and set to high). We isolate pitch stiffness because this axis is the only dynamic rotational axis of interest in the cutting behaviors studied.

4.2.2 Developing an Analytical Reward Function

A key research question in the pipeline of learning and optimizing robot cutting skills is how to define reward functions for the task of food cutting. We also want to understand the feasibility of defining a general reward function that can be used to optimize across different cutting skills and food material types. This would be useful so we can avoid the time-intensive task of hand-engineering specific reward functions for each of the different cutting behaviors. In this section we discuss the two methods of defining rewards for food cutting that we focus on in this thesis.

We first experiment with cut type-specific rewards to understand which features are important in the different skills. For example, we experiment with different reward functions containing penalties for x-y-z forces and movement, and additionally terms that penalize for not returning to the starting position (e.g. for the pivoted chop, adding a term penalizing for the difference between the up and down movements, or for the normal cut, adding a term penalizing the difference between forward and backward movements). While these cut-type specific reward functions seem to work for the individual cutting tasks, they require a fair amount of engineering and rely on our knowledge of the specific cutting skill and what we as the human expect the robot to do. Therefore, we aim to define a more generalized analytical reward that can be more broadly applicable to a range of cutting skills and food types.

When thinking about the task of food cutting, we can intuit that forces, displacement, and cutting speed are likely important features in determining the success of a cut. From a visual standpoint, it seems that an image of a slice could also be used to determine the success of the cut that generated the slice. However, in this study we decided not to use an image-based reward approach for a few reasons. First, we observed that often times bad cutting policies did not manifest in the slice generated itself, but rather resulted in the robot destroying or squishing the remaining section of the food being grasped in the gripper. Thus, the final slice produced is not always indicative of the quality of the skill execution. Additionally, oftentimes in a bad policy execution, the robot is not able to complete the cut and only generates a partial slice that is still attached to the food in the gripper. This makes inferring slice quality (and therefore reward) from an image quite difficult.

Based on these observations, we select the following features in our reward function: y-z forces, x-y-z movement, and total time to complete cut (with a very high value place on this feature if the robot is unable to complete the cut). Note that we do not include x forces because the x-direction is the main cutting axis, and we do not care as much about minimizing forces in this direction, but rather we care more about the amount of displacement in this direction.

Finally, since the rewards are not sparse, we also define a task success metric to better quantify and compare the success of the learned policies from a task completion standpoint. We define this task success metric as 0, 1, 2, where 0 is a failure to complete the cut, 1 is if the robot completed the cut but the cut was “average”, and 2 is if the robot completed the cut and it was a “good” cut (“average” vs. “good” is subjective and determined by the human watching the execution). For the percent task success results shown in Section 4.2.6 and 4.2.8, both 1 and 2 are considered successful task executions since the cut was completed with both a score of 1 or 2.

We perform initial robot experiments to select adequate scaling weights for the features in the generalized analytical reward that work sufficiently for the tasks. Since the relative weights of the different reward terms greatly impacts the learned policy, we will explore in Section 4.2.7 how to use human input to learn rather than hand-select these scaling weights and tune the learned cutting behavior to a human’s preferences.

For the lower level parameter optimization experiments, we use a generalized analytical reward function as follows:

$$R = -0.1F_{y_{max}} - 0.15F_{z_{max}} - 10d_{x_{max}} - 100d_{y_{max}} - 10d_{z_{max}} - 100d_{up_{z_{max}}} - 0.2t,$$

where $F_{y_{max}}$ is the max y-axis force, $F_{z_{max}}$ is the max z-axis force, $d_{x_{max}}$, $d_{y_{max}}$, $d_{z_{max}}$ are the max x, y, and z displacements, respectively during the skill execution. The $d_{up_{z_{max}}}$ is a term penalizing specifically for upward z movements (since these skills all require downward movement towards the cutting board in order to achieve cut success we add an additional high penalty for this), and t is the total cut time. The variable t is assigned an arbitrarily high value of 200 if the robot does not successfully cut a slice, so this term incorporates a notion of task failure. This reward function seeks to minimize control effort while also minimizing the time to cut a slice, and the trade-off between these two priorities lies in how the weights are scaled for each of the terms. We run initial experiments to select scaling weights in a reasonable range, although it should be noted that the hand-selected weights are not necessarily optimal and can be selected to tune the cutting behavior in different ways. This will be discussed in greater detail in Section 4.2.2.

For the higher level controller combination selection experiments, we assume a fixed set of lower level parameters. Since the DMP weight parameters are not being explored and optimized, most of the cut executions are unsuccessful (i.e. unable to generate fully cut slices). To address the issue that arises when all samples receive bad rewards and there is no differentiation between samples, we add an additional term to the above generalized reward function for these experiments to penalize for deviating too far from the human demonstrated trajectories. This additional penalty term, Δ_{demo}, is shown in the reward below:

$$R = -0.1F_{y_{max}} - 0.15F_{z_{max}} - 10d_{x_{max}} - 100d_{y_{max}} - 10d_{z_{max}} - 100d_{up_{z_{max}}} - 0.2t - 50000\Delta_{demo}$$

4.2.3 Controller Combination Selection: Approach

For the higher level parameter selection problem, we use a standard bandits algorithm, Upper Confidence Bound (UCB) sampling, to determine the x-y-z controller combination with the highest expected reward for each of the three cutting skills (normal cut, pivoted chop, and scoring). This problem is fairly trivial, since there are only eight possible discrete actions (x-y-z controller combinations). However, we perform this experiment for completeness and to see if, through RL, we can in a few samples eliminate some x-y-z controller combinations that are obviously not suitable for these cutting tasks. We select one hard food and one soft food and run an experiment with 20 samples for each of the three cut types to determine the best combination of x-y-z axes controllers for each type of cutting skill. We then generalize these selected controller combinations to the four other food types in the lower level parameter learning experiments discussed in the policy optimization sections in this chapter.

Initial Experiments to Explore Controller Selection Problem

Once we defined a reward function, we evaluated it on three shortened normal cut experiments, each with a different set of controller combinations: one with x-axis position control and z-axis position control with variable pitch stiffness, another with x-axis position control and z-axis force control with variable pitch stiffness, and a third with x-axis position control and z-axis force control with constant pitch stiffness (i.e. pitch stiffness was not a learned policy parameter in this experiment). The main goals of these experiments were to test out the reward function defined above and to evaluate if there are performance differences between using position vs. force control in the z-axis for this task.

We conclude two main takeaways from these initial experiments: first that the analytical reward function we defined does indeed allow for policy optimization of the cutting skill. Second, there is a significant difference in the rewards achieved between using force control vs. position control in the z axis for this cutting skill, indicating that force control in the z axis is the preferred choice.

Rather than running lower level parameter learning experiments to test all the different controller combinations, we separate the problems of selecting the high level controllers and tuning the lower level policy parameters. Using a multi-armed bandit algorithm, we seek to efficiently learn the best controller combinations for each of the three cutting skills.

4.2.4 Controller Combination Selection: Experimental Results

Figure 4.8 shows results from the high level action selection experiment to determine the best x-y-z axis controller combination for each of the three cutting skills (normal cut, pivoted chop, and scoring) using Upper Confidence Bound (UCB) sampling. Mean expected reward is shown on the y axis, and the x axis shows the 8 different axis controller combinations, with 0 indicating force control and 1 indicating position control. Solid lines indicate results for the hard food (potato) used in these experiments, and dashed lines indicate results for the soft food used in these experiments (tomato). The actions with the highest mean expected reward for each cut and food type are shown in Table 4.1. We note that there is very little difference between the expected rewards between some of the actions, indicating that more than one controller combination could be selected and yield very similar results. More specifically, if there is no movement in a given axis (e.g. y-axis, or x-axis in the pivoted chop skill), choosing a zero-force vs. a zero-position constraint for this axis does not seems to have a noticeable impact. However, there are some controller combinations which are obviously not preferable for the skills so we can eliminate these.

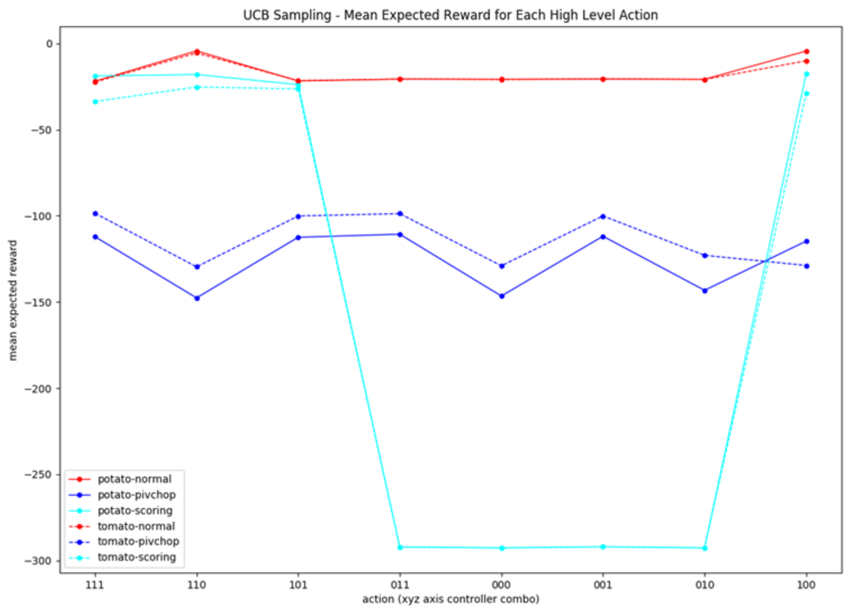


Figure 4.8: UCB experiment results showing mean expected reward for each of the 8 different xyz axis controller combinations for three different cut types (normal cut, pivoted chop, and scoring) and two different food types (potato and tomato).

Cutting Skill	**Food Type**	**Best Action**
Normal Cut	Hard	110 [Position, Position, Force]
	Soft	110 [Position, Position, Force]
Pivoted Chop	Hard	011 [Force, Position, Position]
	Soft	111 [Position, Position, Position]
Scoring	Hard	100 [Position, Force, Force]
	Soft	110 [Position, Position, Force]

Table 4.1: High level action selection experimental results calculated from 20 samples for each of the six experiments.

4.2.5 Policy Optimization with Analytical Reward: Approach

To fine-tune the continuous low level skill parameters, namely the DMP position trajectory parameters, force targets, and pitch axis stiffnesses, we use the policy search algorithm Relative Entropy Policy Search (REPS) to optimize the initialized policies, which we treat as multivariate Gaussian policies parameterized with a mean and covariance.

REPS is a policy search algorithm which places a bound on the information loss between policy updates (ϵ), in the form of a KL divergence bound between the current policy and the updated policy [28].

$$\sum_{s,a} \mu^{\pi}(s)\pi(a|s) \log \frac{\mu^{\pi}(s)\pi(a|s)}{q(s,a)} \leq \epsilon,$$

where $q(s,a)$ is the current sample distribution and $\mu^{\pi}(s)\pi(a|s)$ is the updated distribution.

Because of this, REPS has been shown to be effective for real robot experiments by minimizing policy updates that stray too far from the initial policy, as shown in [8, 28]. It also has been shown to work well for low dimensional policies, which ours are (our Gaussian policies are 9-10 parameters, depending on the cutting skill). Although a potential downfall of REPS is that it can be prone to getting stuck in local optima, we are less concerned about this for our particular application because we assume decently good policy initialization (from human demonstrations) and are just looking to fine-tune these policies rather than learn them completely from scratch.

As mentioned earlier, we initialize the DMP trajectory policy parameters from human demonstrations. The z-axis force parameter is initialized roughly from prior cutting force data, although there is very high variability in the force data across different food types. We therefore ultimately select an initial z-axis force value of -10 N and use RL to fine-tune this parameter for different food types. The pitch stiffness values are initialized as either high (200 N/m) or low (20 N/m), depending on the cutting skill. For skills that show non-trivial pitch axis rotational movement during the human demonstrations, we initialize with a low stiffness (pivoted chop skill), and for skills that show no pitch axis rotation during the demonstrations, we initialize with a high stiffness (normal cut and scoring skills). The specific high and low initial values are selected using prior experimental data. The exploration variances for each of the policy parameters are treated as hyperparameters that we select to tune the amount of exploration the agent undertakes during learning. For these experiments, we use an exploration variance of 0.01 for the DMP weight parameters, 120 for the force parameter, and 800 for the pitch stiffness parameter. Note that when the agent is sampling from the policy mean during the robot experiments, we clip the force and stiffness parameters to avoid safety risks due to the robot holding a knife. Z-axis force is clipped between [-40N, -3N] and pitch stiffness is clipped between [5 N/m, 600 N/m]. Finally, we run each experiment for five epochs with a total of 115 samples in each experiment.

Lastly, we formulate the reinforcement learning problem as a Multi-Armed Bandits problem, rather than a sequential decision making problem, to simplify the overall learning problem and because we feel that selecting a set of DMP trajectory parameters, force targets, and pitch stiffness for an entire cutting skill execution is sufficient to achieve cut success for the tasks we are focused on.

4.2.6 Policy Optimization with Analytical Reward: Experimental Results

Figure 4.9 shows the distribution of z-axis force and pitch stiffness sampled parameters from the normal cut (first column) and scoring (second column) lower level parameter RL experiments across all six food types, and just the pitched stiffness sampled parameters from the pivoted chop experiments (third column) because the pivoted chop skill uses position rather than force control in the z-axis. Figure 4.10 shows the mean and standard deviations of the peak x forces (top row) and peak z forces (bottom row) for each of the 18 lower-level parameter learning experiments (across the three cutting skills and six food types). Note that these are peak forces across the entire cut execution, so some of the very high z forces are due to the knife being in contact with and dragging along the cutting board surface.

From the z-axis force data, we can see that the generalized analytical reward function used in these experiments seems to be favoring lower force policies, due to a high penalty applied to high forces. Also, the softer foods generally converge to policies with lower z-axis forces than the harder foods, although this is not true in all cases.

Figure 4.11 visualizes some of the DMP x and z position trajectories sampled during the RL experiments. The starting DMP policies initialized from human demonstrations are shown in blue, the sampled trajectories are shown in red, and the final policy mean trajectories after five epochs are shown in green. Foods shown in this figure are banana (first column), carrot (second column) and tomato (third column).

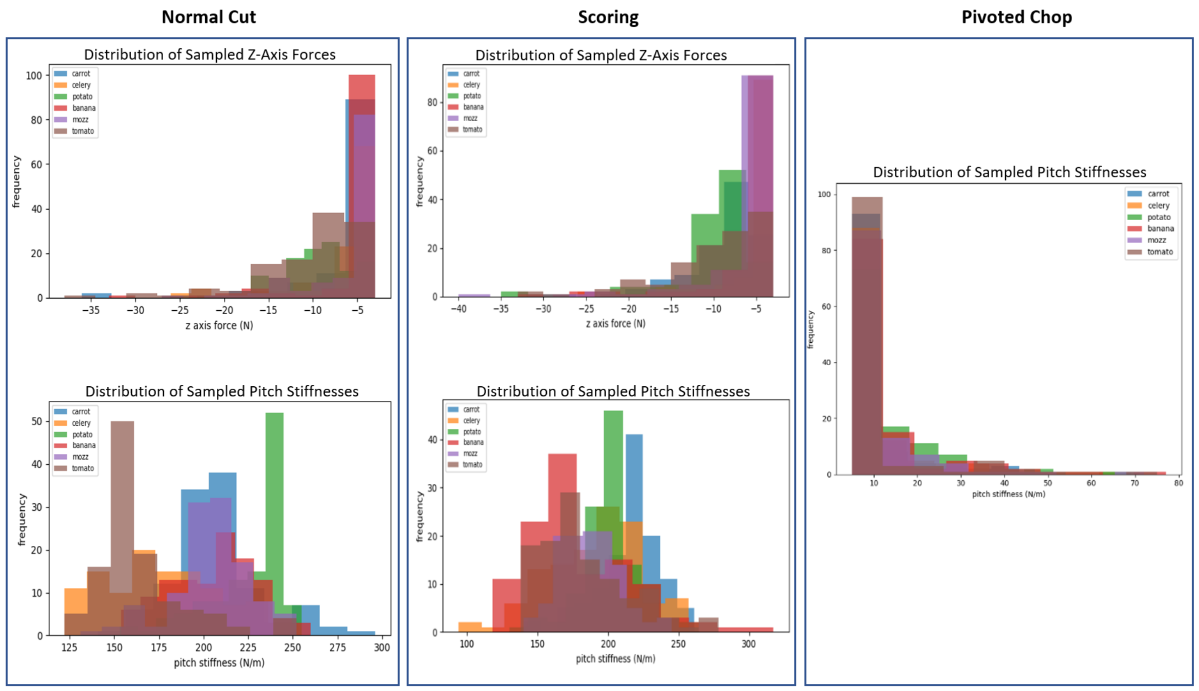


Figure 4.9: Sampled z-axis force and pitch stiffness parameters during lower level policy learning experiments - columns one, two, and three show normal cut, scoring, and pivoted chop experiment data, respectively.

Figure 4.12 shows the average rewards vs. epochs for the 18 lower level parameter learning experiments (across the three cut types and six food types) using the same generalized analytical reward function for all experiments. Figure 4.13 shows average task success (0,1,2) in the

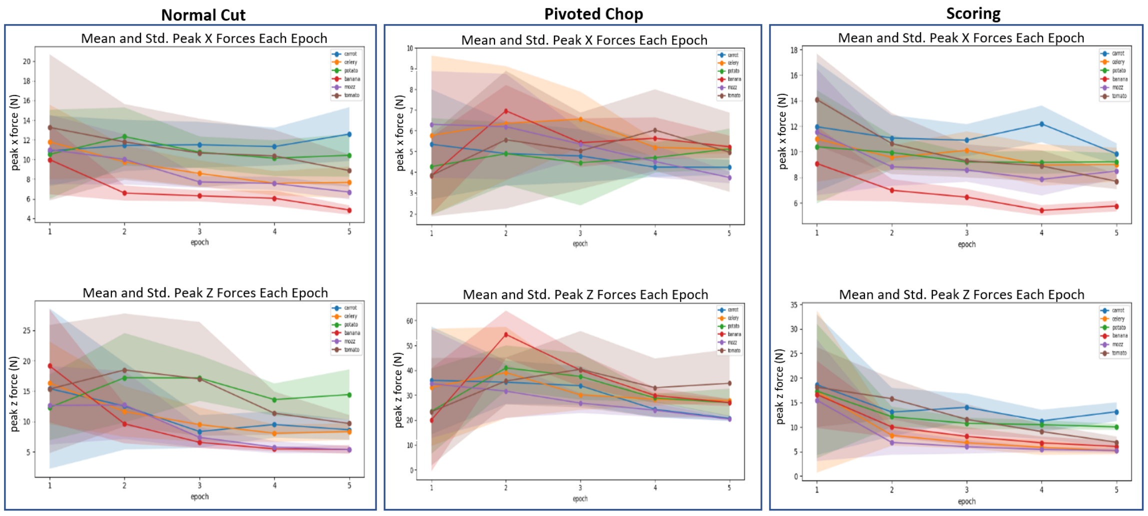


Figure 4.10: Mean and standard deviation of peak x forces (top row) and z forces (bottom row) each epoch - columns one, two, and three show normal cut, pivoted chop, and scoring experiment data, respectively.

top row, and percent successful cuts (where both 1 and 2 are considered successful because they both indicate a fully cut slice) vs. epochs in the bottom row. With the exception of the tomato scoring experiment, the robot is able to achieve $\geq 95\%$ task success in each of the 18 experiments, indicating successful fine-tuning (from a task success standpoint) of the lower level policy parameters using the generalized analytical reward function. These task success results are summarized in Table 4.2.

Table 4.3 shows the final policy means (after the 5th epoch) for the z-axis force and pitch stiffness policy parameters for each of the cutting skills and food types. For mozzarella and banana, which are the softest foods in the set, the robot learns a very low z-axis force parameter since this is sufficient to cut through these foods. For the potato, on the other hand, it learns a higher z-axis force policy which also makes sense given the hardness of a potato and higher forces required to cut through it. For pitch stiffness, the robot learns a high pitch stiffness for the normal cut and scoring skills and a low stiffness for the pivoted chop skill, which requires rotation about the pitch axis.

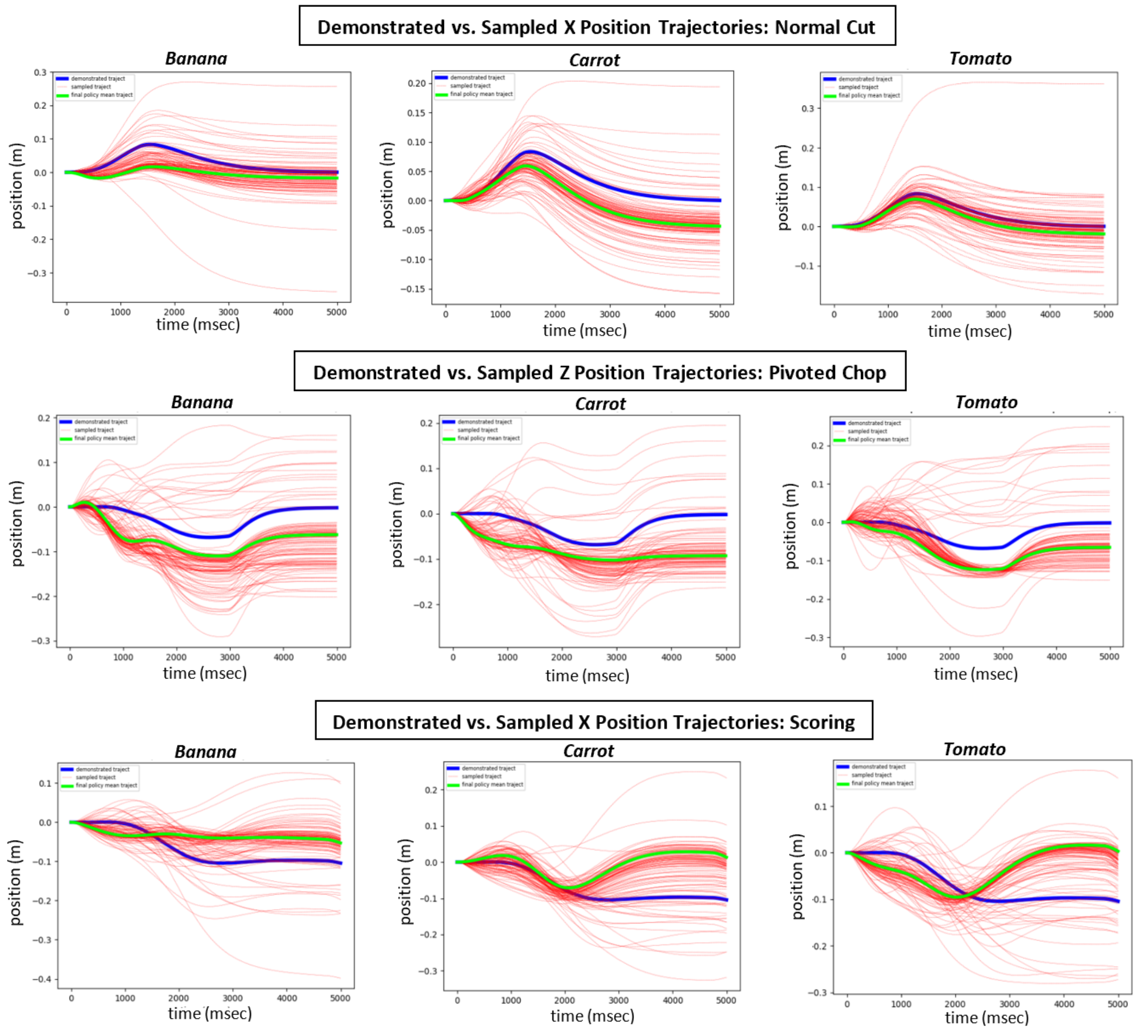


Figure 4.11: Sampled DMP position trajectories for three different food types during lower level parameter learning normal cut (first row), pivoted chop (second row), and scoring experiments (third row). Starting DMP policies initialized from human demonstrations shown in blue, sampled trajectories shown in red, and final policy mean trajectories after five epochs shown in green.

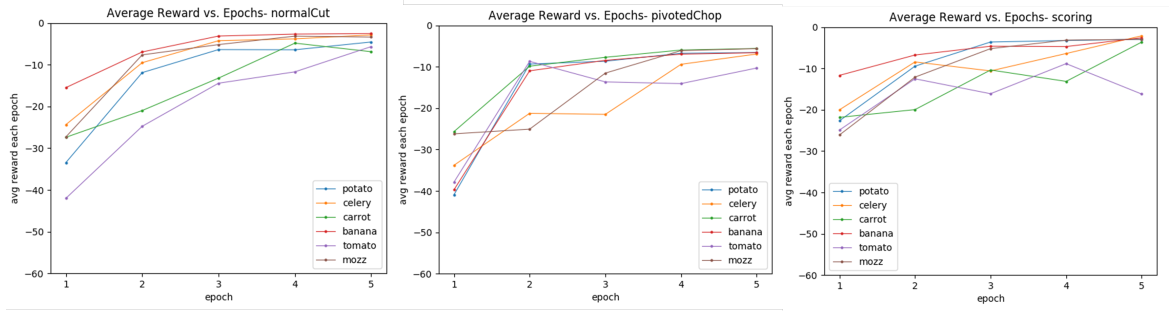


Figure 4.12: Average rewards vs. epochs for 18 lower level parameter learning experiments.

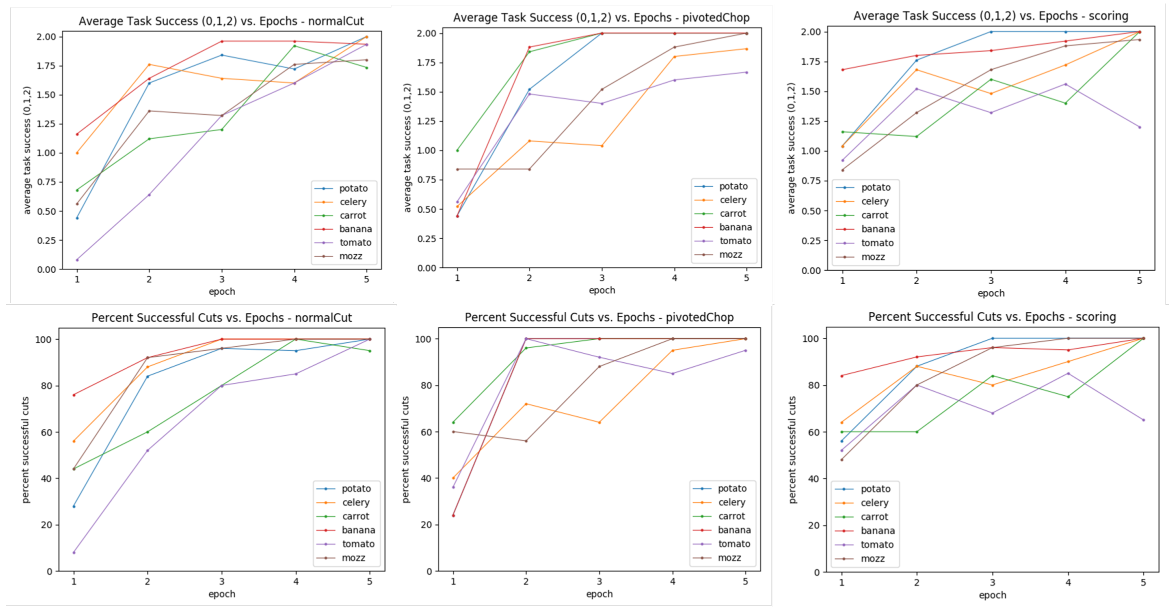


Figure 4.13: Task success for 18 lower level parameter learning experiments - first row shows average task success (0,1,2) each epoch, and second row shows percent task success. Normal cut, pivoted chop, and scoring results shown in 1st, 2nd, and 3rd columns respectively.

Food Type	Normal Cut % Task Success Final Epoch	Pivoted Chop % Task Success Final Epoch	Scoring % Task Success Final Epoch
Carrot	95	100	100
Celery	100	100	100
Potato	100	100	100
Banana	100	100	100
Mozzarella	100	100	100
Tomato	100	95	65

Table 4.2: Summarized final epoch task success results for lower level parameter learning robot cutting experiments.

	Normal Cut		Pivoted Chop	Scoring	
Food Type	**Z-Axis Force (N)**	**Pitch Stiffness (N/m)**	**Pitch Stiffness (N/m)**	**Z-Axis Force (N)**	**Pitch Stiffness (N/m)**
Carrot	-4.42	203.41	5.00	-6.04	234.89
Celery	-4.90	141.82	5.04	-3.00	166.33
Potato	-9.37	240.29	5.35	-8.16	211.00
Banana	-3.00	212.70	10.19	-3.00	159.81
Mozzarella	-3.00	212.26	5.00	-3.22	198.70
Tomato	-6.89	150.08	5.00	-4.55	151.81

Table 4.3: Final policy mean z-axis force and pitch stiffness parameters (after 5th epoch) for generalized analytical reward experiments.

4.2.7 Policy Optimization with Human-in-the-Loop Reward Learning: Approach

The next question with respect to defining rewards for learning robotic food cutting skills is how to handle outlier foods that do not achieve good performance with the generalized analytical reward function defined above. Outliers are often the more deformable foods that behave most differently from rigid body objects. A human cutting these types of foods would modify their behavior with subtle nuances to successfully cut through (for example, by moving the knife slightly forward while applying a gentle downward pressure when cutting a tomato to avoid squishing it). Since it is difficult to hand-code rewards that capture these nuances without extensive parameter tuning and feature engineering, we next consider how a human observing the robot's cuts can specify rewards in an intuitive way that will enable the robot to successfully learn policies for the more difficult to cut food items. In this section, we evaluate an approach to allowing a human to provide input to the robot to guide policy learning based on the human's preferences.

Overview of the Approach

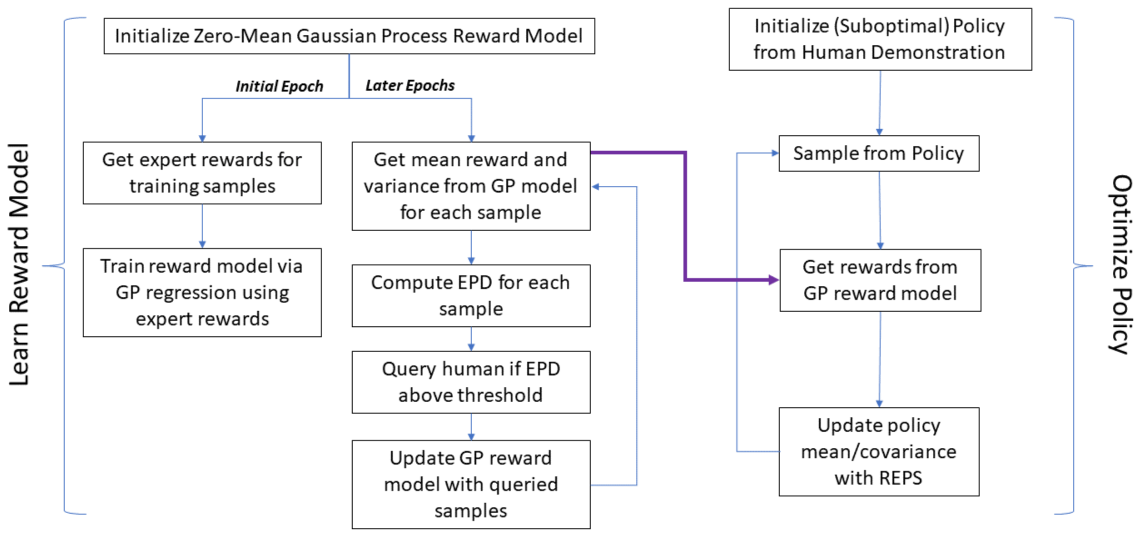

Learn Reward Model
Initialize Zero-Mean Gaussian Process Reward Model
Initial Epoch
Later Epochs
Get expert rewards for training samples
Train reward model via GP regression using expert rewards
Get mean reward and variance from GP model for each sample
Compute EPD for each sample
Query human if EPD above threshold
Update GP reward model with queried samples
Initialize (Suboptimal) Policy from Human Demonstration
Sample from Policy
Get rewards from GP reward model
Update policy mean/covariance with REPS
Optimize Policy

Figure 4.14: High level overview of our human-in-the-loop reward and policy learning approach based on work proposed by Daniel et al [9].

Using the active reward learning framework proposed by Daniel et al [9], our goal is to have the robot simultaneously learn a policy and a reward model from continuous human-specified rewards. We model the human's rewards using a Gaussian process (GP), with reward

$$R(o) = GP(\mu(o), \Sigma(o, o')),$$

where $\mu(o)$ is the mean, $\Sigma(o, o')$ is the covariance kernel function for the GP, and o is a sample outcome consisting of a $1 \times n$ feature vector from the sample [9]. In our case, the feature vector

o consists of the reward features from the analytical reward function: forces, displacement, and time to cut. We initialize the GP with a zero-mean prior, and we define the covariance kernel as the radial basis function (RBF) kernel with an amplitude hyperparameter θ_0^2. This is also known as the standard squared exponential kernel, defined by the function:

$$k(o, o') = \theta_0^2 \exp(-\frac{||o - o'||^2}{2\theta_1^2})$$

The hyperparameters θ_0^2 and θ_1^2 represent the signal variance and lengthscale of the kernel, and are optimized by maximizing the log likelihood of the model. We use automatic relevance determination (ARD) for the lengthscale hyperparameter, allowing each dimension of the feature vector to have a different lengthscale since some reward features may be more relevant than others.

Compared to the previous RL approach with the generalized analytical reward function using hand-selected scaling weights, the goal of this approach is to learn better scaling weights for the reward function features using human input to guide policy learning.

Reward Feature Scaling

For the cutting task, the reward features mentioned above vary quite a bit in scale (x-y-z displacements vs. forces vs. time are all different units), so we standardize the reward features using the mean and standard deviation of each dimension of the feature vectors from training data collected during prior experiments. We compute a mean and standard deviation for each dimension for each cut type using 70-100 training samples collected previously, and then at test time we standardize the features for each sample using these pre-computed mean and standard deviation before inputting them to the GP reward model for prediction.

Determining When to Query Human for Rewards Using Expected Policy Divergence (EPD)

We use an acquisition function proposed by Daniel et al called the expected policy divergence (EPD), as a metric of determining which samples are most informative to the policy, and only query the human for rewards for these samples [9]. Unlike traditional Bayesian optimization acquisition functions that seek to maximize a blackbox function (e.g. expected improvement, probability of improvement, upper confidence bound, etc.), the EPD takes into consideration the fact that that the reward model itself is not necessarily of primary interest to the agent, but rather its effect on the policy learned by the robot. Daniel et al showed that using the EPD metric performed better than other traditional acquisition functions in the context of human-in-the-loop RL, presumably because it allow for more directed human reward queries that are informative to the policy learning [9]. To compute the EPD, we quantify the KL divergence between the policy under the current reward model and the policy under the reward model if we were to update it for a given sample, and use this to determine whether or not to query the human for a reward for a given sample using a query threshold hyperparameter, which we refer to as κ.

$$EPD(o) = \mathbb{E}_{p(R|o,D)}[KL(\pi^*(w|D^*)||\widetilde{\pi}(w))]$$

Since we do not assume access to the expert rewards when computing the EPD to actually compute the updated reward model for a given sample, we use a sigma point approach to approximate the updated reward model for a given sample using $[\mu(o) + \sigma(o), \mu(o) - \sigma(o)]$ and compute the update reward model for each of these points as proposed by [9]. We define the policy under the current reward model as $\widetilde{\pi}$, and the policy under the updated reward model for a given sample as π^*, and compute the Kullback-Leibler divergence (KLD) between these two policies. As mentioned above, the query threshold κ is a hyperparameter we select to control the trade-off between a more accurate reward model and minimizing human queries. The agent queries the human if $EPD(o) \geq \kappa$. To compute this, we approximate the KLD between π^* and $\widetilde{\pi}$ in weight space to avoid numerical instabilities that sometimes occur with the analytical KLD calculation [9], where the weights for each sample (γ_i^* and $\widetilde{\gamma}_i$) are calculated based on the rewards under the updated reward model (R^*) and current reward model (R), respectively, using relative entropy optimization [28].

$$\gamma_i = \exp(\frac{R_i}{\eta}),$$

where η is the temperature parameter and is found by optimizing the following objective function:

$$g(\eta) = \eta\epsilon + \eta \log \frac{1}{N} \sum \exp(\frac{R}{\eta}),$$

where ϵ is the relative entropy bound and is set to 1.5 in all experiments discussed here.

$$\eta^* = \underset{\eta}{\operatorname{argmin}}\, \eta\epsilon + \eta \log \frac{1}{N} \sum \exp(\frac{R}{\eta})$$

Using the weights for samples under R and R^*, the approximated KLD calculation becomes, per [9]:

$$KL(\pi^*(w|D^*)||\widetilde{\pi}(w) \approx \sum_{i=1}^{N} \gamma_i^* \log \frac{\gamma_i^*}{\widetilde{\gamma}_i}$$

This approach allows us to minimize expensive reward queries to the human expert while still learning a policy that can complete the task. We also explore the trade-off between improving the reward model and minimizing human reward queries and the impact on policy learning in the results presented in Section 4.2.8.

Batch vs. Iterative Reward Model Updates

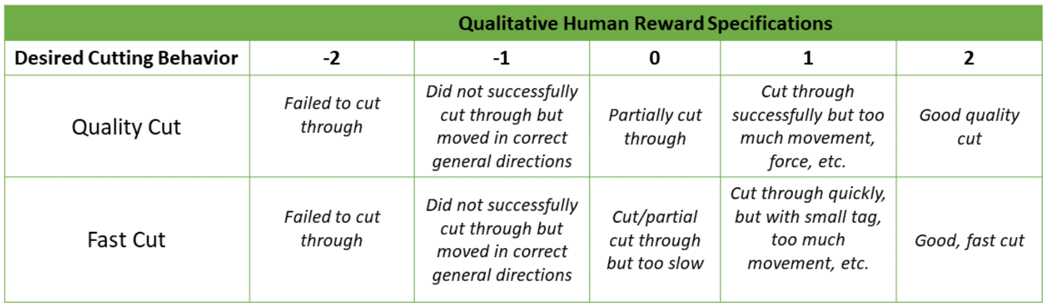
We also look at two approaches to updating the GP reward model - first, a batch approach in which we compute the EPD for all the samples in a given epoch, query the human for all the samples above the query threshold κ, and update the reward model with the queried samples. In the iterative approach, we compute the EPD for all the samples in a given epoch, select the one with the highest KLD, query this sample if it is greater than κ, and update the GP model with this sample. We continue iteratively updating the GP model one sample at a time in this way until the sample with the highest KLD value is below the query threshold, at which point we stop querying and continue to the policy update step. For the batch approach, we run the GP hyperparameter optimization during initial GP model training, and in any later update that involves greater than 10 additional queried samples. With the iterative updates, since we are updating the GP reward model one sample at a time, we do not run the hyperparameter optimization each time and instead initialize the kernel hyperparameters using prior experimental data. We try both batch and iterative approaches, but use the iterative approach for most experiments because it tends to avoid redundant queries to the human for samples that are all uncertain but in the same part of the input feature space.

4.2.8 Policy Optimization with Human-in-the-Loop Reward Learning: Experimental Results

We next apply the HIL approach to real robot cutting experiments. We focus on the tomato scoring task, which did not achieve good task success in the generalized analytical reward function experiments (results shown in Section 4.2.6). The goal of this part of the study is to explore if the robot can use the human's input to tailor the learned cutting behavior based on the human's preferences. Essentially, how well can the robot converge to a cutting behavior that the human likes? This is particularly important for food cutting skills and more broadly cooking skills, since these are often subjective and depend on the preference of the human. Unlike with a task such as door opening where there is an objective metric to evaluate the robot's execution, cooking tasks are quite different. Everyone cuts, plates, flips, and cooks food differently and with their own style. Since different people may prefer different behaviors over others, there is not a clear "correct" or "optimal" answer for many of these skills. For example, in the context of cutting, some people may prefer faster cutting motions while others may be more careful with the knife and apply more gentle forces to generate slower, more gradual cuts.

For the HIL experiments we conduct in this thesis, a human observes each of the robot's cutting policy executions and provides continuous rewards between -2 to 2, with qualitative reward definitions shown in Figure 4.15. We focus on two desired cutting behaviors: quality cut and fast cut.

For these human-in-the-loop (HIL) experiments, we first focus on the tomato scoring skill, which did not achieve good task success in the RL experiments using the generalized analytical



	Qualitative Human Reward Specifications				
Desired Cutting Behavior	**-2**	**-1**	**0**	**1**	**2**
Quality Cut	*Failed to cut through*	*Did not successfully cut through but moved in correct general directions*	*Partially cut through*	*Cut through successfully but too much movement, force, etc.*	*Good quality cut*
Fast Cut	*Failed to cut through*	*Did not successfully cut through but moved in correct general directions*	*Cut/partial cut through but too slow*	*Cut through quickly, but with small tag, too much movement, etc.*	*Good, fast cut*

Figure 4.15: Qualitative human reward specifications for continuous rewards specified between -2 to 2 for two types of desired cutting behaviors: Quality cut and Fast cut.

reward discussed in Section 4.2.6. We then perform an auxiliary experiment with the normal cut skill on a potato to evaluate if we can tune the cutting behavior to be faster based on the human's input (this experiment is referred to as "Potato Fast Cut").

Figure 4.16 shows an example of the GP model input train and test distributions (in blue and yellow, respectively) for each dimension of the standardized reward feature vectors during one of the tomato scoring HIL experiments. We can see that the test distribution falls within the train distribution, indicating that at test time the GP is making reward predictions on inputs that are within its training distribution.

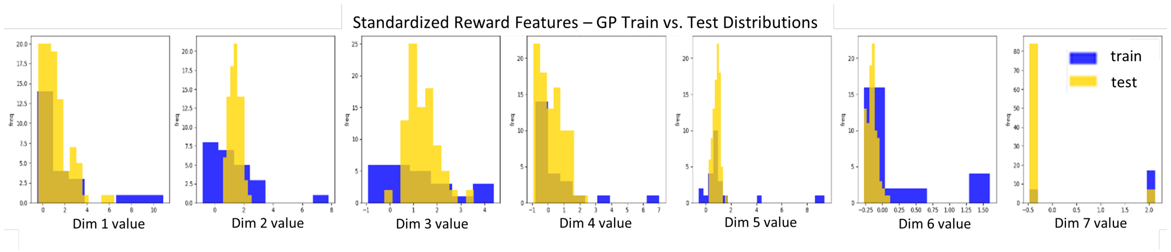


Figure 4.16: GP model train and test distributions for each dimension of the standardized reward feature vector for tomato scoring HIL experiment.

Batch vs. Iterative Reward Model Update Evaluations

To evaluate the batch vs. iterative GP model update approach discussed in Section 4.2.7, we collect a small set of data (25 samples) and perform analysis looking at various ways of training the GP model. Figure 4.17 shows GP model rewards vs. human rewards (top row) and variance (bottom row) for the batch model update approach with different training data sizes: s = 5, 10, 15, 20, 25 samples. Vertical red lines in the bottom row plots indicate which additional samples out of the 25 total samples would be queried based on the EPD calculation with a query threshold of 0.01.

Next, Figure 4.18 shows the GP model rewards and variances pre- and post-reward model update using the samples queried shown by the red dashed lines in Figure 4.17, for each of the different training set sizes.

Finally, Figure 4.19 shows results from applying the iterative approach to the same dataset, with GP model vs. human rewards shown in the top row, and GP model variances shown in the bottom. With the iterative approach, we train the GP model with only the first sample, and then iteratively compute the EPD for the remaining samples and query the sample with the highest EPD if it is above the query threshold. Results shown are for 1 to 15 queries, i.e. for a GP model training size of s = 1 to 15.

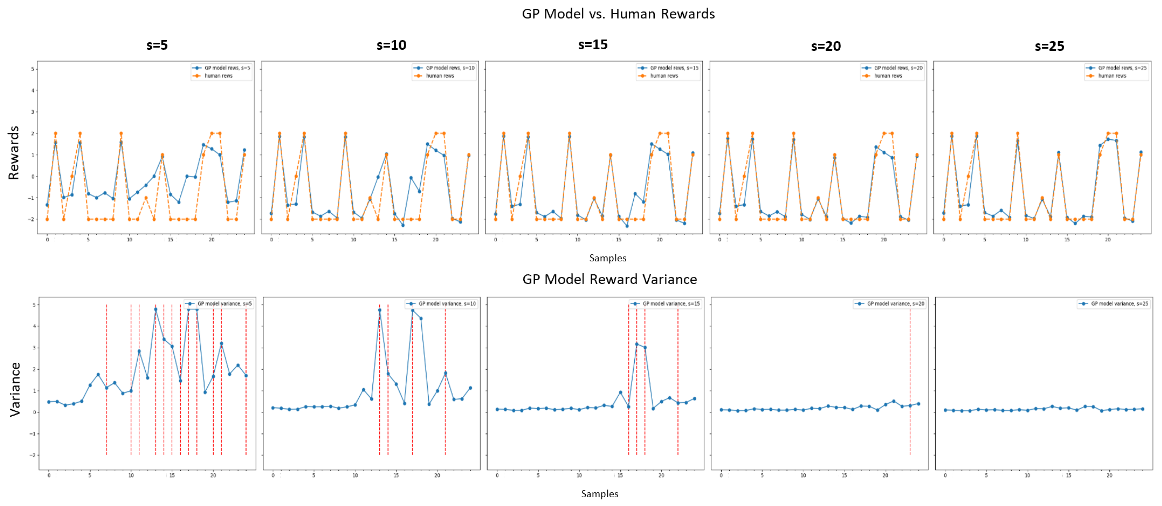


Figure 4.17: GP model rewards vs. human rewards (top row) and variance (bottom row) for batch model update approach with different training data sizes: s = 5, 10, 15, 20, 25 samples. Vertical red lines in bottom row plots indicate which additional samples out of the 25 total samples would be queried based on the EPD calculation with a query threshold of 0.01.

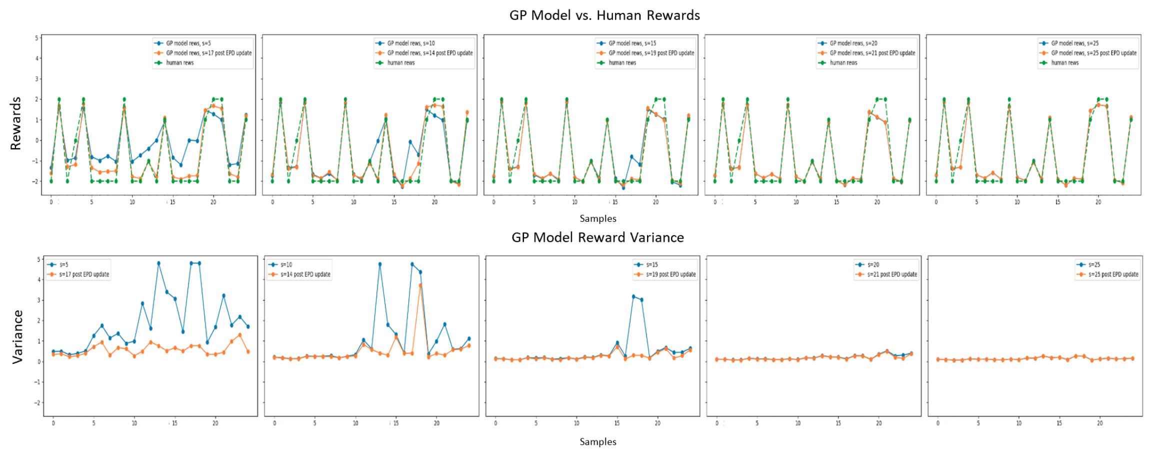


Figure 4.18: GP model rewards vs. human rewards (top row) and variance (bottom row) shown pre- and post- reward model update based on queried samples shown in Figure 4.17.

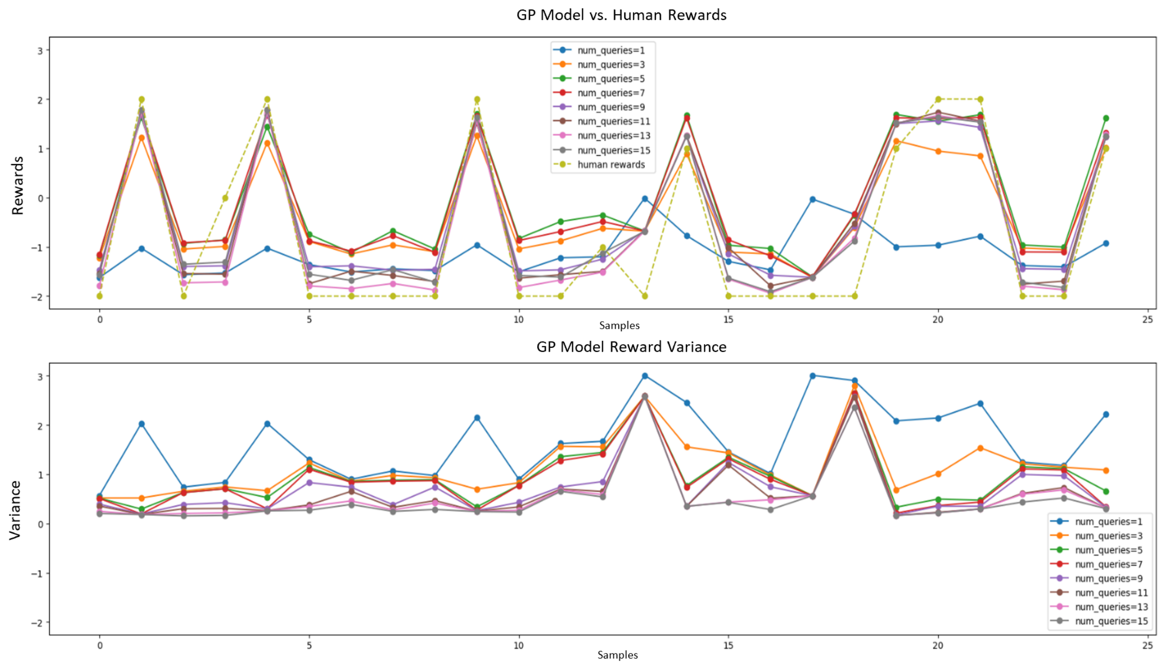


Figure 4.19: Results from applying the iterative approach to the same dataset as shown in 4.17. GP model vs. human rewards shown in the top row, and GP model variances shown in the bottom row, for GP training data sizes from 1 to 15.

We conclude from these results that using the iterative approach is preferable in this case to limit the number of redundant human queries (i.e. samples in similar regions of the input feature space). Also the iterative approach helps to ensure that the initial samples used to train the GP model are informative, rather than just randomly selecting initial training samples as is done in the batch update approach.

GP Model Training Data Gram Matrices

Figure 4.20 shows a visualization of the Gram matrices (i.e. the GP covariance kernel matrix) of the GP model training data samples for four different HIL experiments: Tomato scoring batch update experiment (top left), Tomato scoring iterative update experiment 1 (top right), Tomato scoring iterative update experiment 2 (bottom left), potato fast cut iterative update experiment (bottom right). Yellow color indicates strongly correlated samples, while darker blue indicates uncorrelated samples. We observe that many of the later samples in the tomato scoring batch update experiment (top left) are correlated, potentially indicating redundant samples being queried from the human by the agent. As mentioned above, we run the rest of the experiments discussed in this section using the iterative GP model update approach to try to decrease the amount of redundant queries to the human.

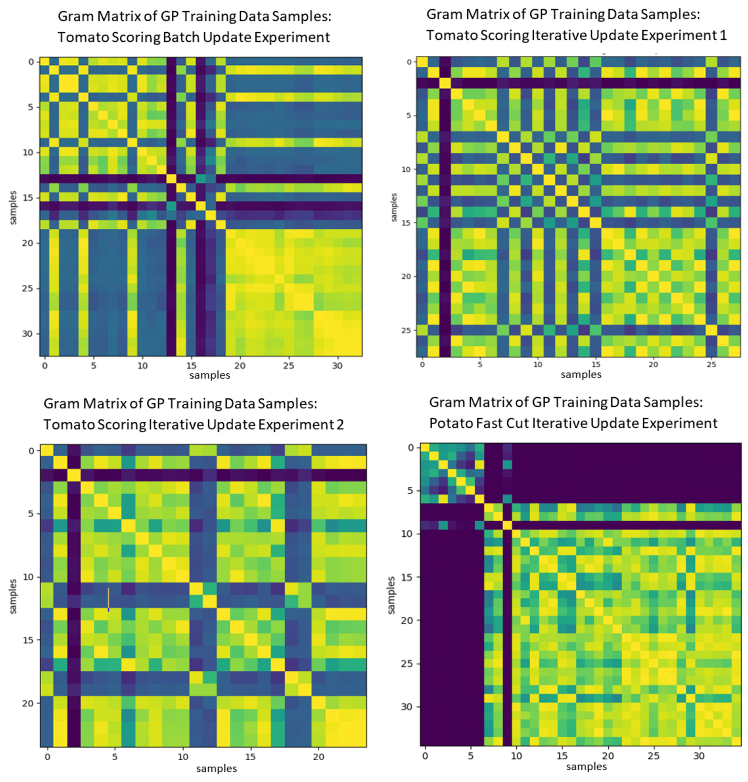


Figure 4.20: Gram matrices of GP model training data samples for 4 different HIL experiments: tomato scoring batch update (top left), tomato scoring iterative update experiment 1 (top right), tomato scoring iterative update experiment 2 (bottom left), potato fast cut iterative update (bottom right). Yellow color indicates strong correlation, while darker blue indicates uncorrelated.

Tomato Scoring Iterative Human-in-the-Loop Experiment Results

We run two experiments in this section with different query thresholds, kappa (κ). $\kappa = 0.1$ in Experiment 1 and $\kappa = 0.3$ in Experiment 2. Figure 4.21 shows both GP model rewards and human rewards for the two HIL tomato scoring experiments (left column), average rewards vs. epochs for both experiments (top right), and cumulative human queries vs. epoch for both experiments. With $\kappa = 0.1$, the agent queried the human 28 times out of the 115 total samples, and with $\kappa = 0.3$, the agent queries the human 24 times.

Figure 4.22 shows a comparison of task success metrics between the two HIL RL tomato

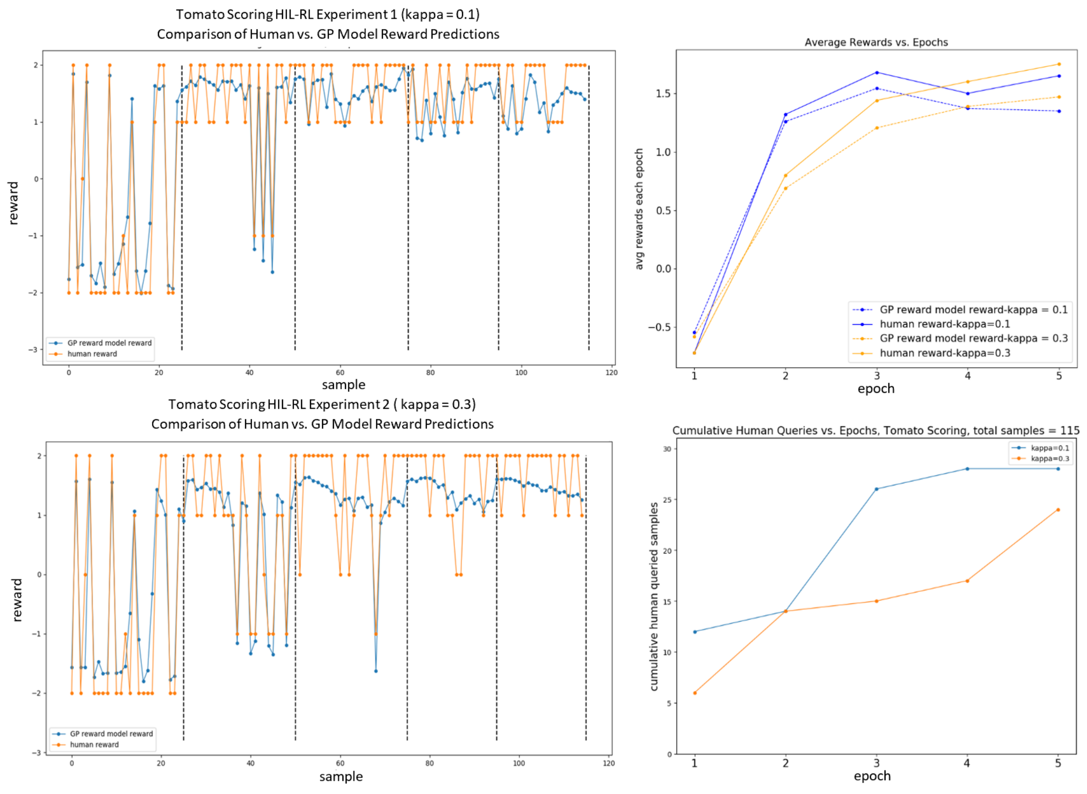


Figure 4.21: Tomato scoring iterative HIL experiment results - GP model predictions vs. human rewards (left column), average rewards vs. epochs (top right), cumulative human queries (bottom right).

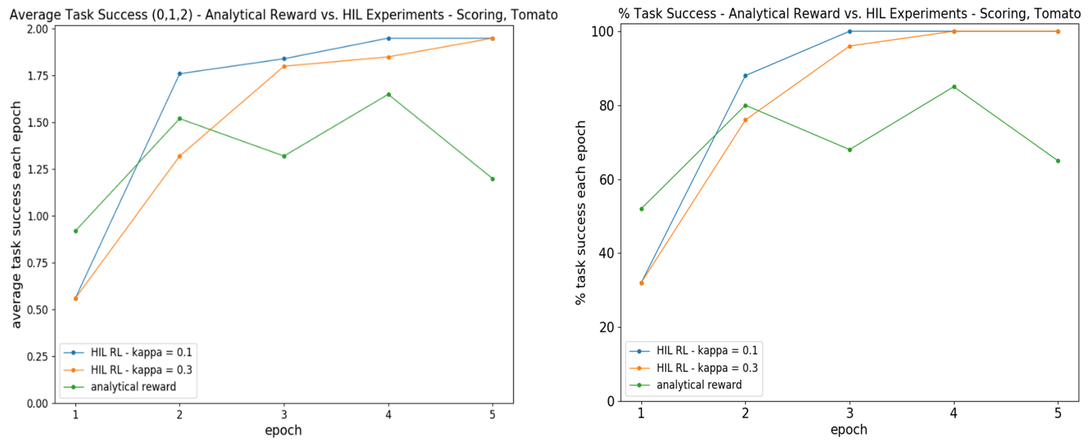


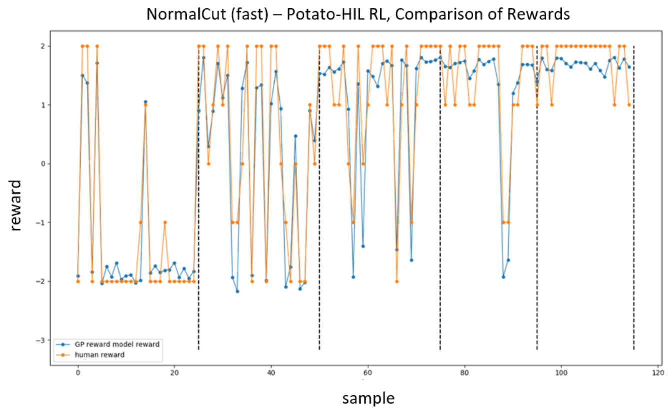
Figure 4.22: Comparison of tomato scoring task success between analytical reward approach vs. human-in-the-loop approach.

scoring experiments and the generalized analytical reward experiment discussed in Section 4.2.6. Both experiments with the human-in-the-loop approach achieved 100% task success, compared to the task success of 65% achieved with the generalized analytical reward experiment, indicating a significant improvement in the learned policy using the HIL approach.

Potato Fast Cut Iterative Human-in-the-Loop Experiment Results

We next perform an experiment to try to tune the normal-cut cutting behavior to be faster on a potato, where the human is specifying rewards based on a desired fast cutting behavior, shown in Figure 4.15. Figure 4.23 shows both GP model rewards and human rewards for the potato fast cut experiment (left) and average rewards vs. epochs (right). Figure 4.24 shows cumulative human queries (total of 33 human queries out of 115 total samples), and Figure 4.25 shows average and percent task success for this experiment. Finally, Figure 4.26 shows the distribution of sampled z-axis force parameters for the normal cut-potato experiment performed with the generalized analytical reward function (results shown in Section 4.2.6) compared to this HIL experiment. The distribution of z-axis forces in the HIL experiment is shifted to the left of the analytical reward experiment, indicating higher forces and therefore faster cuts. This is reflected in the final policy mean z-axis force parameter, which converged to a force of -19.96 N in HIL experiment, compared to the force of -9.37 N from the analytical reward experiment. These results indicate that the HIL approach applied here enabled tuning the robot's normal cutting policy to be faster based on human rewards, while still achieving 100% task success.

Table 4.4 shows the summarized data from the three HIL RL experiments discussed in this section.



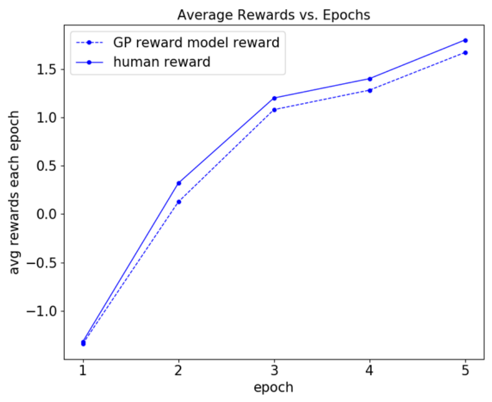


Figure 4.23: Potato fast cut iterative HIL experiment results - GP model predictions vs. human rewards (left), average rewards vs. epochs (right).

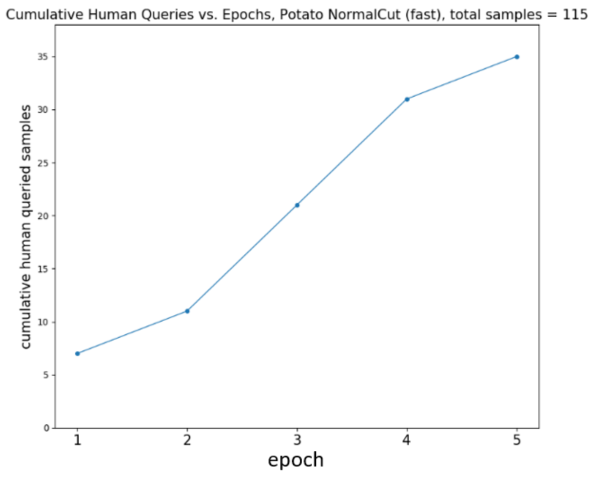


Figure 4.24: Potato fast cut iterative HIL experiment results - cumulative human reward queries.

Experiment	Final Policy Z-Axis Force (N)	Final Policy Pitch Stiffness (N/m)	Human Reward Queries (% of Total Samples)	Task Success Final Epoch (%)
Tomato-Scoring Experiment 1	-17.47	222.28	24.3	100
Tomato-Scoring Experiment 2	-17.30	206.06	20.9	100
Potato-Fast-Cut	-19.96	257.85	28.7	100

Table 4.4: Human-in-the-loop RL experiment results summarized.

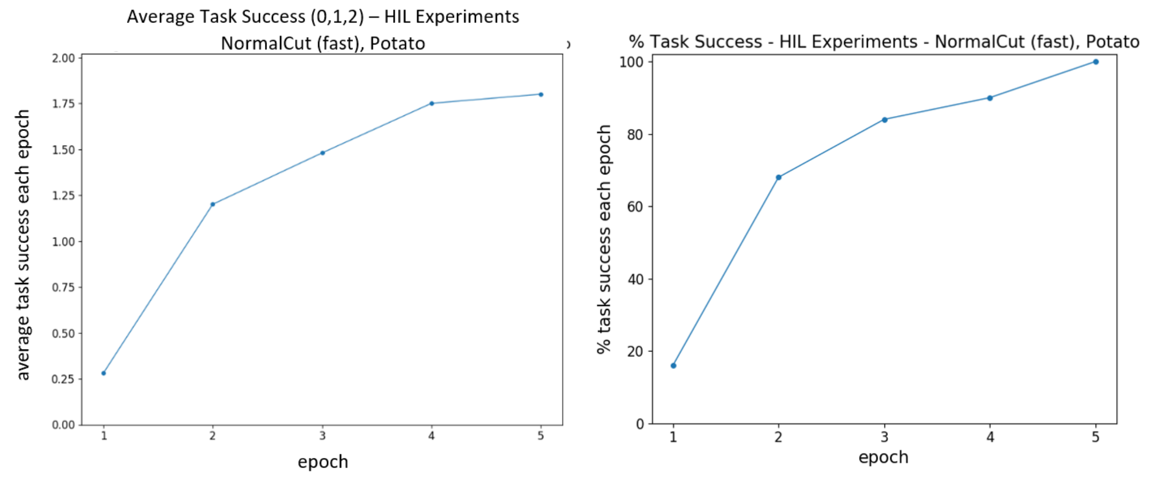


Figure 4.25: Potato fast cut iterative HIL experiment results - task success.

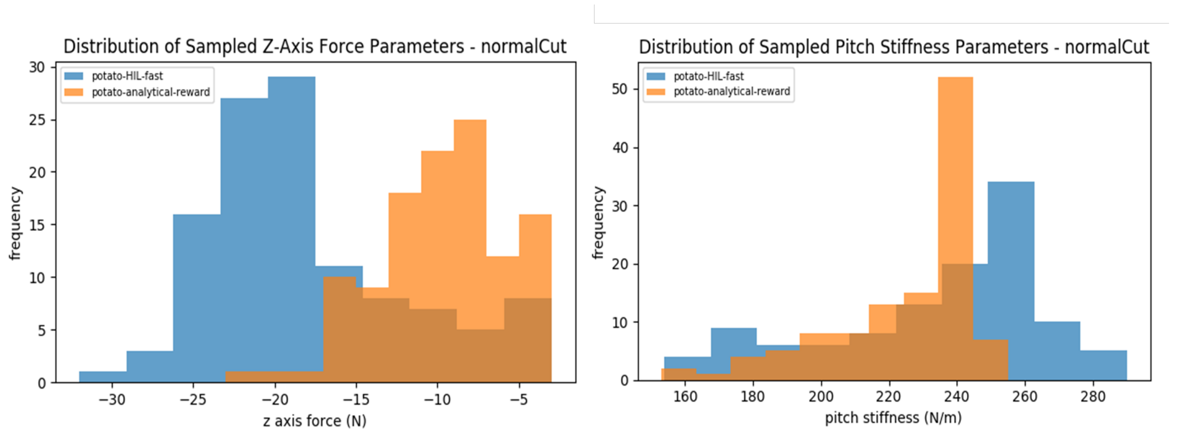


Figure 4.26: Comparison of z-axis forces and stiffnesses in potato normal cut analytical reward experiment vs. human-in-the-loop approach potato fast cut experiment.

4.3 Discussion - Robotic Food Cutting

Learning Cutting Skills from Demonstrations

We show that we are able to teach a range of cutting skills to the robot and parameterizing these skills using DMPs. There is a fair amount of variability in x-y-z axis movements across the eight different cutting skills, and each of the different cutting skills has a different primary axis where position control is needed (e.g. x axis for normal cut and scoring, z axis for dice and pivoted chop, x and z axis for moving pivoted chop). Because of this, we find that combining all the DMP weight data from all eight cutting skills and applying PCA appears to obfuscate the cutting motions that make each of the individual skills unique. Therefore, we conclude that it is difficult to find a single skill parameterization that would generalize across all the eight different behaviors. Additionally, sometimes the trajectories produced by the PCA eigenvectors have behaviors at the beginning or end of the trajectory that cause difficulties when trying to execute them on the real robot, since arbitrary behaviors at the beginning or end of the trajectory often lead to joint velocity or acceleration discontinuities on the Franka. We therefore decide to separate the parameterizations of the different cutting skills so that each skill has its own set of DMP weight parameters. We also note that a set of DMP parameters learned from a cutting skill demonstration on one food material type do not usually work on other food material types, which is why the RL approach is useful to fine-tune and adapt the skill parameters to a range of food material types.

Optimizing Cutting Skills Using Reinforcement Learning

From the high level axis controller combination selection experiment results we see that the more dynamic skills like pivoted chop and dice that require interaction with the cutting board seem to perform better with position control in the z axis, while the normal cut performs better with force control in the z-axis. From the results we can see that with a minimal amount of samples, we can eliminate some x-y-z controller combinations that are obviously not suitable for these cutting tasks. Finally, the controller combinations selected based on the tomato and potato data generalize well to the rest of the food types studied in the lower level parameter learning experiments.

Our takeaway from the the lower level policy fine-tuning experiments using the generalized analytical reward and human-specified rewards results is that the generalized reward function works well to optimize a range of cutting behaviors and food material types (both hard and soft). However, there are some outliers that it does not work as well for, in this case specifically tomatoes. This indicates that the generalized reward function does not fully capture some of the nuances associated with cutting this type of material, and our approach of handling the outlier by incorporating human input to tune the policies based on the human's preferences seems to work well in this case. The policies learned in the tomato scoring experiments with the human-in-the-loop approach showed a 30% improvement in task success compared to analytical reward, while only querying the human for rewards for 20-25$\%$ of samples.

We also note that the tomato had particularly noisy rewards in the analytical reward experiments, where there was some instability in learning and the task success oscillated up and down across epochs. Because the tomato is difficult to grasp, is very slippery, and deformable, the tomato frequently slipped out of the grippers or was easily pushed out of the grippers or destroyed by the knife housing during the experiments, which led to noisier results.

Comparing $\kappa = 0.1$ to $\kappa = 0.3$ in the tomato scoring HIL results, while there is not much difference in the converged policy parameters, $\kappa = 0.1$ converged to 100% task success more quickly (but both achieve 100% success in the final epoch).

Finally, the potato fast cut experiment showed that we were able to use the HIL approach to tune the robot's cutting behavior to be faster using human input.

Overall, the HIL experiments converged to policies with higher z-axis force parameters compared to the analytical reward experiments. From this we can infer that the human observing the executions placed less emphasis on minimizing or penalizing high forces than the analytical reward function did.

Chapter 5

Conclusions

In this thesis we explored two main areas within the realm of robotic food manipulation - first, we looked at how a robot can autonomously and playfully interact with food items and use sensory information collected during these interactions to learn about food item material properties. We then explored how the robot can learn cutting behaviors from human demonstrations and adapt them to a range of food material types through interaction using an RL framework.

Within the first area, we presented an experimental approach to collecting a rich multimodal, interactive, 21-class food dataset consisting of foods with a wide range of material properties. We then showed how the visual, audio, and proprioceptive data collected while the robot interacted with the food slices can be used to learn embedding representations in a self-supervised manner that encode various material properties without the need for explicit human labelling of these material properties. At test time, the trained embeddings networks only require an image as input rather than the full sensory information used during training. We also envision that this dataset, made publicly available, can be leveraged by other researchers in the robotics community to develop new algorithms and innovate in the food robotics space.

On the robot food cutting side, we demonstrated that the robot can acquire a range of cutting skills via human kinesthetic demonstrations, which can be used as a starting point for optimization. The robot can then use an RL approach to fine-tune the cutting policy parameters for a range of hard and soft foods, since foods with different material properties require different policy parameters. Initializing the cutting policies from human demonstrations enabled sample-efficient policy fine-tuning using RL.

We also explored how to define rewards for food cutting and how these different rewards affect the policies learned by the robot. First, we looked at using the same generalized analytical reward function to optimize across three different cutting skills and six different food types. We then evaluated a human-in-the-loop learning approach where the robot learns a model of the rewards using human input while also learning a cutting policy.

Overall, both reward approaches showed positive results from a task success perspective in the robot experiments we conducted, indicating that we can define a generalized reward function that can be used across different cutting skills and food types and optimize the policies to achieve good task success on most food material types. We also observed that tomatoes are a particularly difficult food to cut due to their unique material properties, and it was in this scenario that the human-specified reward approach showed a large improvement in performance compare to the analytical reward.

We observe that the scaling weights for the terms in the reward function greatly impact what type of policy the robot learns, and the HIL approach provides a more intuitive way of specifying rewards without the need to spend of lot of time engineering the reward function. The results from human-in-the-loop reward learning experiments show promising potential to bridge the gap between robotic applications of reinforcement learning in simulation vs. the real world. While tasks in simulation are often easy to specify reward functions for, real world tasks involving deformable objects are often much more difficult to specify reward functions for. This approach enables a “lay person”, i.e. a non-engineer without prior robotics experience, to rate the robot’s executions of a task in an intuitive way, which can enable a robot to learn more complex cooking skills in the real world.

In summary, we hope this research opens new avenues to further explore how home cooking robots can become a reality and how human and robots can cooperatively work in the home kitchen environment.

Chapter 6

Future Work and Open Challenges

There are many interesting future directions and also remaining challenges that will need to be addressed before we can deploy autonomous cooking robots in home kitchens.

On the learning food representations side, an initial area of future work is to use the learned food embeddings for skill planning to select action parameters for food manipulation skills, such as picking and placing, flipping, mixing, and cutting food. These continuous embeddings may enable development of a continuous mapping between food material types and optimal skill parameters, which could help the robot to better adapt and generalize its actions to different foods. For example, the robot could be shown an image of a desired slice of a particular food, and the embeddings from this image could be used to select cutting skill parameters to produce a similar-looking slice.

Another interesting extension to this work is having the robot monitor food while it is being cooked, which is an important skill that would enable the robot to perform more complex cooking tasks. The robot could interact with the food during the cooking process and use sensor information to determine to what extent the food is cooked and decide when to intervene, potentially in collaboration with a human. For example, grilling a steak is a nuanced task that requires knowledge of how well done the meat is and how well done the the human prefers it. The robot could poke or squeeze the steak regularly while it is cooking and use sensor information to determine when it is ready to be flipped or when cooking is complete.

On the topic of food cutting, an important area of future work on both the hardware and algorithmic side is in grasping deformable foods. The difficulties of grasping smaller, softer, low friction, and irregularly shaped foods are often overlooked, however this is still not a solved problem in robotic manipulation. Because these foods are difficult to grasp with our current end effectors, they often slip out of the grippers during cutting or are deformed by the grippers if grasped with too high force. To approach this, we likely need to develop new end effectors that enable more dexterous grasping compared to the simple parallel tongs we currently use. An end effector that better mimics how a human hand would grasp food while cutting without introducing too much complexity or reliability concerns would greatly improve the robot's ability to perform more complex food manipulation skills. On the control side, we can develop new algorithms that adapt grasping forces and location of these applied forces as the food is being manipulated such that the robot can better replicate a human's touch without deforming the food in its grasp.

Another area of future work is exploring how the robot can learn more complex behaviors that require in-hand or two-hand dynamic manipulation: for example, cutting a pitted fruit such as an avocado or peach, or peeling a vegetable that requires dynamic re-grasping as the food is being peeled. The question of how to optimally and dynamically grasp and re-grasp deformable foods while avoiding slippage is an interesting research area that would likely also have applications beyond food manipulation.

Finally, there is a great deal more to explore in the area of human-in-the-loop learning for cooking robots. For example, a worthwhile research direction is looking at other methods to incorporate human input in different ways during skill learning (e.g. preference-based rankings or active coaching during executions) and how to adapt the feedback the human gives to the robot as it gets better at the skill over the course of training. Since research in cooking robots is still in its early stages, we also have an opportunity to think creatively about how to bring the robot and human together to work collaboratively in the kitchen. For example, how can we optimally split up cooking tasks between the robot and the human? How do we best leverage the robot's strengths to perform the more tedious tasks and the human's strengths that lie more on the artistic side of cooking (e.g. garnishing and seasoning)? How do we impart the human element of creativity involved in cooking to the robot? In order to design robotic systems with end users in mind, it is critical that we continue to explore how the robot can best learn from and collaborate with a human in a home kitchen environment.

While there are many remaining challenges both on the hardware and software side before autonomous at-home cooking robots can be made a reality, the end goal has the potential to help people who are unable to cook and make home cooking more accessible to consumers. We hope the research presented in this thesis sparks other research directions and helps to to push forward to goal of bringing robots to home kitchens.

Bibliography

[1] Heni Ben Amor, Erik Berger, David Vogt, and Bernhard Jung. Kinesthetic bootstrapping: Teaching motor skills to humanoid robots through physical interaction. In *Annual conference on artificial intelligence*, pages 492–499. Springer, 2009. 2

[2] Aude G Billard, Sylvain Calinon, and Florent Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54(5): 370–384, 2006. 2

[3] Arne Böckmann and Tim Laue. Kick motions for the nao robot using dynamic movement primitives. In *Robot World Cup*, pages 33–44. Springer, 2016. 2

[4] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017. 2

[5] Pasu Boonvisut and M Cenk Çavuşoğlu. Estimation of soft tissue mechanical parameters from robotic manipulation data. *IEEE/ASME Transactions on Mechatronics*, 18(5):1602–1611, 2012. 2

[6] Sylvain Calinon and Aude Billard. Statistical learning by imitation of competing constraints in joint space and task space. *Advanced Robotics*, 23(15):2059–2076, 2009. 2

[7] Hyeong Soo Chang. Reinforcement learning with supervision by combining multiple learnings and expert advices. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006. 2

[8] Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Learning sequential motor tasks. In *2013 IEEE International Conference on Robotics and Automation*, pages 2626–2632. IEEE, 2013. 4.2.5

[9] Christian Daniel, Oliver Kroemer, Malte Viering, Jan Metz, and Jan Peters. Active reward learning with a novel acquisition function. *Autonomous Robots*, 39(3):389–405, 2015. (document), 2, 4.14, 4.2.7

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 3.1.2

[11] Mevlana C Gemici and Ashutosh Saxena. Learning haptic representation for manipulating deformable food objects. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 638–645. IEEE, 2014. 2

[12] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. Georgia Institute of Technology, 2013. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3.1.2

[14] Micha Hersch, Florent Guenter, Sylvain Calinon, and Aude Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*, 24(6):1463–1467, 2008. 2

[15] Phillip Isola, Joseph J. Lim, and Edward H. Adelson. Discovering states and transformations in image collections. In *CVPR*, 2015. 2

[16] Biao Jia, Zhe Hu, Jia Pan, and Dinesh Manocha. Manipulating highly deformable materials using a visual feedback dictionary. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 239–246. IEEE, 2018. 2

[17] Kshitij Judah, Saikat Roy, Alan Fern, and Thomas Dietterich. Reinforcement learning

via practice and critique advice. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010. 2

[18] Dov Katz and Oliver Brock. Manipulating articulated objects with interactive perception. In *2008 IEEE International Conference on Robotics and Automation*, pages 272–277. IEEE, 2008. 2

[19] W Bradley Knox and Peter Stone. Tamer: Training an agent manually via evaluative reinforcement. In *2008 7th IEEE International Conference on Development and Learning*, pages 292–297. IEEE, 2008. 2

[20] Oliver Kroemer and Gaurav Sukhatme. Meta-level priors for learning manipulation skills with sparse features. In *International Symposium on Experimental Robotics*, pages 211–222. Springer, 2016. 2

[21] Ian Lenz, Ross A Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*. Rome, Italy, 2015. 2

[22] Qiang Li, Oliver Kroemer, Zhe Su, Filipe Fernandes Veiga, Mohsen Kaboli, and Helge Joachim Ritter. A review of tactile information: Perception and action through touch. *IEEE Transactions on Robotics*, 2020. 2

[23] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, pages 1–11, 2000. 3.1.1

[24] Jan Matas, Stephen James, and Andrew J. Davison. Sim-to-real reinforcement learning for deformable object manipulation, 2018. 2

[25] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. 2015. 3.1.1

[26] Ioanna Mitsioni, Yiannis Karayiannidis, Johannes A Stork, and Danica Kragic. Data-driven model predictive control for the contact-rich task of food cutting. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 244–250. IEEE, 2019.

2

[27] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013. 2

[28] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010. 4.2.5, 4.2.7

[29] Amrita Sawhney, Steven Lee, Kevin Zhang, Manuela Veloso, and Oliver Kroemer. Playing with food: Learning food item representations through interactive exploration. *arXiv preprint arXiv:2101.02252*, 2021. 3

[30] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 3.1.2

[31] Mohit Sharma, Kevin Zhang, and Oliver Kroemer. Learning semantic embedding spaces for slicing vegetables. *arXiv:1904.00303*, 2019. 2

[32] Jivko Sinapov, Taylor Bergquist, Connor Schenck, Ugonna Ohiri, Shane Griffith, and Alexander Stoytchev. Interactive object recognition using proprioceptive and auditory feedback. *The International Journal of Robotics Research*, 30(10):1250–1262, 2011. 2

[33] Jivko Sinapov, Connor Schenck, Kerrick Staley, Vladimir Sukhoy, and Alexander Stoytchev. Grounding semantic categories in behavioral interactions: Experiments with 100 objects. *Robotics and Autonomous Systems*, 62(5):632–645, 2014. 2

[34] Gyan Tatiya and Jivko Sinapov. Deep multi-sensory object category recognition using interactive behavioral exploration. In *ICRA*, pages 7872–7878. IEEE, 2019. 2

[35] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations, 2020. 2

[36] Akihiko Yamaguchi and Christopher G Atkeson. Combining finger vision and optical

tactile sensing: Reducing and handling errors while cutting vegetables. In *Humanoids*, pages 1045–1051. IEEE, 2016. 3.1.1

[37] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation, 2020. 2

[38] K. Zhang, M. Sharma, M. Veloso, and O. Kroemer. Leveraging multimodal haptic sensory data for robust cutting. In *Humanoids*, pages 409–416, 2019. 2, 3.1.1, 4.1

[39] Kevin Zhang. https://github.com/firephinx/sounddevice_ros. 3.1.1

[40] Kevin Zhang, Mohit Sharma, Jacky Liang, and Oliver Kroemer. A modular robotic arm control stack for research: Franka-interface and frankapy. *arXiv preprint arXiv:2011.02398*, 2020. 3.1.1, 4.1.2