Learning to Represent and Accurately Arrange Food Items

Steven Lee

CMU-RI-TR-21-05
May 2021

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

Thesis Committee:
Oliver Kroemer, Chair
Wenzhen Yuan
Timothy Lee

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Arrangements of objects are commonplace in a myriad of everyday scenarios. Collages of photos at one's home, displays at museums, and plates of food at restaurants are just a few examples. An efficient personal robot should be able to learn how an arrangement is constructed using only a few examples and recreate it robustly and accurately given similar objects. Small variations, due to differences in object sizes and minor misplacements, should also be taken into account and adapted to in the overall arrangement. Furthermore, the amount of error when performing the placement should be small relative to the objects being placed. Hence, tasks where the objects can be quite small, such as food plating, require more accuracy. However, robotic food manipulation has its own challenges, especially when modeling the material properties of diverse and deformable food items. To deal with these issues, we propose a framework for learning how to produce arrangements of food items. We evaluate our overall approach on a real world arrangement task that requires a robot to plate variations of Caprese salads.

In the first part of this thesis, we propose using a multimodal sensory approach to interacting with food that aids in learning embeddings that capture distinguishing properties across food items. These embeddings are learned in a self-supervised manner using a triplet loss formulation and a combination of proprioceptive, audio, and visual data. The information encoded in these embeddings can be advantageous for various tasks, such as determining which food items are appropriate for a particular plating design. Additionally, we present a rich dataset of 21 unique food items with varying slice types and properties, which is collected autonomously using a robotic arm and an assortment of sensors. We perform additional evaluations that show how this dataset can be used to learn embeddings that can successfully increase performance in a wide range of material and shape classification tasks by incorporating interactive data.

In the second part of this thesis, we propose a data-efficient local regression model that can learn the underlying pattern of an arrangement using visual inputs, is robust to errors, and is trained on only a few demonstrations. To reduce the amount of error this regression model will encounter at execution time, a complementary neural network is trained on depth images to predict whether a given placement will be stable and result in an accurate placement. We demonstrate how our overall framework can be used to successfully produce arrangements of Caprese salads.

Acknowledgments

I would like to first thank my advisor, Professor Oliver Kroemer, for his remarkable support and supervision over the two years that I have spent at Carnegie Mellon University. Words cannot completely convey my gratitude for the guidance, opportunities, and companionship that he has provided. I am also grateful for the wonderful lab environment that he has formed around him. My lab mates in IAM Lab have been extremely helpful in my research, academics, and personal life. Amrita Sawhney has been an amazing lab mate and friend over the last two years as we laughed and struggled through our work together. Kevin Zhang, Mohit Sharma, Jacky Liang, Tetsuya Narita, and Alex LaGrassa have all always been there to provide me with guidance when I'm stuck and friendship outside of research. Pragna Mannam, Shivam Vats, and Saumya Saxena have been kind and helpful since I've joined IAM Lab. Although the last two years have been incredibly stressful at times, this environment made it possible for me to have come this far.

Additionally, I would like to thank the other members of my committee. I would like to thank Professor Wenzhen Yuan for the support and generosity she has shown to me by always making the time for aiding me when I request it. Also, I would like to thank the last member of my committee, who is also my lab mate, Timothy Lee for his guidance and feedback on my research, as well as my career path. He has been exceptionally kind and helpful to me since I have joined the IAM Lab and I'm very grateful to him for it.

I'm also grateful for the funding received from Sony AI that enabled the research discussed in this thesis. I would like to thank my friends for their emotional support and companionship over the many years we have known one another.

Finally, I would like to thank my parents and sisters for their love and support throughout my life. My parents sacrificed their dreams, passions, and personal relationships to come to a foreign land and support my sisters and I so we could live better lives than them. I don't think I can ever properly convey the respect and gratitude I have to them for raising me despite their struggles. And to my five older sisters, I would to thank them for their love, kindness, and advice over the years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Learning to accurately recreate arrangements from visual demonstrations is a process that humans regularly undergo when learning to perform certain tasks, such as arranging displays at department stores, decorating desks at home, or plating food at restaurants. For robots that will be deployed in customer service or home environments, being able to easily learn various arrangements from visual demonstrations is a valuable ability. The process for learning arrangements typically consists of two sub-tasks: 1) Making high-level decisions on the appearance of the arrangement, and 2) Performing the physical actions of picking and placing the objects according to those high-level decisions.

When making the high-level decisions for arrangements, the choice of what objects should be used for an arrangement can drastically affect the final result. For example, deciding to place a whole tomato part way through arranging a Caprese salad that only contains slices would not look as visually appealing as using a disc-shaped tomato slice. Having knowledge on an object's material properties and shape can help to inform these types of decisions. Furthermore, this information is important for robots learning to perform tasks that involve physical interactions, since modeling object dynamics can be crucial for said tasks. However, obtaining this information for food items can be difficult and time consuming. Food items in particular can vary widely between and within food types depending on factors such as ripeness, temperature, how they were stored, and whether they have been cooked [1]. Humans, as a result of our prior knowledge and access to multiple forms of sensory information, can roughly ascertain material properties of food items through interacting with them. For example, one can squeeze an avocado to identify its ripeness. Analogously, there is evidence that the ability to distinguish properties between food items can be learned using multimodal sensor data from interactive robot exploration [2–4]. The sensory feedback of these interactions can form the basis for learning visual representations that are better suited to capturing the material properties of objects.

This thesis is composed of two works. In chapter 2, we explore a method of learning food items embeddings, in an self-supervised manner, that encodes various material properties using multiple modes of sensory information. As a means to learn these embeddings, a unique multi-modal food interaction dataset is collected consisting of vision, audio, proprioceptive, and force data, which is acquired autonomously through robot interactions with a variety of food items. The network used to learn these embeddings is trained using a triplet loss formulation, which groups similar and dissimilar data samples based on the different modalities of interactive sensor data. In this manner, the robot can learn complex representations of these items autonomously without the need for subjective and time-consuming human labels.

Once these food item representations are learned, they can be utilized to improve different manipulation tasks. For example, they can be used as a basis for determining what items are appropriate for an arrangement's appearance. However, making high-level decisions on an arrangement's appearance also require that the robot can adapt to varying object sizes and unexpected misplacements that may occur, while only requiring a small number of training demonstrations. Furthermore, a robot making these decisions would also need to take into account human preferences, which is difficult since there is not an unbiased and quantitative way to measure whether an arrangement will be appealing to humans or not.

Even if a decision making system is able to accomplish these feats, robotic placements that lead to large and irreversible errors can still occur. Therefore, it is also essential that the resulting placements are as accurate as possible. One could try to resolve these errors after the placement by physically adjusting their positions, but this could lead to more operation costs or even damage the objects. For example, this would be impractical for tasks involving delicate and deformable objects, such as plating thin slices of food. Instead, we can learn a method to determine when these types of errors may occur.

In chapter 3, we aim to address these issues by learning how to accurately recreate the patterns found in arrangements in a manner that is robust and can accommodate for errors in placement, while still accommodating human preferences. For making decisions on where to place objects, we learn a local product of experts model for each placement, which is trained using a low number of samples and can easily incorporate additional information through weighting of the training samples and additional experts. We also train a complimentary neural network to perform binary classification on whether a particular object, and the surface it will be placed on, will lead to a misplacement when given only depth images of the two. Rather than trying to accurately predict complex shifting behaviours using regression, our approach focuses on identifying locations where the placed objects will not shift significantly.

To demonstrate the utility of the learned food item representations, which will be discussed in chapter 2, we use them as input features to train regressors and classifiers for different tasks and compare their performances with baseline vision-only and audio-only approaches. For our high-level arrangement approach, which we will discuss in chapter 3, we evaluate it on multiple different arrangements. Similarly, we evaluate our binary classification network on a variety of placements in simulation and the real world. We evaluate our entire pipeline in the real world by constructing multiple plating arrangements of Caprese salads.

Chapter 2

Learning Food Item Representations through Interactive Exploration

2.1 Introduction

In this chapter, we explore a self-supervised method to learning food item representations that embed various material properties and uses only visual input at test time. These embeddings are learned from a self-procured, unique multimodal dataset consisting of audio, vision, proprioceptive, and force data acquired autonomously through robot interactions with a variety of food items. The neural network used to learn these embeddings is trained using a triplet loss formulation, where groups of similar and dissimilar samples are formed based on the different modes of the interactive sensor data. In this manner, the robot can learn complex representations of these items autonomously without the need for subjective and time-consuming human labels. Furthermore, we share our dataset, which consists of 21 unique food items with varying slice types and properties, here[1]. Our project website can also be found here[2].

We demonstrate the utility of the learned representations by using them as input features to train regressors and classifiers for different tasks and compare their performances with baseline vision-only and audio-only approaches. Our experiments show that classifiers and regressors that use our learned embeddings outperform the baseline approaches on a variety of tasks, such as hardness classification, juiciness classification, and slice type classification. These results indicate that our visual embedding network encodes additional material property information by utilizing the different modalities, while only requiring the robot to interact with the object at training time. The work described in this chapter was performed in collaboration with Amrita Sawhney and Kevin Zhang in [5].

[1]https://tinyurl.com/playing-with-food-dataset
[2]https://sites.google.com/view/playing-with-food

2.2 Related Work

Researchers have made contributions on learning deep embeddings of objects in order to create useful representations, which can be used on a wide range of tasks [6–8]; however, performing the same task for food items has not been studied extensively. Sharma et al. [9] learn a semantic embedding network, based on MaskRCNN [10], to represent the size of food slices in order to plan a sequence of cuts. Isola et al. [11] use collections of human labeled images to generalize the states and transformations of objects, such as varying levels of ripeness in fruit. Although the above works learn embeddings for food objects, both of them focus on using solely visual inputs during both train and test time. By contrast, the methods discussed in this chapter incorporate additional synchronized multi-modal sensory information during the training phase of our vision-based embedding networks.

Instead of learning deep embeddings through vision, many recent works aim to learn the dynamics models and material properties of deformable objects through simulation [12, 13]. For example, Yan et al. [14] learn latent dynamics models and visual representations of deformable objects by manipulating them in simulation and using contrastive estimation. Matl et al. [15] collect data on granular materials and compare the visual depth information with simulation results in order to infer their properties. By comparison, the work discussed in this chapter involves collecting and exploiting data from real-world robots and objects, since representations of food that are learned through a simulation environment may not accurately transfer to real world. This is due to the variable behaviors of food during complex tasks, such as large plastic deformations during cutting. There are simulators that can simulate large elasto-plastic deformation [16], but they are computationally expensive, unavailable to the public, and have not yet shown their efficacy in this particular domain.

Other works also use multimodal sensors to better inform a robot of deformable object properties for more sophisticated manipulation [4, 17, 18]. Erickson et al. [19] use a specialized near-infrared spectrometer and texture imaging to classify the materials of objects. Meanwhile, Feng et al. [20] use a visual network and force data to determine the best location to skewer a variety of food items. Tatiya et al. [21] utilize video, audio, and haptic data as input to a network to directly categorize objects into predefined classes. Finally, Zhang et al. [22] use forces and contact microphones to classify ingredients for obtaining the necessary parameters to execute slicing actions. In contrast, the work discussed in this chapter only uses images as input to our embedding network during inference time, while still incorporating multi-modal data during training.

Finally, numerous works have focused on using interactive perception to learn about objects in the environment [2, 23]. Yuan et al. [24, 25] use the images from GelSight [26] sensors to create embeddings of deformable objects to determine the tactile properties of cloth. Conversely, our work aims to learn material properties of deformable food items. Katz et al. [27] use interactive perception to determine the location and type of articulation on a variety of objects. Sinapov et al. [21, 28, 29] have a robot interact with objects using vision, proprioception, and audio in order to categorize them. Chu et al. [30] utilize 2 Biotac haptic sensors on a PR2 along with 5 exploratory actions in order to learn human labeled adjectives. In the work discussed in this chapter, we focus our exploratory data collection on deformable food objects instead of rigid ones.

2.3 Robotic Food Manipulation Dataset

2.3.1 Experimental Setup

Our robotic food manipulation dataset was collected using a pipeline of two different experimental setups: one setup for robotic food cutting and another for food playing. For the food cutting setup, the robot cuts food items, using predefined cutting actions, into specific types of slices. For the food playing setup, the robot physically interacts with these food slices in a fixed sequence of actions. The specific types of slices and the sequence of actions will be discussed later in this section. The multi-modal sensor data collected during the cutting and playing data collection processes was gathered using the Franka Emika robot control framework mentioned in [31]. The following sections describe the 2 experimental setups in further detail.

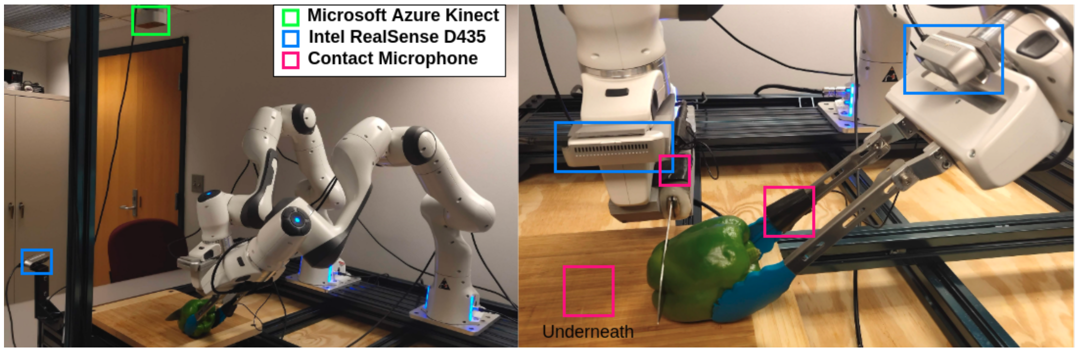


Figure 2.1: The experimental setup for collecting robot cutting data described in Section 2.3.1. The colored boxes show the locations of the cameras and contact microphones used to record visual and audio data respectively. Note that the contact microphones are not clearly visible in the images shown.

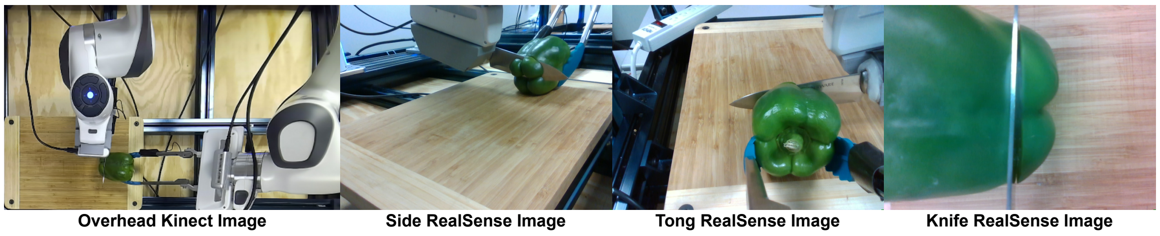


Figure 2.2: Example images from the 4 cameras used in the robotic food cutting data collection setup described in Section 2.3.1.

Robot Cutting:

The experimental setup for the robotic cutting includes two Franka Emika Panda Arms mounted on a Vention frame with a cutting board in the center, which is shown in Figure 2.1. One arm has its end-effector fingers replaced with a pair of mounted 8” kitchen tongs, while the other is grasping a custom 3D printed knife attachment made of PLA. Four cameras are used in this setup: an overhead Microsoft Azure Kinect Camera mounted on the Vention frame, a side-view Intel Realsense D435 camera that is also mounted on the Vention frame, another D435 camera mounted above the wrist of the robot holding the knife, and a third D435 camera mounted on the wrist of the robot holding the tongs. Example images from each of the 4 cameras are shown in Figure 2.2. Additionally, there are 3 contact microphones: one mounted underneath the center of the cutting board, another mounted on the knife attachment, and the last mounted on one of the tongs.

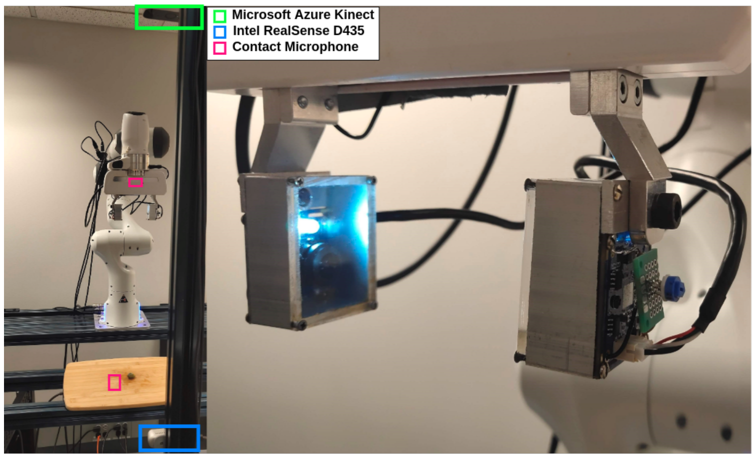


Figure 2.3: The experimental setup for collecting data as the robot plays with food. The setup is described in Section 2.3.1. The colored boxes in the left image show the locations of the cameras and contact microphones used to record visual and audio data respectively. Note that the contact microphones are not clearly visible in the images shown. The right image shows the FingerVision cameras.

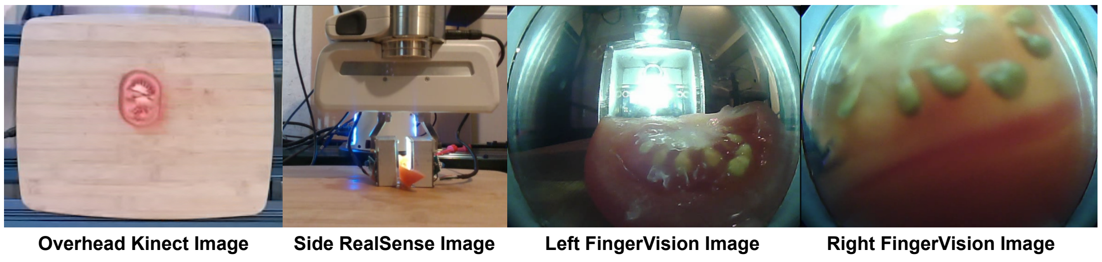


Figure 2.4: Examples images of a cut tomato from the Kinect, Realsense, and FingerVision cameras described in Section 2.3.1.

Robot Playing:

For the robotic food playing setup, a single Franka Emika Panda Arm is mounted on a Vention frame with a cutting board in the center, as shown in Figure 2.3. An overhead Microsoft Azure Kinect Camera and a front-view Intel Realsense D435 camera, which is facing the robot, are mounted to the Vention frame. A fisheye 1080P USB Camera [3] is attached to each of the end-effector fingertips as in FingerVision [32]. It should be noted that a laser-cut clear acrylic plate cover is used instead of a soft gel-based cover over the camera. This acrylic plate allows for observation of the compression of the object being

[3]https://www.amazon.com/180degree-Fisheye-Camera-usb-Android-Windows/dp/B00LQ854AG/

grasped relative to fingertips. A white LED is added to the interior of the FingerVision housing to better illuminate objects during grasping. Example images from all of these cameras are shown in Figure 2.4. To record audio information, 2 contact microphones are used: one is mounted underneath the center of the cutting board and the other is mounted on the back of the Franka Panda hand. The Piezo contact microphones [4] from both cutting and playing setups capture vibro-tactile feedback through the cutting board, end-effector fingers, and tools. The audio from the contact microphones of both the cutting and playing setup are captured using a Behringer UMC404HD Audio Interface [5] and synchronized with ROS using sounddevice_ros [33].

2.3.2 Data Collection

Using the robotic food cutting setup described in Section 2.3.1, we teach the robot simple cutting skills using Dynamic Movement Primitives (DMPs) [34, 35]. Ridge regression is used to fit the DMP parameters to trajectories, which were collected using kinesthetic human demonstrations as in [22]. These DMPs are then chained into multiple slicing motions until the food items were completely cut through.

Data was collected for 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. For each of the food types, we obtained 10 different slice types. It should be noted that across all 21 food types, there are 14 different types of slices, which were created using similar skill parameters across the food types. The skill parameters vary in slice thickness from 3mm to 50mm, angles from ± 30 degrees, and slice orientation where we had normal vs. in-hand cuts when the knife robot cut between the tongs. 14 slice types were used instead of 10 because not all slice types can be executed on every food type, especially the angled cuts, due to the variations in size across food types. As the slices are obtained, audio, image, and force data is collected. Figure 2.5 shows the resulting slices for each of the food types. In total, the cutting data in our dataset consists of 210 samples of audio, visual, and force data, one for each of the different slices.

After the different slice types have been created, they are transferred to the robotic food playing setup to begin the remainder of the data collection process. First, one of the slices is manually placed on the cutting board at a random position and orientation. A RGBD image of the slice is then captured from the overhead Kinect camera and square cropped to the cutting board's edges. Next, the robot performs a sequence of actions, which we refer to as the playing interactions, on the single slice. To begin these interactions, the center of the object is manually specified and the robot then closes its fingers and pushes down on the object until 10N of force is measured. The robot's position and forces during this action are recorded. Next, the robot resets to a known position, grasps the object, and releases it from a height of 15cm. Audio from the contact microphones and videos from the Realsense cameras are recorded during the push, grasp, and release actions. Videos from the FingerVision cameras are only recorded during the grasp and release actions. Additionally, the gripper width when the grasp action has finished is recorded. Finally, a

[4]https://www.amazon.com/Agile-Shop-Contact-Microphone-Pickup-Guitar/dp/B07HVFTGTH/
[5]https://www.amazon.com/BEHRINGER-Audio-Interface-4-Channel-UMC404HD/dp/B00QHURLHM

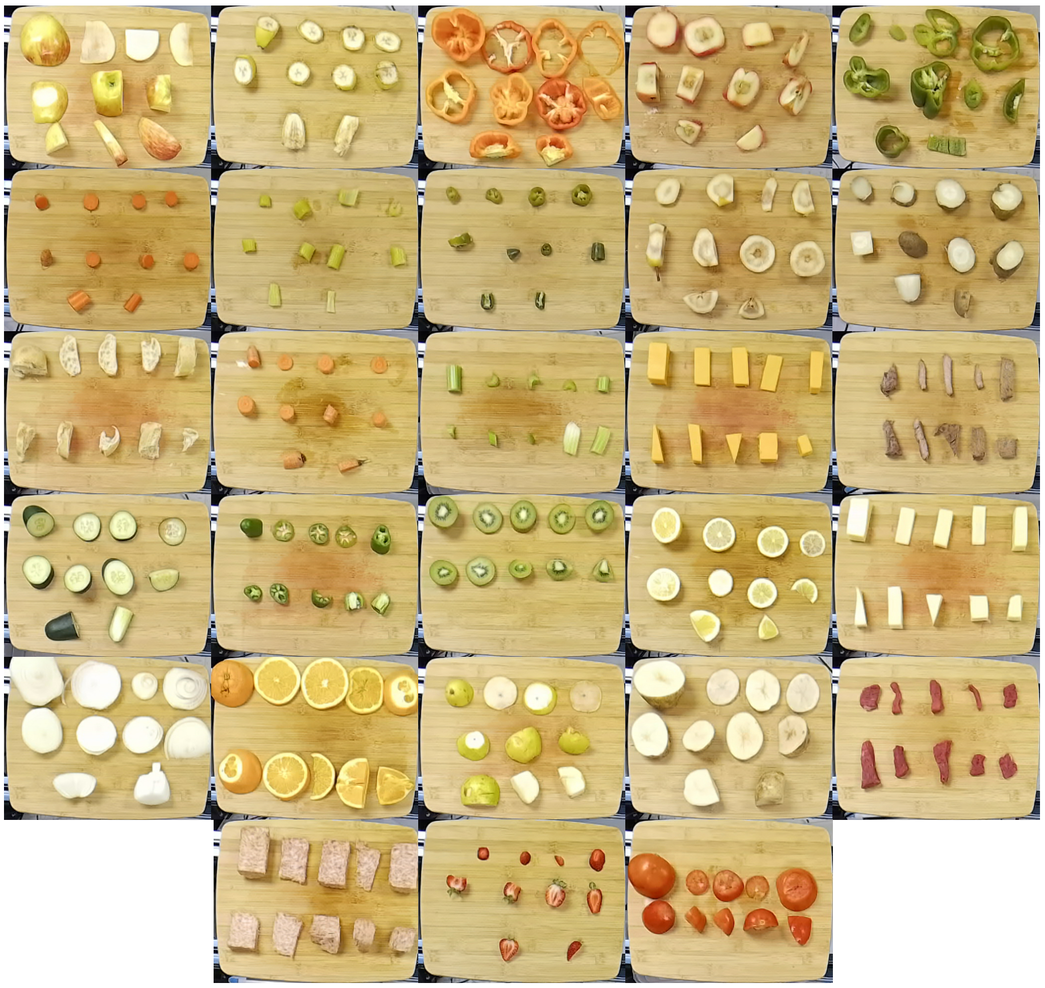


Figure 2.5: Example images of the food item slices in our dataset. From left to right and top to bottom, the images show: apple, banana, bell pepper, boiled apple, boiled bell pepper, boiled carrot, boiled celery, boiled jalapeno, boiled pear, boiled potato, bread, carrot, celery, cheddar, cooked steak, cucumber, jalapeno, kiwi, lemon, mozzarella, onion, orange, pear, potato, raw steak, spam, strawberry, and tomato. Note that the boiled food items are part of the auxilary dataset that is discussed in Section 2.5

RGBD image from the overhead Kinect camera is captured. This sequence of actions is performed 5 times in total on each of the 10 slice types from each of the 21 food types, meaning there are 1050 samples of audio, visual, and proprioceptive data in total for the playing data. It should be noted that in order to capture variations in the objects' behaviors due to differing initial positions, orientations, and changes over time, we randomly place the slices at the beginning of each of the 5 trials.

The full dataset is available for download here[6]. The data is located in the appropriately named folders and are sorted first by food type, then slice type, and finally trial number. Additionally, food segmentation masks are provided in the silhouette data folder. Deep Extreme Cut (DEXTR) [36] was used to obtain hand labeled masks of the objects in the

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

overhead and side view images. A PSPNet [37], pre-trained on the Ade20k [38] dataset, is then fine-tuned using the manually labeled masks previously mentioned. This fine-tuned network is then used to generate additional segmentation masks of the other images in the dataset. Lastly, additional playing data is provided in the *old_playing_data* folder. However, the slices in this dataset were all hand cut, and the data was collected using a different experimental environment.

2.3.3 Data Processing

Features are extracted from the multi-modal data so embeddings networks can be trained. We will discuss in more detail how these features are used for training in Section 2.4. For the audio data, the raw audio that was recorded during the cutting and playing experiments (playing audio was recorded during the release, push-down, and grasp actions), was transformed into into Mel-frequency cepstrum coefficient (MFCC) features [39] using Librosa [40]. MFCC features are used because they have been shown to effectively differentiate between materials and contact events [22]. Subsequently, PCA is used to extract a lower-dimensional representation of the cutting ($\mathcal{A}_{cut}$) and playing ($\mathcal{A}_{play}$) audio features.

Proprioceptive features ($\mathcal{P}$) are formed using 3 values, which are extracted from the robot poses and forces recorded during the push down and grasp actions. The first of the 3 values is the final z position (z_f) of the robot's end-effector, which is recorded during the push down action once 10N of force has been reached. z_f is then used to find the change in z position between the point of first contact and z_f. We will refer to this distance value as Δz, which provides information on an object's stiffness. The last value for $\mathcal{P}$ is the final gripper width (w_g) recorded during the grasping action once 60N of force has been reached. These three values (z_f, Δz, and w_g) are combined to form $\mathcal{P}$.

Lastly, the slice type labels ($\mathcal{S}$) and food type labels ($\mathcal{F}$) for each sample are used as features. These labels are created according to the enumerated values that identify the slice type performed during cutting and the food type of each individual sample, respectively. The features discussed in this section are used as metrics for differentiating between similar and dissimilar samples. We will discuss this in further detail in Section 2.4

2.4 Learning Food Embeddings

Convolutional neural networks are trained in an unsupervised manner to output embeddings from images of the food items. These images are taken from the overhead Kinect camera mentioned in Section 2.3.1. Our architecture is comprised of ResNet34 [41], which is pre-trained on ImageNet [42] and has the last fully connected layer removed. Three additional fully connected layers, with ReLU activation for the first two, are added to reduce the dimensionality of the embeddings. A triplet loss [43] is used to train the network, so similarities across food and slice types are encoded in the embeddings. The loss is given by

$$Loss = \sum_{i}^{N} \left[\left\| f\left(x_i^a\right) - f\left(x_i^p\right) \right\|_2^2 - \left\| f\left(x_i^a\right) - f\left(x_i^n\right) \right\|_2^2 + \alpha \right]_+ \tag{2.1}$$

where $f()$ is the embeddding network described above, N is the number of samples, x_i^a is the i^{th} triplet's anchor sample, x_i^p is the i^{th} triplet's positive sample, x_i^n is the i^{th} triplet's negative sample, and α is the margin that is enforced between positive and negative pairs.

The different modalities of data mentioned in Section 2.3.3 are used as metrics to form the triplets and a seperate neural network is trained using each of these metrics. More specifically, 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 metrics. When combining multiple features (or modalities), we concatenate the output embeddings that were learned from the respective networks. The $\mathcal{F}$ and $\mathcal{S}$ values are directly used to differentiate between samples. For example, when using $\mathcal{F}$ as a metric, samples of tomatoes are considered positive samples with one another and samples of any other food type are identified as negative samples. For the other metrics ($\mathcal{A}_{play}$, $\mathcal{A}_{cut}$, $\mathcal{P}$, and $\mathcal{A}_{play}$+$\mathcal{P}$), we define the n nearest samples in feature space (using the L2 norm) as the possible positive samples in a triplet and all other samples as the possible negative samples, where n is a hyperparameter ($n = 10$ was used for our experiments). Prior to training, we identify all possible positive and negative samples for every sample in the training dataset. At training time, triplets are randomly formed using these positive/negative identifiers. Figure 2.6 shows an overview of our approach.

To evaluate the utility of these learned embeddings, we train multiple 3-layer multilayer perceptron classifiers and regressors for a variety of tasks, using the learned embeddings as inputs. These results are discussed in Section 2.5.

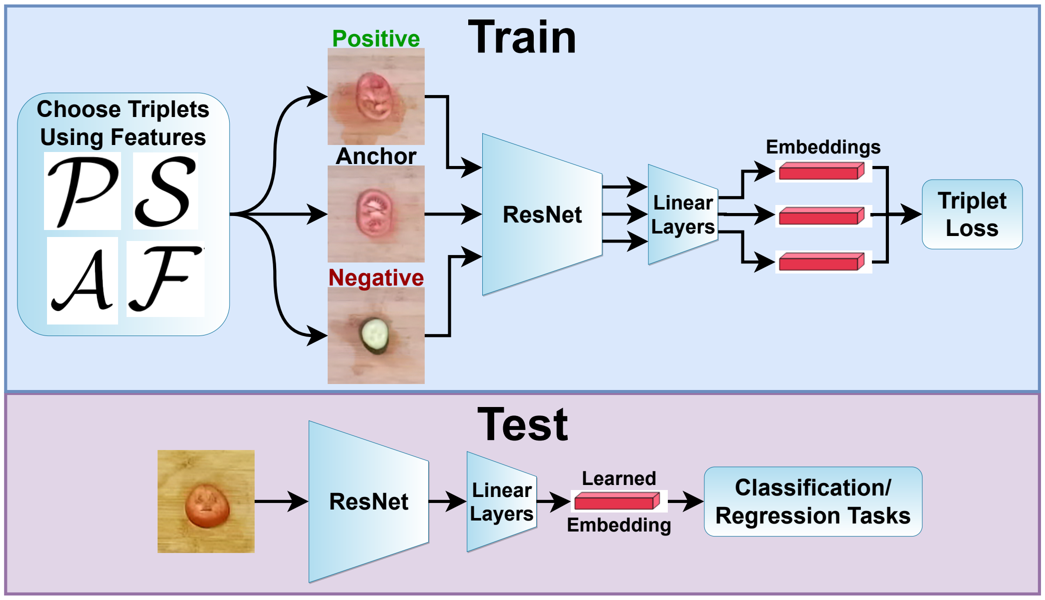

Train
Choose Triplets Using Features
$\mathcal{P}$ $\mathcal{S}$ $\mathcal{A}$ $\mathcal{F}$
Positive
Anchor
Negative
ResNet
Linear Layers
Embeddings
Triplet Loss
Test
ResNet
Linear Layers
Learned Embedding
Classification/ Regression Tasks

Figure 2.6: An overview of our approach to learning embeddings from our dataset. The different features (modalities) defined in Section 2.3.3 are used to form triplets to learn embeddings in an unsupervised manner, which are used for supervised classification and regression tasks (blue text).

2.5 Experiments

As described in Section 2.4, embedding networks were trained using 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 ($\mathcal{A}_{play}$+$\mathcal{P}$) for creating triplets. These embeddings were then used to train multiple multi-layer perceptrons to predict the labels or values for 5 different tasks: classifying food type (21 classes), classifying the hardness (3 human-labeled classes - hard, medium, and soft), classifying juiciness (3 human-labeled classes - juicy, medium, dry), classifying slice type (14 different classes based on the types of cuts the robot performed), and predicting slice width (the width of the gripper after grasping, w_g). The loss function used for predicting the slice width is the mean squared error, while categorical cross entropy loss [44] is used for the other 4 classification tasks. For comparison, 2 baseline approaches are also trained on the 5 tasks. One baseline is a pre-trained ResNet34 network that is trained only using visual data and the other baseline is a 3-layer mutli-layer perceptron that uses only $\mathcal{A}_{play}$ as input. We chose these baselines to see how incorporating our learned embeddings, which utilize multiple modes of sensory information, can improve performance over methods that use only one mode of sensory information, such as visual or audio information.

To assess the generalizability of our approach, we evaluated the hardness and juiciness classification tasks based on leave-one-out classification, where we left an entire food class out of the training set and evaluated the trained classifiers on this class at test time. We then averaged the results across the 21 leave-one-out classification trials. The performance of all the trained networks on each of the tasks are shown in Table 2.1.

As shown in Table 2.1, the purely visual baseline outperforms our embeddings in the food type classification task. This is expected since ResNet was optimized for image clas-

Embeddings	**Food Type Accuracy - 21 classes (%)**	**Hardness Accuracy - 3 classes (%)**	**Juiciness Accuracy - 3 classes (%)**	**Slice Type Accuracy - 14 classes (%)**	**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 2.1: The table shows the baseline and multi-layer perceptron results on 5 evaluation tasks. The baseline results are shown on the last 2 rows, while the first 6 rows show results from the networks that used the learned embeddings, described in Section 2.4, as inputs. The best scores are shown in bold.

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 2.2: Results on 3 evaluation tasks for different learned embedding networks that were trained using the auxiliary cooked vs. uncooked dataset.

sification tasks and was pre-trained on the ImageNet dataset, which contains a vast number of labeled images. However, the ResNet baseline performs worse on the other 4 tasks, as ImageNet did not contain prior relevant information on the physical properties of the food items, which are important for these tasks. Additionally, ResNet was structured to differentiate object classes instead of finding similarities between them. Due to this, when an entire food class was left out of the training dataset, extrapolating the correct answer from the limited training data becomes much more challenging. Conversely, our learned embeddings contained auxiliary information that encoded the similarity of slices through various multimodal features, without ever being given explicit human labels. These results indicate that our interactive multimodal embeddings provided the multi-layer perceptrons with a greater ability to generalize to unseen data as compared to the supervised, non-interactive baselines.

Additionally, the results show that the audio embeddings provide some implicit information that can help the robot distinguish food types, which can be seen in the accuracy difference between $\mathcal{P}$ and $\mathcal{A}_{play}$+$\mathcal{P}$. It is also reasonable that the proprioceptive embeddings are more useful at predicting hardness and slice width as their triplets were generated using similar information. However, absolute labels were never provided when training the embedding networks, so the learned embeddings encoded this relevant information without supervision. It should also be noted that in the hardness and juiciness leave-one-out classification tasks, some food types, such as tomatoes, were more difficult to classify when left out of the training dataset than others, such as carrots. This may be due to the small size of our diverse dataset, which has few items with similar properties.

Lastly, with respect to the slice type prediction task, poor performance was observed across all the methods due to the inherent difficulty of the task. Since there was a high variation in shapes and sizes between food items, the slices generated by the cutting robot experiments greatly differed at times. Therefore, it is reasonable that only the embeddings trained using slice type labels as a metric performed relatively well on this classification task. Overall, the results of the evaluations indicate that our learned embeddings performed better on relevant tasks, which supports the hypothesis that they are encoding information on different material properties and can be advantageous for certain use cases.

Auxiliary Study with Additional Cooked vs. Uncooked Food Data

As an addendum to the 21-food class dataset described in Section 2.3.2, an additional dataset of boiled food classes was collected to explore and evaluate our method's ability to detect whether a food item is cooked or not through interactive learning. The additional boiled food classes collected were: apples, bell peppers, carrots, celery, jalapenos, pears, and potatoes. Each item was boiled for 10 minutes and can be seen in Figure 2.5. It should be noted that for these additional cooked food classes, the robot did not cut the food slices due to difficulties with grasping the objects. Instead, the slices were created manually by a human. These boiled classes were combined with their uncooked counterparts from the full dataset to form a smaller 14-class dataset. A subset of evaluations were conducted on embeddings learned from this dataset. Namely, predicting hardness, juiciness, and whether an item is cooked or not. All of these evaluations were performed as leave-one-out classification tasks. The results are shown in Table 2.2.

The auxiliary study with the boiled food dataset shows that the playing audio data is effective at autonomously distinguishing between cooked and uncooked food items, even with the absence of human-provided labels. This behavior is likely due to the significant changes in material properties that occur due to boiling. The high performance of the ($\mathcal{A}_{play}$+ $\mathcal{P}$) embeddings on the hardness, juiciness, and cooked leave-one-out classification tasks on this smaller dataset demonstrate that our approach can generalize to new data when a food class with similar properties was present during training. For example, the following pairs shared similar properties: apples and pears, potatoes and carrots, bell peppers and jalapenos).

Figure 2.7 visualizes the $\mathcal{A}_{play}$ of the different cooked and uncooked food classes in this dataset using the top 3 principal components. Figure 2.8 visualizes the top 3 principal components of the learned embeddings, which were based on $\mathcal{A}_{play}$. The colored dots represent uncooked items, while the thinner, colored tri-symbols represent cooked items. As shown in the plots, there is a distinct separation between the boiled and raw foods in the audio feature space, as well as in the learned embedding space. Within the cooked and uncooked groupings, there are certain food types that tend to cluster together. For example, the uncooked pear and apple cluster close together, which is reasonable given the similarities between the two fruits. Interestingly, they remain clustered close to one another even after they are cooked, even though there is a shift in the feature space between cooked and uncooked.

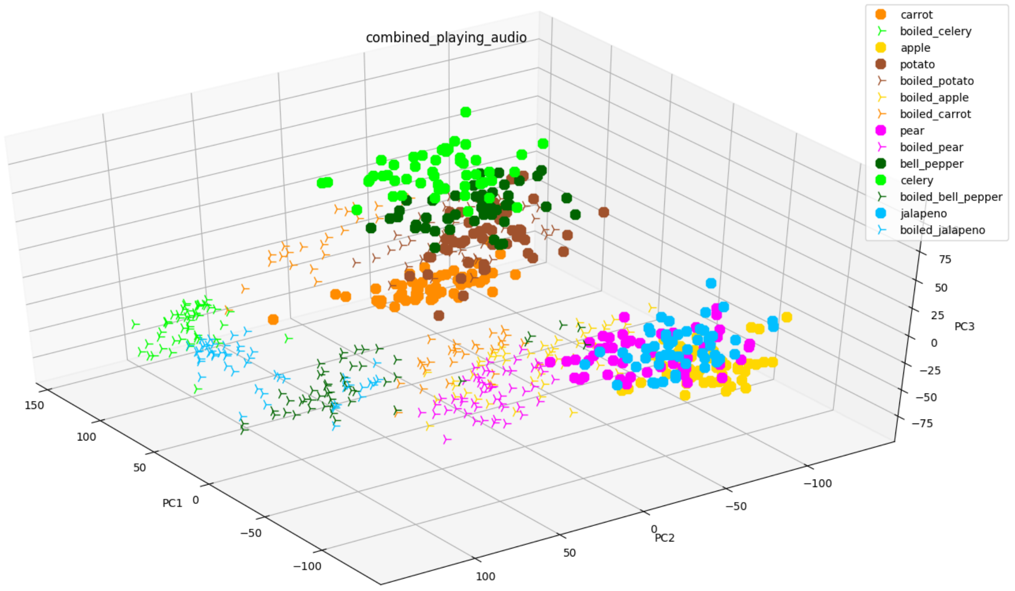


Figure 2.7: Visualization of the playing audio features, $\mathcal{A}_{play}$, in PCA space. The uncooked items are shown as dots, while the cooked items are shown as lined tri-symbols.

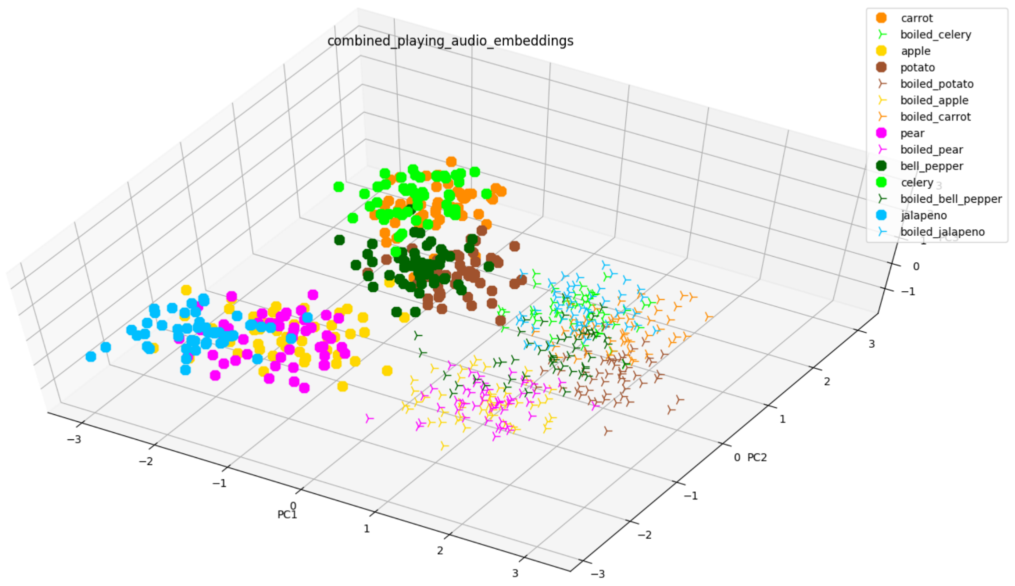


Figure 2.8: Visualization of the our learned embeddings, which were learned by using the playing audio features to choose triplets, in PCA space. The uncooked items are shown as dots, while the cooked items are shown as lined tri-symbols.

Chapter 3

Learning to Accurately Arrange Objects

3.1 Introduction

In this chapter, we aim to learn how to accurately recreate the patterns found in arrangements in a manner that is robust and can accommodate for errors, such as misplacements.Our approach consists of two sub-tasks: 1) Learning the underlying pattern of arrangements in a manner that is data efficient and robust to errors, while still preserving the fundamental shapes of the arrangements, and 2) Learning where to place objects locally so as to minimize the placement error. Our proposed method for the first task involves learning a local product of experts model for each placement, which is trained using a few samples and can easily incorporate additional information through weighting of the training samples and additional experts. We also explore the use of a recurrent neural network to perform predictions on where objects should be placed. For the second task, we train a neural network to perform binary classification on whether a particular object will lead to a stable placement on a specific surface given only depth images as input. Rather than trying to accurately predict complex shifting behaviours using regression, our approach focuses on identifying locations where the placed objects will not shift significantly.

We evaluate our proposed high-level arrangement approach on multiple arrangements and with different sets of experts and weighting hyper-parameters. Similarly, we evaluate our binary placement classification network on a variety of placements in simulation and the real world. Lastly, we evaluate our entire pipeline in the real world by producing multiple plating multiple arrangements of Caprese salads.

3.2 Related work

A significant amount of work has been performed on exploring different grasping techniques and gripper designs that improve the success of grasps [45–49]. Development of advanced grippers made specifically for food grasping tasks have also been explored [50–53]. Conversely, the amount of work related to robotic placing and arranging is not as extensive. Many of the works that do investigate robotic placing and arranging either only perform placements on flat and stable surfaces, or are performed on tasks that do not require a high degree of accuracy [54–57]. Jiang et al. [56, 58] introduce different supervised learning approaches that can be used to find stable placement locations given point clouds of the objects and placement surface. However, these placements are performed on larger scale tasks, such as placing objects on a bookshelf or dish rack, that do not require the same amount of precision as the tasks we are evaluating on. Later, Jiang et al. [59, 60] learn to incorporate human preferences into their arrangements using human context in a 3-D scene. Our approach assumes that human preferences are inherently encoded in our training data, which is not as intensive to collect, considering we do not need to extract point clouds, human poses, or object affordances.

Other works have looked at high-level reasoning methods for dealing with arrangements. Fisher et al. [61] synthesize 3-D scenes of arrangements of furniture using a probabilistic model for scenes based on Bayesian networks and Gaussian mixtures. Some work use symbolic reasoning engines to plan complex manipulation tasks involving house keeping [62–64]. Sugie et al. [65] use rule-based planning to have multiple robots push a collection of tables in a room. Lozano-Perez et al. [66] use task-level planning to pick and place rigid objects on a table. These works deal with objects on a much larger scale or focusing on producing task-level plans and parameterized actions instead seeking specific placement locations. Cui et al. [67] utilize probabilistic active filtering and Gaussian processes for planning searches in cluttered environments. Moriyama et al. [68] plan object assemblies with a dual robot arm setup. Although these tasks work on smaller scale objects, they deal with rigid objects instead of deformable objects, such as food.

Finally, a limited number of works have also looked into robotic placement of food. Matsuoka et al. [69] use a general adversarial imitation learning framework to learn policies to plate tempura dishes. This work utilizes tempura ingredients, which are relatively rigid, for their arrangements, while our work deals with plating specific patterns of deformable objects. Jorgensen et al. [70] introduce a complex robotic pick and place system for pork in an industrial environment. Conversely, our system is aimed towards a home or kitchen environment and does not require a specialized robotic system, only a camera and access to a 3-D printer.

3.3 Technical approach

In this section, we will describe our approach for reproducing the patterns in different arrangements and the techniques used to perform accurate placements. Additionally, we will describe the process we used to collect our training data. An overview of our overall approach is depicted in Figure 3.1.

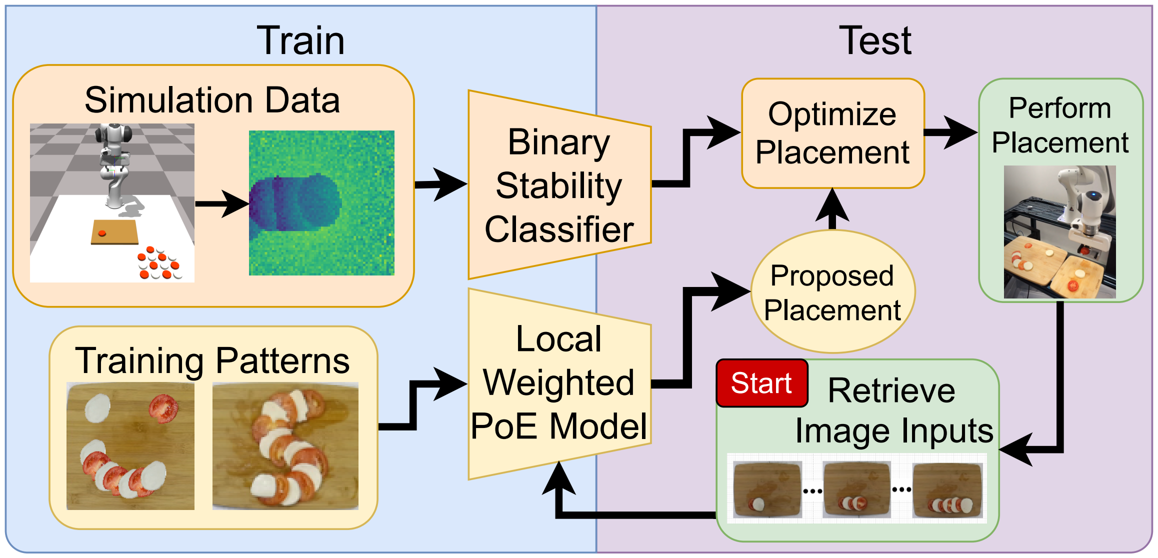


Figure 3.1: An overview of our approach to placing arrangements of objects. At each time step, a local product of experts regression model is trained on images of a specific pattern and used to make predictions on where the next object in a sequence should be placed. These predictions are optimized using a binary stability classifier that determines whether a placement will be stable or not.

3.3.1 Learning High-level Arrangements

Given sequences of images of object arrangements as training data, we aim to learn the principal pattern of the arrangements so it can be recreated in a robust and flexible manner. More specifically, the training data consists of sequences of images that are taken each time an object is placed. We extract the bounding box of the last object in each sequence and use this information to train the model to predict where we should place the objects to recreate the patterns. We evaluate 2 different types of models: One being a set of low sample local weighted regression models that use a product of experts formulation and the other being a recurrent neural network.

Local Weighted Product of Experts (PoE) Model

For a target arrangement that contains M objects, $(M-1)$ local regression models are trained, since the model for the initial object in an arrangement is either explicitly defined by the user or taken from the distribution of initial objects. Training data is extracted from the images and bounding box information mentioned above. Each image and bounding box pair across all the sequences in a training dataset is considered a training sample. We will describe the specific feature values extracted from these samples in more detail later in this section. Using this data, we train a Product of experts (PoE) model [71] to perform each of the $(M-1)$ predictions of where placements should be made to recreate patterns. The m^{th} model is given by

$$f_m\left(x \middle| \{\Delta_{mk}\}\right) = \frac{1}{Z}\prod_{k=1}^{K} f_{mk}\left(x \mid \Delta_{mk}\right) \tag{3.1}$$

where Z is a normalization factor, $m \in \{1, ..., M-1\}$, K is the number of experts for each local model, and Δ_{mk} are the k^{th} feature values that the m^{th} model is conditioned on. We will elaborate on these values later in this section.

For this task, we chose a multidimensional Gaussian distribution to represent each individual expert. Thus, the mean and inverse covariance of the resulting joint distribution for the m^{th} model are given as

$$\mu_m = C_m\left(\sum_{k=1}^{K} C_k^{-1}\mu_k\right) \tag{3.2}$$

$$C_m^{-1} = \sum_{k=1}^{K} C_k^{-1} \tag{3.3}$$

where μ_k and C_k are the mean and covariance of the k^{th} expert. These values are given by

$$\mu_k = \frac{\sum_{i=1}^{N} w_s^{(i)} w_t^{(i)} x^{(i)}}{\sum_{i=1}^{N} w_s^{(i)} w_t^{(i)}} \tag{3.4}$$

$$C_k = \frac{\sum_{i=1}^{N} w_s^{(i)} w_t^{(i)}\left(x^{(i)}-\mu\right)\left(x^{(i)}-\mu\right)^T}{1-\sum_{i=1}^{N}(w_s^{(i)} w_t^{(i)})^2} \tag{3.5}$$

where N is the number of training samples used, $x^{(i)}$ is the i^{th} training sample's feature value, and $w_s^{(i)}$ and $w_t^{(i)}$ are the sample weights for the N training samples. These weights are calculated using two distance metrics. One metric being their euclidean distance to a reference object. $w_s^{(i)}$ uses this metric to weigh training samples that are spatially closer more heavily. When training the m^{th} model, the reference object would be the $(m-1)^{th}$ object in the current arrangement and their euclidean distance would be calculated using the bounding box centers of each object. Note that $m = 0$ for the first object in an arrangement, for which we did not train a model. The other weight, $w_t^{(i)}$, uses the temporal distance as a metric for weighing samples more heavily. The temporal distance refers to the difference between the m value of the current model, which would be the reference value, and of the training samples. The weights are calculated using a normal distribution

$$w_s^{(i)}(x_s^{(i)}|\mu_s, \sigma_2) = \frac{1}{\sigma_s\sqrt{2\pi}}\exp\left(-\frac{1}{2}\frac{(x_s^{(i)}-\mu_s)^2}{\sigma_s^2}\right) \tag{3.6}$$

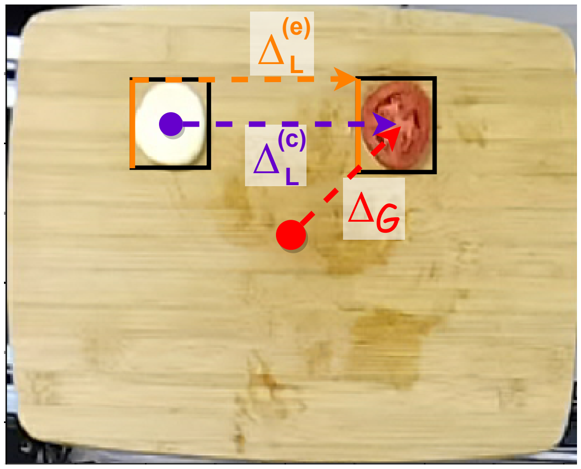


Figure 3.2: Features used to train the experts described in Sec. 3.3.1 for our experiments. The orange and purple lines shows local features conditioned on the distance between the objects' edges and centers, respectively. Red depicts a global feature conditioned on the distance between the centers of the placing surface and objects.

where σ_s is a free parameter and μ_s is the reference value, and x_s^i is cartesian coordinates of the center of i^{th} training sample's bounding box. The $w_t^{(i)}$ values are calculated in the same manner, but σ_s and μ_s are replaced with σ_t and μ_t, respectively.

The feature values we use to train each expert can be divided into two sub-classes, which we will refer to as local and global features. We will denote local and global features using L and G subscripts, respectively. The global feature data, Δ_G, is extracted from the bounding information by calculating the euclidean distance between a training sample and a fixed global reference point. This reference point is defined as the center of the placing surface.We train one expert using Δ_G.

For local features, we extract two types of data: $\Delta_L^{(c)}$ and $\Delta_L^{(e)}$. $\Delta_L^{(c)}$ is calculated as the distance between the m^{th} object's bounding box center relative to the center of the $(m-n)^{th}$ object, where both objects are in the same arrangement. Similarly, $\Delta_L^{(e)}$ is calculated as the distance between the m^{th} object's bounding box edge relative to the edge of the $(m-n)^{th}$ object, where both objects are in the same arrangement. We use the edges with the lowest x and y values when calculating $\Delta_L^{(e)}$. For the local features, n can range from 1 to $(M-1)$ and requires that we include a separate expert for each value of n. Note that the m^{th} model can only contain experts where $(m-n) \geq 0$. Figure 3.2 shows representations of the three global and local features.

We can use the σ_s and σ_t values to control the importance of samples for each feature. For example, we set σ_s to a large value (irrelevant) and σ_t to a small value (relevant) for expert conditioned on the global feature data, Δ_G. This Gaussian will tend to fit over all of the samples whose m value is closer to the current model's value more heavily. since it is more likely that those samples were placed in similar locations. For the experts conditioned on the local feature data, Δ_L, we set σ_s to a smaller value and σ_t to a value that is slightly higher than σ_s. We want these experts to fit over samples that have similar m values, but that are also close in proximity.

After the m^{th} model is learned, we use the resulting distribution to determine the next placement location. Cross entropy optimization is used by sampling placement locations across the placing surface and evaluating density values for each. We propose the sample that has the best score as the next placement location. This location is further optimized using the methods that will later be described in 3.3.2. The process is repeated for each m.

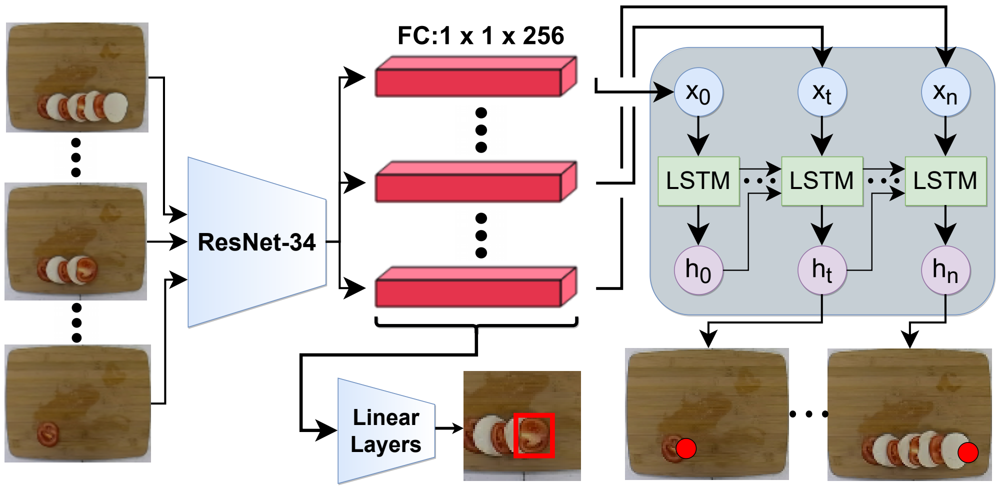


Figure 3.3: Network architecture for our recurrent neural network predictor. A sequence of images is given as input and predictions are made on the bounding box of the last object, as well as the where the next placement should be made.

Recurrent Neural Networks

Aside from the PoE approach mentioned in the previous section, we additionally explore a deep learning method using recurrent neural networks (RNN). The network is trained in a supervised manner to predict placement locations using sequences of images as training data. Figure 3.3 shows an overview of our network architecture. Each training sample for this network is a single fixed length sequence of images taken after each placement for a specific pattern. Note that we train separate models for different patterns. For a single sample, we first pass each image in the sequence through ResNet-34 [41], which has its last layer removed, and and an additional 3 linear layers to create 256 length embeddings. We then use these embeddings for two separate predictions. The first prediction is obtained by passing the sequence through LSTM units to predict the placement location, which is compared to the ground truth value since we know the bounding box of where the placement will be made during training. As a means to improve the network's ability to make these predictions, we additionally make predictions on the bounding box of the last object in the input images. We use the ResNet embeddings to make predictions on the bounding box center coordinates and dimensions after they have been passed through another 3 linear layers to reduce the embedding size to 4. We use the Adam optimizer [72] during training and the loss we use is a combination of L2 errors given by

$$Loss = \frac{1}{n}\sum_{i=1}^{n}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right]+\frac{1}{n}\sum_{i=1}^{n}\left[(u_i-\hat{u}_i)^2+(v_i-\hat{v}_i)^2\right]+$$
$$\frac{1}{n}\sum_{i=1}^{n}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right] \quad (3.7)$$

where n is the number of training samples per iteration, (x_i, y_i) are the coordinates of the predicted placement location, $(\hat{x}_i, \hat{y}_i)$ is the ground truth placement location, (u_i, v_i) are the coordinates of the center of the bounding box around the last object, and $(\hat{u}_i, \hat{v}_i)$ is the ground truth bounding box center, (w_i, h_i) are the bounding box dimension of the last object, and $(\hat{w}_i, \hat{h}_i)$ is the ground truth bounding box dimensions.

3.3.2 Placing Strategies

Once we have a proposed placement location, we wish to make the placement as accurate and reliable as possible. To accomplish this, we train a neural network to perform binary classification on whether a given placement will be stable, meaning that a large shift will not occur, or lead to a misplacement. Rather than trying to accurately predict complex shifting behaviours using regression, our approach focuses on identifying locations where the placed objects will not shift significantly. Additionally, we develop a gripper design to improve the consistency of the grasping and placing actions.

Binary Stability Classification:

We train a convolutional neural network to perform binary classification on whether a placement will be stable. More specifically, we are referring to whether an object's bounding box center will be within some distance, r, of the expected placement location after the object has been released from the robot's grippers. Note that movement along the vertical y-axis is ignored, only movements parallel to the table are considered when calculating these placement shift values that are to be compared to r. Additionally, a predefined placing skill is used from a fixed height to make the placements more consistent.

We use the network architecture proposed in [73]. The inputs to the network are two 64x64 depth images concatenated along the 3rd dimension and the end-effector pose. The first depth image is centered at the expected placement location, while the second depth image is centered at the center of the object that will be placed. The second image is taken prior to the object being grasped, but we refer to this object as the in-hand object. The two images give information on the surface of the placement area and the in-hand object, respectively. Additionally, we rotate the image input of the in-hand object according to what the object's rotation about the axis normal to the placing surface will be when being placed.

The loss is a weighted binary cross entropy loss that is weighted batchwise. The optimizer used was RMSProp [74]. We use curriculum learning [75] with a manual curriculum to train our network. Specifically, we decrease the value of r by 0.5 cm every 500 epochs. Figure 3.4 shows the training pipeline used for learning the weights of the binary stability classifier.

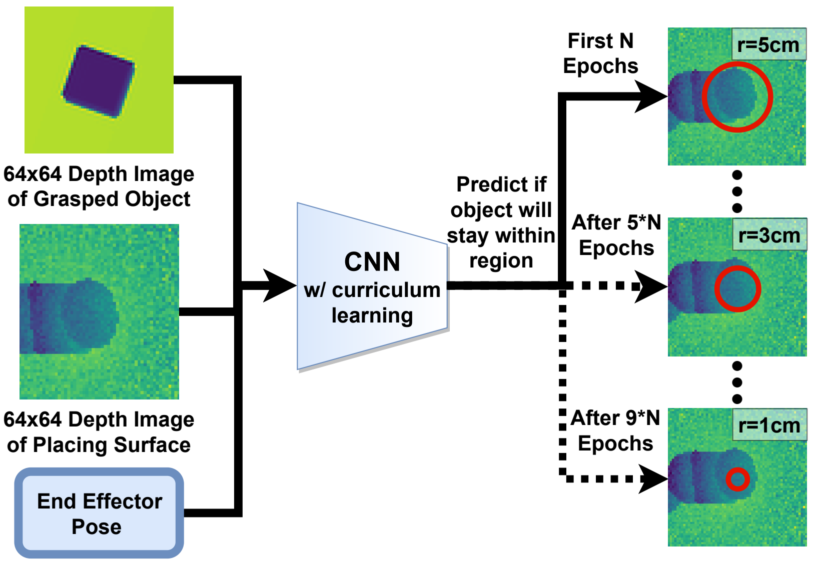

64x64 Depth Image of Grasped Object
64x64 Depth Image of Placing Surface
End Effector Pose
CNN w/ curriculum learning
Predict if object will stay within region
First N Epochs
After 5*N Epochs
After 9*N Epochs
r=5cm
r=3cm
r=1cm

Figure 3.4: Training pipeline for our binary stability classifier.

Note that when making predictions on placement stability, we assume that the object is grasped properly and would be placed within r of the expected location when placed on a flat surface. We perform the training with a rigid body assumption and will evaluate the performance in the real world on deformable food objects.

Gripper Design

Since our approach on determining whether a placement will be stable or not, described in Section 3.3.2, makes predictions according to the expected landing location, we propose a gripper design that will improve the accuracy and consistency of placements for a real robot. Our gripper uses a sliding pushing finger and spatula design, as shown in Figure 3.5. The spatula finger, shown as the gray portion, moves according to the robot finger's movements, while the pushing finger, shown as the black portion, is fixed to the robot's wrist and does not move when the spatula finger does. Knowing the location of the pushing finger and the width of the object, we determined the expected landing location as 0.5 the width of the object from the pushing finger's wall. This assumes the shift along the pushing finger's wall will not be significant.

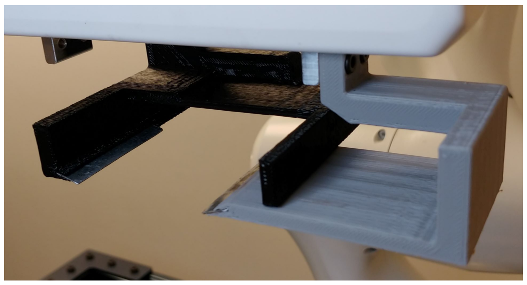


Figure 3.5: Real world gripper design used for our experiments and described in Section 3.3.2.

3.3.3 Data Collection

Here, we will describe our data collection process for the data used to train the models described in Section 3.3.1.

Pattern Data

First, we manually place cheese and tomatoes in an alternating manner to produce arrangements for multiple patterns, namely S-shapes, diagonal lines, and smiley faces. 4 sample arrangements were collected for each of these patterns. Each sample arrangement contained a sequence of objects of length, M, which ranged from 8 to 16. RGBD images were taken each time an object was placed, so each sample arrangement contained M images. We train a pretrained YOLOv3 [76] network on 1 hand labeled arrangement of each pattern type to predict the bounding boxes of the last object in each image, which are needed for the models described in Section 3.3.1.

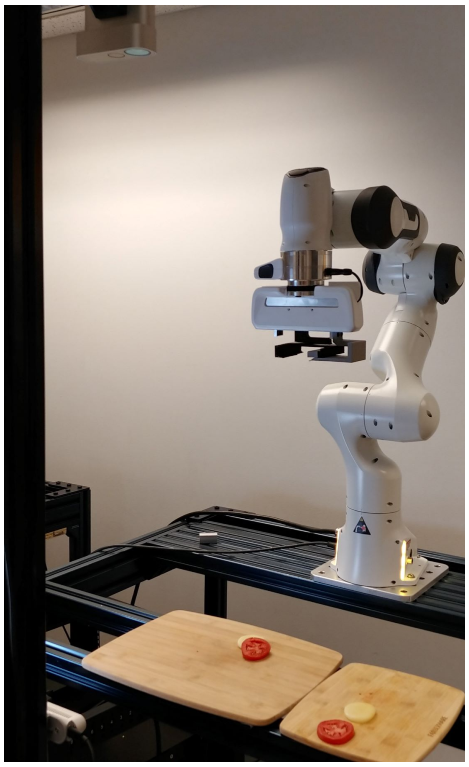
To supplement this data, we also create artificial sample arrangements. These were created by cropping images of the objects in the manually collected dataset. The crops were overlaid onto an image of the placing surface to simulate manual placing of a pattern. Each pattern was produced using predefined functions with Gaussian noise added. These functions being sine, quadratic, and linear functions. When gathering bounding box information for the artificial dataset, we directly used the placement information to create bounding box labels. Examples of the artificial data can be seen on Figure 3.10. Note that for all of our evaluations discussed in Section 3.4.2, we use the artificial images as the training data for the models.

Placing Data

The placement data was created by simulating approximately 50,000 placements of objects using NVIDIA's Isacc Gym simulator [77]. The simulator uses NVIDIA's FLEX physics engine. A simulation environment was created that contained a 1.2 m by 1.2 m table with a Franka Emika Panda Arm at the center of one end of the table. See Figure 3.6b. An overhead camera was placed above the scene at a fixed position and recorded depth images that are used to train the neural network described in Section 3.3.2. The objects being placed included thin rectangular prisms and discs of varying sizes. The height, width, and thickness of the objects being placed had mean values of 5cm, 5cm, and 1cm, respectively. In the case of the discs, the height and width values refer to the first and second radii. The dimensions of each placed object were scaled by a scaling factor that was sampled from a normal distribution centered at 0 with a standard deviation of 0.01. The objects could have either flat or rounded edges. The friction value of the objects were sampled from a normal distribution with a mean of 0.65 and standard deviation of 0.02.

Sequences of 6 to 10 objects were used for collecting the placing data. After each placement is performed, the pose of the object that was just placed is recorded and used to determine the pose of the next placement location. The magnitude of the offset for the next placement location is sampled uniformly from a value between 0% and 95% of the previous object's height or width. We always chose the smaller of the two to avoid making placements on flat surfaces. The direction of the offset is calculated by sampling a rotation from a normal distribution with mean 0 and a standard deviation of 45°. This rotation is applied relative to the orientation of the robot's end effector for the previous placement. Since we are assuming the grasp was perfect, the orientation of the object in hand is the same as the end effector. The height the robot releases the object from is a fixed height of 6cm.

The simulator checks that no irregularities occur during the placement, such as a placements that exceed the robot's joint limits, penetrate other objects, or land outside of the surface of the table. Any placements where these behaviors occur are ignored and the state of the objects were set to the state they were expected to be in. A sample used for training the binary stability classifier used only a single item in the sequence and was not dependent on the index in the sequence. Figure 3.6b shows an image of our simulation setup.


(a) Real robot setup.

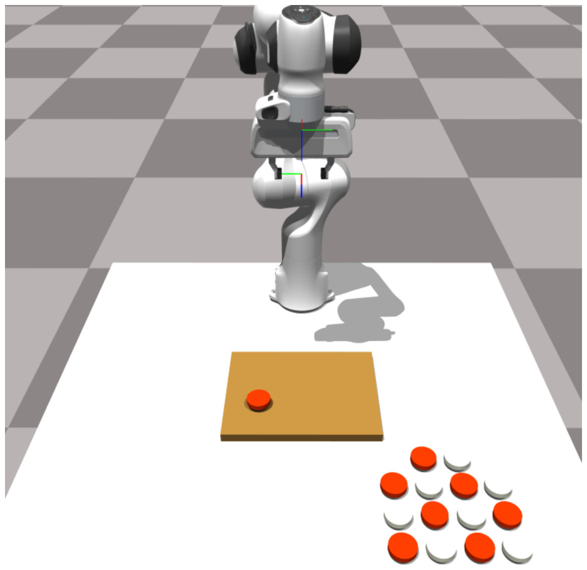

(b) Our simulation setup.

Figure 3.6: Fig. 3.6a shows our real robot setup described in Section 3.4.1 and Fig. 3.6b shows our simulation setup, which is described in Section 3.3.3

3.4 Experiments

Our overall approach is evaluated on a real robot, by generating 3 different plating patterns: a forward slash, a letter S, and a smiley face. Additionally, we evaluate the performance of the binary stability classifier, described in Section 3.3.2, on random placements made in simulation and the real world. We also investigate the effects that changing model parameters have on the PoE model. Specifically, we show the effect of changing the spatial and temporal weights (Eq.3.6), the number of experts used, and the training sample size. Lastly, an auxiliary study is performed using the the learned embeddings, discussed in Section 2.4, to predict what slices are appropriate for an arrangement.

3.4.1 Experimental Setup

For our real world experimental setup, we have a single Franka Emika Panda Arm mounted on a Vention frame with a cutting board in the center and another cutting board to the side that we use for grasping objects, as shown in Figure 3.6a. An overhead Microsoft Azure Kinect Camera is mounted to the Vention frame.

Our approach on determining whether a placement will be stable or not, described in Section 3.3.2, makes predictions according to the expected landing location of the object coming out of the grippers. To try to reduce the variation that would be produced by improper grasping, Our grippers for the real robot, which we discussed in Section 3.3.2 were 3D printed using PLA. Figure 3.5 shows an image of the grippers. The spatula finger, shown as the gray portion, moves according to the robot finger's movements, while the pushing finger, shown as the black portion, is fixed to the robot's wrist and does not move when the spatula finger does. To improve the grippers ability to grasp objects, we include slots for 28 gauge steel sheet metal on the grippers to help scrape the objects off of the cutting board.

3.4.2 Experimental Results

For the PoE model discussed in Section 3.3.1, we investigate the behavior of the model when changing the number of experts, the training sample size, and adjusting the sample weights. These effects are shown in Figure 3.7, Figure 3.9, and Figure 3.8, respectively. Note that the arrangements shown in Figure 3.7 and Figure 3.9 where not performed on a real robot. Instead they were created synthetically, by pasting images of the objects on the predicted placement locations exactly. This is done as a means to evaluate the effects of the changes on the PoE model irrespective of any possible placement errors.

Figure 3.7 shows the effect of including more experts conditioned on additional values of n , which we discussed in Section 3.3.1. Note that for the results shown in Figure 3.7, each of the arrangements used models that had the same sample weighting. From these results, we found that the final arrangement becomes more compressed when including additional values of n and not adjusting the weighting hyper-parameters. This behavior may be caused by the additional weighting parameters introduced by the additional experts. Since including additional experts requires a higher degree of hyper-parameter tuning with each expert, we found that using 2 to 3 additional values of n was the most ideal for our tasks.

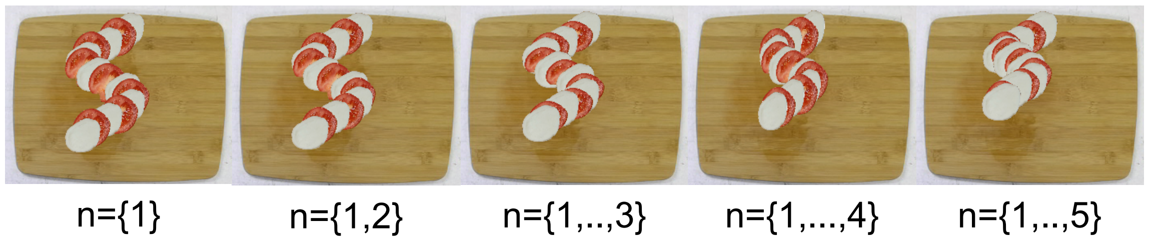


Figure 3.7: Arrangement results of the Product of Experts model described in Section 3.3.1 when using additional experts controlled by the value of n. We use the same weighting hyperparameters for each of the models shown.

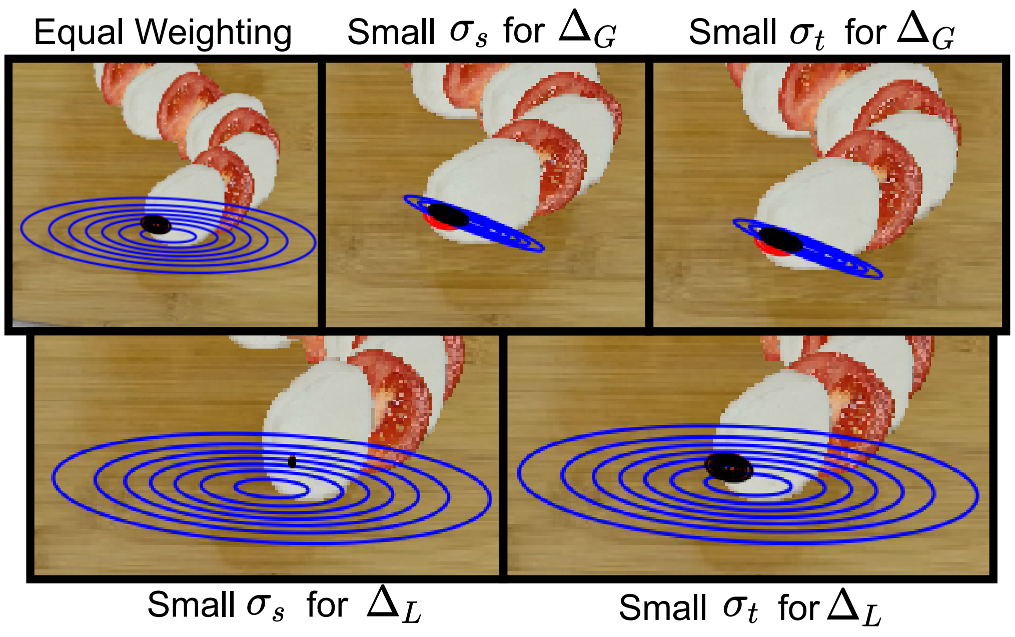


Figure 3.8: Behavior of the gaussian experts using different length scales in Eq. 3.6. The blue and red lines represent the experts conditioned on Δ_G and Δ_L, respectively. The product is shown as the black lines. All of these models use n=1.

Similarly, Figure 3.8 shows the effects that changing the length scale parameters of the sample weighting function, Eq. 3.6, has on each of the gaussian experts. We can see that this weighting approach allows for significant flexibility during the placement tasks. For the Δ_G of our models, we typically set the σ_s to be long (irrelevant) and σ_t to be small (relevant). Hence, the model tends to fit the Gaussians over all of the samples from the same time step. For the Δ_L of our models, we typically set the σ_s to be smaller so it is more relevant and σ_t to be a bit larger so it is not as relevant, since the time step and global location are both relevant.

Additionally, Figure 3.9 shows the effect that sample size has on our PoE model. The figure shows arrangements of each of the three patterns we are evaluating on, as well as an example of what the training data may look like. Note that the exact examples in the figure are not necessarily used for training these specific models. We train 4 different

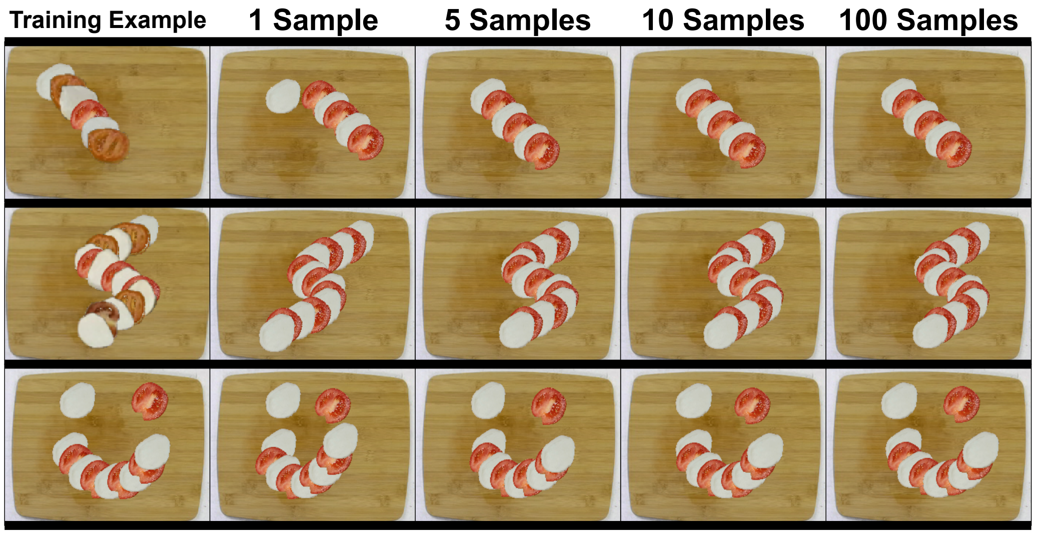


Figure 3.9: Arrangement results of the Product of Experts model described in Section 3.3.1 when trained on different sample sizes. We use the same weighting hyperparameters for each of the models. A random initial position is used for each individual pattern type.

PoE models for each of the patterns, each of which are trained using different sample sizes. Specifically, the models are trained on samples sizes of 1, 5, 10, and 100. To better observe the effect of sample size, we use the same weighting hyper-parameters for each of the models. Additionally, we choose a random initial placement location to test for robustness. From the figure, we can see that the models trained on only 1 sample tend to either have a break in the arrangement or show some type of distortion, such as compression or stretching. However, the arrangements begin to improve after using only 5 training samples, which demonstrates the strong sample efficiency of our model. Additionally, there is not a significant advantage of using a large number of samples, such as 100.

We also evaluate our binary stability classifier on placements in simulation and real world placements. When tested on the simulation data, our binary stability classifier had an average validation accuracy of 77.8%. Note that the dataset the network was trained on, was not balanced in terms of the number of samples with label values of 1 versus 0. Thus, we also calculated a weighted accuracy which weighs the accuracy batchwise to account for this disparity. This balanced average accuracy was 66.1%. Additionally, we evaluated our classifier on 70 real world sample placements, where it achieved an accuracy of 71.4%. These results demonstrate that our method was able to successfully learn information about a placing surface through visual depth information and transfer some of this knowledge to a real world task. However, the network did fail to classify over 20% of the validation samples in both simulation and real world, which may be due to a number of reasons. For the simulation, we found that outliers sometimes occurred when the simulated contacts between objects behaved unexpectedly, for example object penetration between multiple objects. For both simulation and the real world experiments, there may have been a plateau caused by the amount of information that could be extracted from a single view depth image. Additionally, the dynamics of the objects in simulation may not have been representative enough of the real world object behaviors, such as the moisture present in

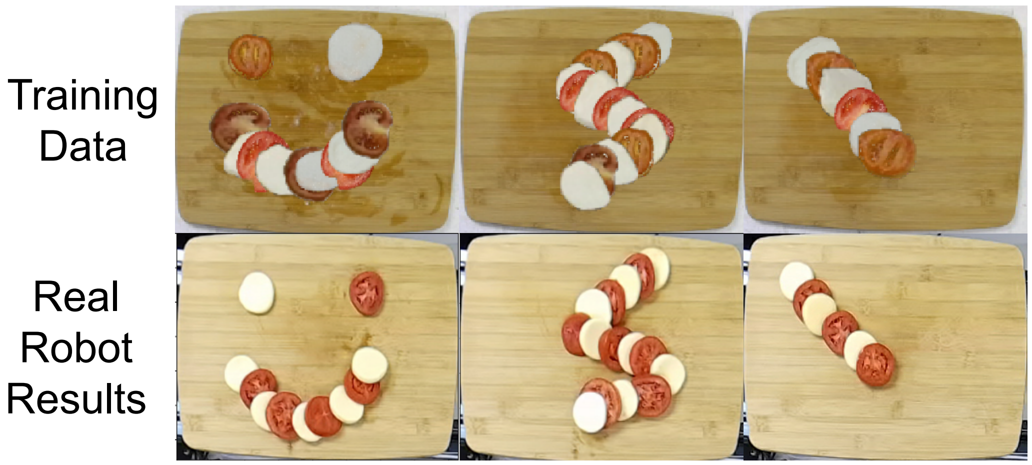


Figure 3.10: Our real world arrangement results when using our overall pipeline compared to the training images. The placement predictions for the real robot results shown used our local weighted PoE models described in 3.3.1.

the tomato and cheese slices. Since the real world experiments occurred over an extended period of time, the properties of the slices of tomatoes and cheese changed over time to be dryer. Even though we did replace slices when we believed them to be compromised, continual manipulation over time did negatively impact their structural integrity.

To assess how our overall approach would perform on a real world plating task, we have a robot generate 3 different Caprese salad arrangements. Figure 3.10 shows these results along with examples of the training data used. The PoE model used for these results contained 5 experts: 1 Δ_G expert, 2 $\Delta_L^{(e)}$ experts, and 2 $\Delta_L^{(c)}$ experts. Note that we used $n = \{1, 2\}$ for these models, which is why $\Delta_L^{(e)}$ and $\Delta_L^{(c)}$ each have two experts. The robot was able to successfully generate the three arrangements through demonstrations using our approach.

We will also breifly discuss the performance of the RNN based approach for predicting placement locations. We trained networks using 10, 100, and 1000 samples for the three different patterns we are evaluating for. We use the root mean square error (RSME) between the predicted placement location and the ground truth location as a measure of accuracy. For the forward slash pattern, the network trained on 1000 samples was able to achieve a RSME value slightly above 2cm, while the networks trained on 10 and 100 samples were not able to achieve a RSME of 3cm. Similarly, none of the networks trained to make predictions on the letter S or smiley face pattern were able to achieve RSME values of under 3cm. A number of variables may have been at play in regard to its poor performance. For example, the architecture/loss function used for training may not be sophisticated enough for the task, the resolution of the training images may hinder higher precision predictions, or the training data may not be large or diverse enough. Due to these results, we deemed our current iteration of the approach to not be accurate enough for our placing task, which requires a higher, sub-centimeter degree of precision.

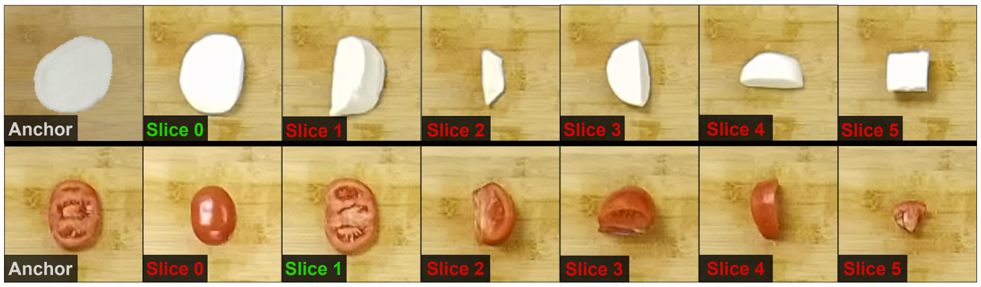


Figure 3.11: Slices of tomatoes and mozzarella cheese used for the classification task of determining which slices would be appropriate for a Caprese salad arrangement. The evaluation task uses the learned embeddings described in Section 2.4 and is described in Section 3.4.2. Table 3.1 shows the results of the task. The anchor images are examples of slices we would use for an arrangement and the slices highlighted with green text are a similar samples would also be used.

Learned Embedding Type	**Food Type**	**Slice Ranking (Best → Worst)**					
Slice Type ($\mathcal{S}$)	Cheese	1	**0**	4	3	5	2
	Tomato	**1**	4	2	0	5	3
Playing Audio ($\mathcal{A}_{play}$)	Cheese	2	5	**0**	4	1	3
	Tomato	**1**	0	3	4	5	2

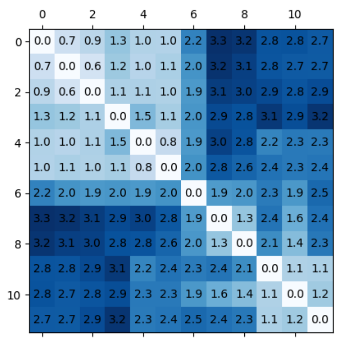
Table 3.1: Results of classifying which food slices should be used for an arrangement. The numbers refer to the slices shown in Figure 3.11. The values furthest to the left are predicted to be the best slices for an arrangement and the values that are in bold text are the correct choices.

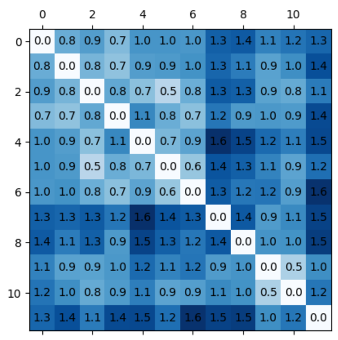
Auxiliary Study: Determining Appropriate Slices for an Arrangement

An auxiliary study is performed where the learned embedding networks, described in Section 2.4, are used as a basis for determining which slices are appropriate for a plating arrangement. Images of multiple slice choices, as well as images of an anchor slice that we deemed to be appropriate for an arrangement, are each passed through the learned embedding networks. Figure 3.11 shows the images of the anchors and the possible choices of both mozzarella and tomato slices that we use for this evaluation. Note the slice numbers on the figure are not directly associated with the slice type numbers discussed in Section 2.3.3. The slice that we deemed to be the most appropriate, given the respective anchor images, are shown with green text (slice 0 for cheese and slice 1 for tomato). We calculate the distance between the learned embeddings of each of the slice choices and their respective anchors. The slices are ranked from best to worst by choosing the slice that is closest to the anchor in the learned embedding space as the best and the slice that is furthest as the worst. Subsequently, the slice best slice is predicted to be the most appropriate slice for the arrangements. Table 3.1 shows these rankings for embeddings learned from the networks trained on slice type and playing audio as the differentiating metric for choosing triplets. Please refer to Section 2.4 for how we use these metrics for choosing triplets.

From the results shown in table, we can see that both of the learned embedding types are successfully able to determine which tomato is the appropriate slice choice, but neither of the learned embedding types are able to correctly classify the correct cheese slice to use. The embeddings learned using slice type, which ranked the correct cheese slice (slice 0) as second, performed better than the embeddings learned using audio, which ranked the correct cheese slice as third. This is reasonable, since the embeddings learned using slice type should inherently perform better on a task dealing with choosing slices. Even though the correct cheese slices were not chosen, it is important to note that the dataset that is used to train these embedding networks, described in Section 2.3, does not contain any relevant images of the specific type of mozzarella cheese used for this study (fresh, high-moisture mozzarella). Instead, the dataset contains only images of low-moisture block mozzarella cheese, which can be significantly different on a visual and material property level. Images of the mozzarella cheese used in the dataset can be seen in Figure 2.5. Conversely, the dataset does contain relevant images of tomato slices, which may be the cause for the superior performance when choosing tomato slices. Given this information, and the fact that the learned embeddings were still able to rank the correct slices in the top 3 without any prior relevant information, there is justification in deeming that the embeddings will perform better on the cheese ranking if the dataset were to contain images of the same type of mozzarella cheese.

Additionally, we present the distance matrices of the learned embeddings in Figure 3.12. Specifically, the embeddings learned using the networks that use food type ($\mathcal{F}$), playing audio ($\mathcal{A}_{play}$), and slice type ($\mathcal{S}$) as the differentiating metrics for choosing triplets. The items in the rows and columns are numbered 0 to 11, where 0-5 are the cheese slices in Figure 3.11 in the same order and 6-11 are tomato slices in the same order (6 is tomato slice 0, 7 is tomato slice 1, etc.). Please note that the anchor slices are not included in the distance matrices. Cells of a lighter color indicate embeddings that are close to one another in the learned embedding space. One interesting thing to note is that there is a clear separation between the cheese and tomato embeddings in Figure 3.12a, which is reasonable given that the network used to produce the embeddings is trained using food type as a metric.


(a) Food Type


(b) Audio

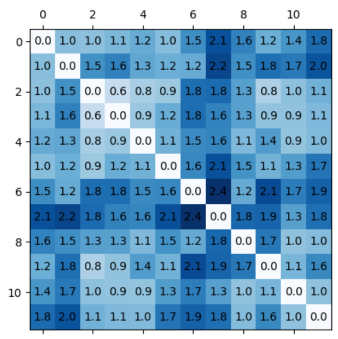

(c) Slice Type

Figure 3.12: Distance matrices between the learned embeddings of the slices shown in Fig. 3.11. Fig. 3.12a shows the embeddings learned using food type, Fig. 3.12b shows the embeddings learned using audio data, and Fig. 3.12c shows the embeddings using slice type. Items 0-5 and 6-11 are the cheese and tomato slices, respectively, in Fig. 3.11.

Chapter 4

Conclusion

In this thesis, we explored research problems involving the difficulties in representing food items due to their complex material property behavior, as well as using those food items for the robot manipulation task of learning to accurately recreate arrangements. In Chapter 2, we addressed this first issue by proposing a method to learn visual embedding networks that encode properties of food items by utilizing multimodal sensory data during training. It was shown that the embeddings learned from these networks encode similarities between food types without the need for explicit human labels. We also observed that these learned embeddings outperformed baselines methods, which use only visual or audio data, on a variety of manipulation-relevant learning tasks. Additionally, we presented a novel dataset consisting of autonomously collected audio, vision, proprioceptive, and force data that was recorded while a robot interacted with a variety of slices from 21 unique food types of differing shapes and properties. In the future, we hope to add further diversity to our dataset, as well as utilize our multi-modal sensory approach for improving other food manipulation-relevant tasks.

In Chapter 3, we addressed the second issue by presenting a method to accurately arrange objects by first learning a local Product of Experts model for deciding where to make placements, and then optimizing these placement decisions by training a neural network to perform binary classification on whether a particular placement would be stable or lead to a misplacement. Our Product of Experts model demonstrated strong sample efficiency, adaptability to errors, and an ability to incorporate additional information through weighting of the training samples and incorporating additional experts. We also evaluate our binary classification network in simulation and found that this approach was able to transfer information to the real world, even though only depth images were used as input. Additionally, we demonstrated that this overall approach can be successfully applied to a real world plating task of generating Caprese salad arrangements. In the future, we hope to extend our Product of Experts approach to generate more complex arrangements and also improve the performance of the binary classification network by either using a more advanced learning technique or incorporating other forms of information, such as our learned food item representations. We showed that our learned food item representation could be used for determining appropriate slices for an arrangement. Building off of this, in the future we can explore the utilization of our learning embeddings to better predict how food items will behave during robotic placing.

Bibliography

[1] Ludger Figura and Arthur A Teixeira. *Food physics: physical properties-measurement and applications*. Springer Science & Business Media, 2007. 1

[2] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research*, 22(30):1–82, 2021. 1, 2.2

[3] 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.

[4] 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. 1, 2.2

[5] Amrita Sawhney, Steven Lee, Kevin Zhang, Manuela Veloso, and Oliver Kroemer. Playing with food: Learning food item representations through interactive exploration, 2021. 2.1

[6] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 2.2

[7] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[8] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016. 2.2

[9] Mohit Sharma, Kevin Zhang, and Oliver Kroemer. Learning semantic embedding spaces for slicing vegetables, 2019. 2.2

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 2.2

[11] Phillip Isola, Joseph J. Lim, and Edward H. Adelson. Discovering states and transformations in image collections. In *CVPR*, 2015. 2.2

[12] Jan Matas, Stephen James, and Andrew J. Davison. Sim-to-real reinforcement learning for deformable object manipulation, 2018. 2.2

[13] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning

to manipulate deformable objects without demonstrations, 2020. 2.2

[14] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation, 2020. 2.2

[15] Carolyn Matl, Yashraj Narang, Ruzena Bajcsy, Fabio Ramos, and Dieter Fox. Inferring the material properties of granular media for robotic tasks, 2020. 2.2

[16] Nuttapong Chentanez, Matthias Müller, and Miles Macklin. Real-time simulation of large elasto-plastic deformation with shape matching. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 159–167, 2016. 2.2

[17] 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. 2.2

[18] 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. 2.2

[19] Zackory Erickson, Eliot Xing, Bharat Srirangam, Sonia Chernova, and Charles C. Kemp. Multimodal material classification for robots using spectroscopy and high resolution texture imaging, 2020. 2.2

[20] Ryan Feng, Youngsun Kim, Gilwoo Lee, Ethan K. Gordon, Matt Schmittle, Shivaum Kumar, Tapomayukh Bhattacharjee, and Siddhartha S. Srinivasa. Robot-assisted feeding: Generalizing skewering strategies across food items on a realistic plate, 2019. 2.2

[21] Gyan Tatiya and Jivko Sinapov. Deep multi-sensory object category recognition using interactive behavioral exploration. In *ICRA*, pages 7872–7878. IEEE, 2019. 2.2

[22] K. Zhang, M. Sharma, M. Veloso, and O. Kroemer. Leveraging multimodal haptic sensory data for robust cutting. In *Humanoids*, pages 409–416, 2019. 2.2, 2.3.2, 2.3.3

[23] Shan Luo, Wenzhen Yuan, Edward Adelson, Anthony G Cohn, and Raul Fuentes. Vitac: Feature sharing between vision and tactile sensing for cloth texture recognition. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2722–2727. IEEE, 2018. 2.2

[24] Wenzhen Yuan, Shaoxiong Wang, Siyuan Dong, and Edward Adelson. Connecting look and feel: Associating the visual and tactile properties of physical materials. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5580–5588, 2017. 2.2

[25] Wenzhen Yuan, Yuchen Mo, Shaoxiong Wang, and Edward H Adelson. Active clothing material perception using tactile sensing and deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4842–4849. IEEE, 2018. 2.2

[26] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017. 2.2

[27] Dov Katz and Oliver Brock. Manipulating articulated objects with interactive per-

ception. In *2008 IEEE International Conference on Robotics and Automation*, pages 272–277. IEEE, 2008. 2.2

[28] 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. 2.2

[29] 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. 2.2

[30] Vivian Chu, Ian McMahon, Lorenzo Riano, Craig G McDonald, Qin He, Jorge Martinez Perez-Tejada, Michael Arrigo, Naomi Fitter, John C Nappo, Trevor Darrell, et al. Using robotic exploratory procedures to learn the meaning of haptic adjectives. In *2013 IEEE International Conference on Robotics and Automation*, pages 3048–3055. IEEE, 2013. 2.2

[31] 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. 2.3.1

[32] 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. 2.3.1

[33] Kevin Zhang. https://github.com/firephinx/sounddevice_ros. 2.3.1

[34] 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. 2.3.2

[35] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. In *Robotics research. the eleventh international symposium*, pages 561–572. Springer, 2005. 2.3.2

[36] K.K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018. 2.3.2

[37] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017. 2.3.2

[38] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, pages 633–641, 2017. 2.3.2

[39] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, pages 1–11, 2000. 2.3.3

[40] 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. 2.3.3

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2.4, 3.3.1

[42] 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. 2.4

[43] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 2.4

[44] D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242, 1958. ISSN 00359246. 2.5

[45] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 348–353. IEEE, 2000. 3.2

[46] Mark R Cutkosky. *Robotic grasping and fine manipulation*, volume 6. Springer Science & Business Media, 2012.

[47] Ashutosh Saxena, Lawson LS Wong, and Andrew Y Ng. Learning grasp strategies with partial shape information. 2008.

[48] A Gafer, D Heymans, D Prattichizzo, and G Salvietti. The quad-spatula gripper: A novel soft-rigid gripper for food handling. In *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 39–45. IEEE, 2020.

[49] Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H Adelson, and Sergey Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *IEEE Robotics and Automation Letters*, 3(4):3300–3307, 2018. 3.2

[50] Zhongkui Wang, Mingzhu Zhu, Sadao Kawamura, and Shinichi Hirai. Fabrication and performance comparison of different soft pneumatic actuators for lunch box packaging. In *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 22–27. IEEE, 2017. 3.2

[51] A Pettersson, Thomas Ohlsson, S Davis, JO Gray, and TJ Dodd. A hygienically designed force gripper for flexible handling of variable and easily damaged natural food products. *Innovative Food Science & Emerging Technologies*, 12(3):344–351, 2011.

[52] Gen Endo and Nobuhiro Otomo. Development of a food handling gripper considering an appetizing presentation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4901–4906. IEEE, 2016.

[53] Ping Yong Chua, T Ilschner, and Darwin G Caldwell. Robotic manipulation of food products–a review. *Industrial Robot: An International Journal*, 2003. 3.2

[54] Aaron Edsinger and Charles C Kemp. Manipulation in human environments. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 102–109. IEEE, 2006. 3.2

[55] Charles C Kemp, Aaron Edsinger, and Eduardo Torres-Jara. Challenges for robot manipulation in human environments [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):20–29, 2007.

[56] Yun Jiang, Changxi Zheng, Marcus Lim, and Ashutosh Saxena. Learning to place new objects. In *2012 IEEE International Conference on Robotics and Automation*, pages 3088–3095. IEEE, 2012. 3.2

[57] M. J. Schuster, J. Okerman, H. Nguyen, J. M. Rehg, and C. C. Kemp. Perceiving clutter and surfaces for object placement in indoor environments. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 152–159, 2010. doi: 10.1109/ICHR.2010.5686328. 3.2

[58] Yun Jiang, Marcus Lim, Changxi Zheng, and Ashutosh Saxena. Learning to place new objects in a scene. *The International Journal of Robotics Research*, 31(9):1021–1043, 2012. 3.2

[59] Yun Jiang, Marcus Lim, and Ashutosh Saxena. Learning object arrangements in 3d scenes using human context. *arXiv preprint arXiv:1206.6462*, 2012. 3.2

[60] Yun Jiang and Ashutosh Saxena. Hallucinating humans for learning robotic placement of objects. In *Experimental Robotics*, pages 921–937. Springer, 2013. 3.2

[61] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics (TOG)*, 31(6):1–11, 2012. 3.2

[62] Dominik Jain, Lorenz Mosenlechner, and Michael Beetz. Equipping robot control programs with first-order probabilistic reasoning capabilities. In *2009 IEEE International Conference on Robotics and Automation*, pages 3626–3631. IEEE, 2009. 3.2

[63] Erdi Aker, Ahmetcan Erdogan, Esra Erdem, and Volkan Patoglu. Housekeeping with multiple autonomous robots: Knowledge representation and automated reasoning for a tightly integrated robot control architecture. In *Knowledge Representation Workshop at IROS 2011*, 2011.

[64] Lorenz Mösenlechner and Michael Beetz. Parameterizing actions to have the appropriate effects. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4141–4147. IEEE, 2011. 3.2

[65] Hiroshi Sugie, Yasuhiro Inagaki, Shuchir Ono, Hidyuki Aisu, and Tatsuo Unemi. Placing objects with multiple mobile robots-mutual help using intention inference. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 2, pages 2181–2186. IEEE, 1995. 3.2

[66] Tomás Lozano-Pérez, Joseph L. Jones, Emmanuel Mazer, and Patrick A. O'Donnell. Task-level planning of pick-and-place robot motions. *Computer*, 22(3):21–29, 1989. 3.2

[67] Yunduan Cui, Jun'ichiro Ooga, Akihito Ogawa, and Takamitsu Matsubara. Probabilistic active filtering with gaussian processes for occluded object search in clutter. *Applied Intelligence*, 50(12):4310–4324, 2020. 3.2

[68] Ryota Moriyama, Weiwei Wan, and Kensuke Harada. Dual-arm assembly planning considering gravitational constraints. *arXiv preprint arXiv:1903.00646*, 2019. 3.2

[69] Junki Matsuoka, Yoshihisa Tsurumine, Yuhwan Kwon, Takamitsu Matsubara, Takeshi Shimmura, and Sadao Kawamura. Learning food-arrangement policies from

raw images with generative adversarial imitation learning. In *2020 17th International Conference on Ubiquitous Robots (UR)*, pages 93–98. IEEE, 2020. 3.2

[70] Troels Bo Jørgensen, Sebastian Hoppe Nesgaard Jensen, Henrik Aanæs, Niels Worsøe Hansen, and Norbert Krüger. An adaptive robotic system for doing pick and place operations with deformable objects. *Journal of Intelligent & Robotic Systems*, 94(1): 81–100, 2019. 3.2

[71] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002. 3.3.1

[72] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3.3.1

[73] Jialiang Zhao, Jacky Liang, and Oliver Kroemer. Towards precise robotic grasping by probabilistic post-grasp displacement estimation, 2019. 3.3.2

[74] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. 3.3.2

[75] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009. 3.3.2

[76] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018. 3.3.3

[77] Jacky Liang, Viktor Makoviychuk, Ankur Handa, Nuttapong Chentanez, Miles Macklin, and Dieter Fox. Gpu-accelerated robotic simulation for distributed reinforcement learning. *Conference on Robot Learning*, 2018. 3.3.3