Independent Study in Robotics: Researching Skill Models for Autonomous Food Preparation

Ganesh Iyer
giyer@andrew.cmu.edu

1 Introduction

In the recent years, we have seen great progress in learning-based methods for deformable object manipulation. Various works like [1], [2] have showcased that large-scale interaction sequences with objects can lead to learning robust downstream skills. These skills can then be used to compose skill libraries, which are then combined with planning based approaches to solve long-horizon tasks. However, one area where such data-driven methods continue to be a challenge to develop is food preparation. Based on the various physical properties of the food object (adhesion, damping, poisson coefficient, plasticity, stress and strain properties, viscosity) it may react and deform differently to the same action. For example, lets take the simple task of pushing a food object. For the same control scheme, actions, and forces, a cut potato may deform and displace very differently as compared to a slice of cucumber. Further, one of the areas we specifically considered was dough manipulation. Dough is especially difficult to model due to its plastic nature which is not only inhibited in other food objects, but whose properties are different from similar plastic objects like clay. In this work we address the task of building a large-scale dataset for interacting with food objects of different properties in order to extend learning-based methods to food preparation skills. Further, due to the current circumstances of the COVID-19, we explore this task in two phases: 1. Using Physical hardware 2. In Simulation.

2 Related Work

Learning-based methods have been successfully applied to robot manipulation [3] for various tasks. With the advent of Deep Learning, large-scale data collection with progress in Deep Reinforcement Learning and Imitation Learning has further pushed the envelope in robot manipulation [4], [5]. A similar progress has been seen in some interaction with some deformable objects like cloth and rope [6], [7], [8]. However similar methods cannot directly be used when interacting with food objects for long horizon food preparation in the kitchen. Model-based learning has made great progress in longer horizon tasks [1], [9], however to apply such methods in the kitchen it is very important to find accurate embeddings for the various possible food objects with limited interactions. Sharma et. al in [10] showed how latent embeddings learnt to solve auxillary tasks can be used for achieving the task of slicing vegetables. This work is a step in the same direction, but leverages on larger collection of data streams to learn implicit representations of all kinds of food objects such as vegetables, meat, and dough. This study also covers methods to interact with deformable objects in simulation. Similar to hardware based methods, sim-to-real transfer for deformable object manipulation is an active area of work [11], [12], [13], [14], [12]. However, this has also not been explored in the context of general deformable manipulation using model-based learning methods. Using such a learning paradigm may allow us to transfer such tasks to generalized deformable manipulation, and in particular to food like dough and vegetables. This aspect is also explored in the study.

3 Method

3.1 Interaction sequences

First, we consider the task of large-scale interaction of food objects with physical hardware. The setup used in the lab is shown in Figure 1. To focus on the multi-modal collection of interaction data, the setup consisted of various extrinsically calibrated cameras capturing depth and RGB frames. All

Figure 1: Caption (a) Left: Franka Panda Robot, fitted with custom end-effectors are used to interact differently with various food objects. Azure Kinect setup facing the chopping board is used as the main overhead camera. (b) Right: Finger-vision camera setup, as seen on the end-effectors. Realsense cameras mounted at different points to capture depth information.

the frames were synchronized based on the skill being executed. Before we discuss, the skill it can be useful to discuss the food objects being considered. We initially focused on vegetable slices, and modeled the interaction of i) cucumber ii) potato iii) carrot iv) steak meat (cooked and uncooked). Each of these objects were further classified based on the slice of the cut (thickness, varied shapes etc.). This was important to capture the physical properties of the food object. For example, we know from experience that a thicker cucumber slice is prone to supporting itself on its thicker edge as opposed to a thinner slice, which will often fall to its side. This was often visible even at the time of interaction with the slice. To reduce the space of possible interactions, we used the YOLOv3 object detection network [15]. The network takes the overhead Azure Kinect RGB frames as input and predicts bounding boxes for the detected food objects. This network was initially trained in a supervised manner on similar frames with food object slices. Based on the the bounding box locations, action could be targeted specifically at the global coordinates of the bounding-boxes.

For the interactions themselves, we considered the following actions. i) Pushing (or pulling) the food object by dragging: For this action, a point on the chopping board is randomly sampled and the closest slice is pushed or pulled by a fixed distance on the board. We specifically tried to capture the adhesive properties through this action. Stickier food substances like cucumber and steak slices would be dragged to smaller distances as opposed to potato and carrot. ii) Picking and Dropping: For this action, one of the food objects is sampled randomly and the robot applies a fixed closed-gripper action to lift the object to a fixed height (default height: 0.1m) and drop it. This action is observed in the finger-vision cameras and the overhead camera frames. One of the key insights here is to capture the deformed shape of the object in the finger-vision setup when grasping. The captured frames would indicate rigidity of the food object. (The lesser the rigidity, the more the deformation, and therefore more of the frame is occluded by the vegetable). Missed grasps (due to slippage) are also considered valuable information. Apart from frame information, we also collected sound information using vibration mics attached to the bottom of the chopping board and end effector, although this wasn't analyzed in as much detail as the RGB and depth frames. The interaction could also be randomized using action states to further generalize the collected sequences.

We were considering the development of a large-scale dataset based on the previously explained primitve skills. One of the problems we noticed was that vegetables would often behave differently based on temperature. For example, colder vegetable slices were noticed to be less adhesive than warm slices, due to lesser surface fluid. To prevent this, we considered an ice-box setup that would allow more interactions while keeping food slices fresher for longer durations. Unfortunately, due to COVID-19 restrictions, we were unable to collect this large-scale dataset.

(a) (b)

Figure 2: a) Left: Example of dataset collection for training object detection network b) Simple manipulation on potato slice, as seen in fingervision and alternate cameras

3.2 NVIDIA FleX Simulation

Nvidia FleX [16] is a powerful particle-based simulation library. The core idea behind FleX is the use of particle dynamics to simulate materials and bodies of various physical properties [17]. Some of the examples in the library include modeling the coupling of fluids and solid mesh objects, modeling the viscosity of fluids, and further introduction of Finite Element Materials. This library is ideal for simulating deformable objects and interacting with them using robot models. Broadly, creating a simulation setup involves the creation of a **`Scene`**, which itself consists of a set of callback functions that define a general template: `Initialize`, `Update`, `DoStats/DoGui`, `Draw` etc. Each of these helper functions have to be overridden based on the requirements of the scene. For example, static rigid objects have to modeled separately in 'child Environments' with each function in the template indicating the updated positions of the various particles or rigid bodies in the scene.

Solvers
Before we discuss object modeling, it is important to discuss the choice of solvers, since this critically impacts the scene design choices. Broadly, two kinds of solvers were available for immediate use 1. Position Based Dynamics Solver (PBD) [18], that define particle dynamics and constraints between particles that are resolved using a non-linear optimization method. 2 . Parallel Cyclic Reduction Solvers [19] that solve large linear systems (specifically tridiagonal systems) using GPU based parallel implementations.

Object Modeling
In order to model dough, we considered two types of particle deformation models available in the library that could be used for interaction:
1.**`SoftBodies`**: In FleX, Softbody objects are defined as clusters of particles. Each particle has simpler properties like radius, position and velocity. Additionally, the object also consists clusters of these particles, along with cluster properties like cluster-spacing, cluster-radius, stiffness, plasticity threshold, and plastic-creep . These properties define the overall behavior of the object. For example, modifying the creep threshold increases the chance of cluster-sliding on low friction surfaces.

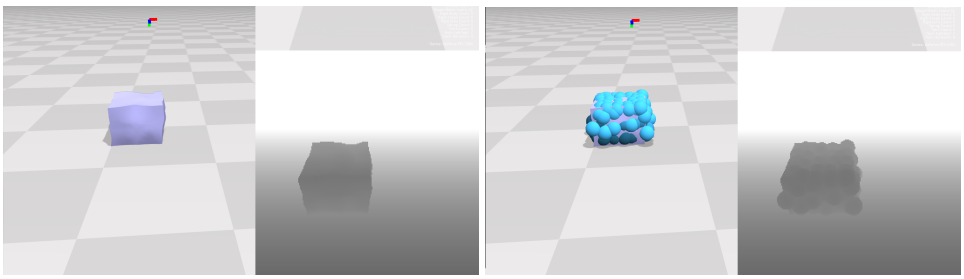

Figure 3: `SoftBody` models, with points, mesh rendering, and depth sensor output

2. **`DeformableMesh`**: DeformableMesh is another soft object model that is defined by hostMesh and tetMesh objects. tetMesh are basically tetrahedral mesh structures where the links and position of particles (as vertices) define the overall properties of the objects. In order to define smooth deformations, affine transformations for each vertex (in this case, particle) are weighted and an energy minimization equation is solved iteratively to find the closest updated positions for the mesh vertices.
While both of these were valid choices to model deformable objects, we decided to proceed with the `DeformableMesh` object. The key reason for this was the solver. SoftBody objects could only be controlled using a PBD Solver, and unfortunately at the current stage, while PBD Solvers could handle particle-based objects well, the solver could not handle currently available rigid URDF-based robot models in the library. Therefore, we had to rely on PCR solvers for robot manipulation tasks in simulation.

Control Scheme
In order to re-enable the skill based interaction as in hardware, we created a similar scene as in the lab. The scene consists of a table-top with a Fetch Robot for manipulation tasks. While the Franka was also available in the library, working with the Fetch in a shorter time frame proved to be much more beneficial. Further, both the Franka and Fetch have similar 7-DoF joint spaces, with the difference only being the joint constraints. At the low level, an inverse dynamics based control is used. We use a planar control method, such that at each time step, the end effector updates its position based on the calculated inverse dynamics, with the end-effector facing downwards parallel to the object. FleX also allows the addition of various sensors in the scene. We use this to replicate the Azure Flex setup as in the lab. Once the scene is created, an openAI gym [20] based episodic setup allows resetting the scene using a python interface for multiple interactions. At the start of every episode, we first spawn a rigid-sphere as a 3D target point for the end-effector. We also spawn the deformable object under this sphere. The end effector has to first reach a rest position above the object, followed by an interaction of choice. We model two types of interactions 1. Pushing down on deformable objects (similar to the act of flattening dough). 2. An additional dragging interaction, similar to the push skill as defined for the hardware setup.

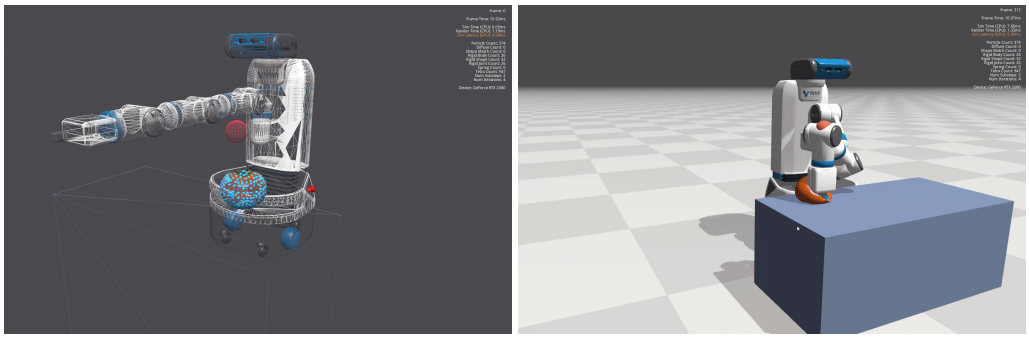

Figure 4: Left: `DeforomableMesh` models with points and tetrahedral wireframe for a tomato mesh object. The Fetch robot and target sphere also indicated. Right: Example of interaction with tomato mesh model. THe End-effector can be seen pushing down on the object with the target sphere indicating the external position.

Data-driven Learning
For both hardware and the simulation setups, the learning framework stays the same. Frames from the rgb and depth sensors are collected before and after a series of skill interactions. Each series of interactions is stored in a replay buffer as a transition of the form (s_t, a_t, s_{t+1}), where the state is defined by the frames taken at that time step. These are then passed into a convolutional encoder-decoder network that predicts the next state given the current state and action similar to model-based reinforcement learning methods. This model can then be used in a variety of ways, such as performing rollouts up to a certain finite horizon, or re-planning using a library of existing skills. Another approach under active consideration is the use of a contrastive divergence loss term to segregate the embeddings in latent space, in order to learn about the various properties of food objects that the robot would interact with. This would allow further generalization to unseen objects within a few interactions. This data would therefore be effective in enabling long-horizon food preparation tasks in the kitchen.

Acknowledgments

In these testing times of COVID-19, Prof. Oliver Kroemer has been tremendously supportive and actively encouraged the work in this study, right from putting up with my meeting lapses to donating one of the systems of the lab for work-from-home. I would like to thank him for his guidance and support. I would also like to thank everyone at IAM-Lab for helping me in getting very quickly acquainted with the robot framework, architecture, and software components. Lastly, I would like to thank Prof. John Dolan for allowing me to participate in this independent study, and for supporting me throughout the MRSD program.

References

[1] T. Kurutach, A. Tamar, G. Yang, S. J. Russell, and P. Abbeel. Learning plannable representations with causal infogan. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8733–8744. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8090-learning-plannable-representations-with-causal-infogan.pdf.

[2] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. *CoRR*, abs/1807.03480, 2018. URL http://dblp.uni-trier.de/db/journals/corr/corr1807.html#abs-1807-03480.

[3] O. Kroemer, S. Niekum, and G. D. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *CoRR*, abs/1907.03146, 2019. URL http://arxiv.org/abs/1907.03146.

[4] A. Mousavian, C. Eppner, and D. Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *International Conference on Computer Vision (ICCV)*, 2019.

[5] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand, 2019.

[6] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation, 2017.

[7] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning, 2017.

[8] R. Jangir, G. Alenya, and C. Torras. Dynamic cloth manipulation with deep reinforcement learning, 2019.

[9] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018.

[10] M. Sharma, K. Zhang, and O. Kroemer. Learning semantic embedding spaces for slicing vegetables, 2019.

[11] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation, 2018.

[12] X. Lin, H. S. Baweja, and D. Held. Reinforcement learning without ground-truth state, 2019.

[13] F. Ruggiero, A. Petit, D. Serra, A. C. Satici, J. Cacace, A. Donaire, F. Ficuciello, L. R. Buonocore, G. A. Fontanelli, V. Lippiello, L. Villani, and B. Siciliano. Nonprehensile Manipulation of Deformable Objects: Achievements and Perspectives from the RobDyMan Project. *IEEE Robotics and Automation Magazine*, 25(3):83 – 92, Sept. 2018. doi: 10.1109/mra.2017.2781306. URL https://hal.inria.fr/hal-01889779.

[14] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience, 2018.

[15] J. Redmon and A. Farhadi. Yolov3: An incremental improvement, 2018.

[16] Nvidia flex. https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/flex/index.html. Accessed: 2010-09-30.

[17] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim. Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4), July 2014. ISSN 0730-0301. doi:10.1145/2601097.2601152. URL https://doi.org/10.1145/2601097.2601152.

[18] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, Apr. 2007. ISSN 1047-3203. doi:10.1016/j.jvcir.2007.01.005. URL https://doi.org/10.1016/j.jvcir.2007.01.005.

[19] Y. Zhang, J. Cohen, and J. D. Owens. Fast tridiagonal solvers on the gpu. In *Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '10, page 127–136, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605588773. doi:10.1145/1693453.1693472. URL https://doi.org/10.1145/1693453.1693472.

[20] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.