Using Tactile Sensors to Learn a Manipulation Task

Mason Llewellyn
Stanford University Computer Science
mason747@stanford.edu

Mentored By
Kaylee Carissa Burns
Stanford University Computer Science (PhD Student)
kayburns@stanford.edu

Abstract

Manipulation tasks in robotics stand to benefit immensely from detailed tactile information. In fine-grained tasks in which the gripper occludes the camera, tactile information can keep the robot appraised of the exact state of its current grasp. We investigate the application of visuo-tactile sensors to a peg insertion task. We use the tactile sensors, in addition to other sensor modalities, to learn a state representation of our peg insertion task. This state representation is then used to train a reinforcement learning agent to solve the peg-insertion task.

1 Introduction

Manipulation in humans benefits from a variety of sensor inputs such as vision, touch, and proprioception. Even simple tasks such as inserting a key into a lock requires a person to use their vision to locate the lock, proprioception to position their hands appropriately, and touch to insert the key smoothly. It is no stretch of the imagination, therefore, to believe that robots could also benefit from a similar combination of these sensor modalities. Recent improvements in sensor, computer, and robotic technologies have made this "human-like" sensor integration possible. As robots move from rigidly controlled environments into unstructured real-world environments, their state representation must be robust against noise and other environmental perturbation which is very difficult with as single sensor. Independent sensor modalities all have their own weaknesses. Visual sensors such as cameras or LIDAR can be occluded and proprioceptive sensors can fall victim to error. Combining these modalities into a single state representation will create a representation that is robust to the errors of any one modality.

The simplest way to create this state representation would be to concatenate readings from all sensor modalities into a single, large feature vector. This state representation would contain all sensor modalities but would lack the ability to encode useful correlations between sensor data. We can use deep learning to parameterize a feature encoder which learns a compact state representation that encodes useful sensor correlations as well as task-specific information. This compact state representation can be passed into a reinforcement learning algorithm, which we do here, to solve a given task.

In addition to vision and proprioception, a good sense of touch is essential for humans to complete manipulation tasks. We can imbue robots with a similar sense of touch using visuo-tactile sensors such as Johnson et al. [2011]. These sensors use a camera beneath a deformable gel to represent whatever the object is touching as an image. When the sensor is touched, the gel deforms and the camera's image changes. Using libraries such as Tacto [Wang et al., 2020], we can simulate visuo-tactile sensors for faster training in simulation.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

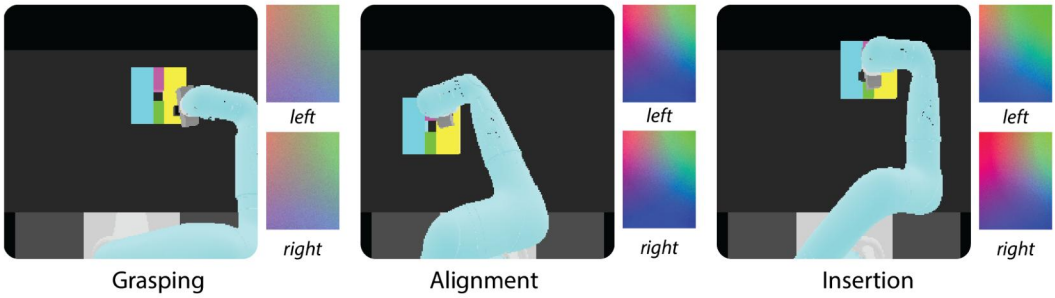

Figure 1: Visualizations of Tacto readings at different stages of the peg insertion task.

In this project, we combine visual, proprioceptive, and tactile sensor modalities to learn a state representation for a peg insertion task parameterized by a neural network. Our objective is to teach a robot arm with a parallel jaw gripper to use the modalities to successfully navigate a peg into an open hole more efficiently and robustly than it would be able to without tactile sensing. Although we expected that this combination of sensor modalities would create a state representation that could train a successful model, we found that our reinforcement learning agent was unable to meaningfully learn its environment.

2 Related Work

Making Sense of Vision and Touch [Lee et al., 2019] has shown the benefits of combining vision, proprioception, and force-sensing into one state representation for use in the peg-insertion task. This merging of vision, proprioception, and force-sensing modalities to complete a task creates a quasi-human policy.

Information from visuo-tactile sensors has been shown to be useful in high-dexterity manipulation tasks such as cable manipulation [Y. She and Adelson, 2020] and USB connector insertion [Li et al., 2014]. Both of these papers use images from Ghttps://www.overleaf.com/project/60ab0b0578a1430051d3c24celSight sensors with heuristic rather than learning-based methods to solve their respective tasks. These types of solutions work well in highly controlled environments such as a lab setting but can be brittle to real-world conditions.

Our project seeks to explore the benefits of adding the tactile modality to the multi-modal state representation in Lee et al. [2019]. We replace the wrist-torque force encoder in their state representation model with our own encoder for data from visuo-tactile sensors. The information from the tactile sensors has the potential to improve the state representation and, by extension, the policy learned in [Lee et al., 2019]. We found, however, that our modifications to the existing encoder were not enough to achieve satisfactory results in our environment.

3 Methods

This project builds directly on Lee et al. [2019] and uses its methods to create a feature encoder that encodes information from visual, proprioceptive, and tactile modalities into a vector of length d (which in this case is 128). The feature encoder is composed of four individual encoders, each paramaterized by a neural network: a color image encoder, a depth image encoder, a tactile encoder, and a proprioception encoder. The color encoder is a convolutional neural network with six convolutional layers and a single linear layer. The color encoder takes in a 224x224 pixel RGB image and outputs a 2 x d vector. The depth encoder is also a convolutional neural network with six convolutional layers and a single linear layer. The depth encoder takes in a 224x224x1 depth image and outputs a 2 x d vector. The proprioceptive encoder is a fully connected neural network with four linear layers each with a leaky RELU activation function with a cutoff of 0.1. The proprioceptive encoder takes in a vector of length 8 featuring the position, velocity, and roll angles of the end effector and outputs a 2 x d vector.

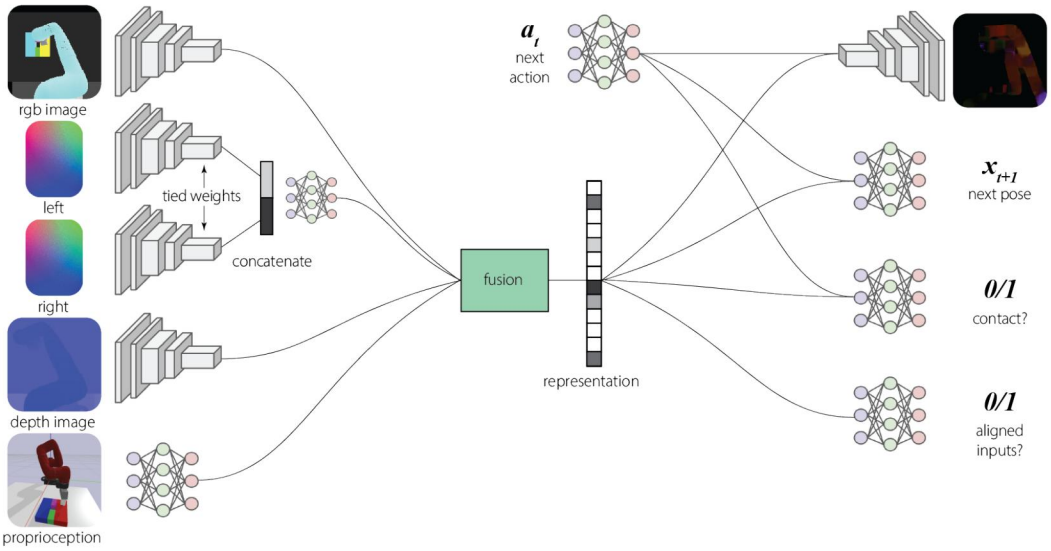

Figure 2: Network architecture for representation learning. Figure and model adapted from Lee et al. [2019]. We replace the force-torque input with readings from Tacto sensors. The Tacto images are passed through convolutional layers, concatenated, and passed through linear layers before being passed to the sensor fusion module.

The Tacto encoder takes in four images, two 120x160x1 depth images and two 120x160x3 RGB color images, from the left and right tactile sensors respectively, and outputs a 2 x d vector. The Tacto encoder is the most complex out of the four encoders. The color images are each encoded by a convolutional neural network with three convolutional layers and one linear layer into two d-length vectors. These vectors are concatenated and multiplied by another linear layer followed by a leaky RELU activation function yielding a d length vector. The two color images are both processed by the same convolutional network to capture the fact that images from both tactile sensors will have similar underlying features. Depending on how the manipulator grasps an object, the images in the left and right sensors can be switched. Using the same convolutional encoder for both images allows us to capture this property of our robot. Despite disregarding orientation within our convolutional encoders, our model uses the post-concatenation linear layers to learn features corresponding to the orientation of the manipulator. The two depth images are processed in a similar fashion, yielding another d length vector. These two vectors are then concatenated and multiplied by two linear layers each followed by a Leaky RELU activation function to output a 2 x d length vector. These four 2 x d vectors are combined into a a single d-dimensional state representation vector using the Product of Experts method as outlined in [Lee et al., 2019].

The state representation vector is trained, using self-supervision, to encode information about its environment. The model predicts the datapoints referenced in Table 2 for each timestep conditioned on the action taken within that timestep.

In addition to the datapoints referenced in Table 2, the self-supervised model must also learn to predict whether its external camera images are time-aligned with its Tacto images and proprioception measurements. This time-alignment prediction addresses the fact that there are redundancies between the different sensor modalities and encourages the model not to disregard one over the oth[2] [Lee et al., 2019]. The learned state representation vector is used as the observation input into a reinforcement learning policy parameterized by a two-layer MLP. After the state representation is learned, the encoder parameters are held constant while the policy is trained using Trust Region Policy Optimization (TRPO) [Schulman et al., 2015].

3.1 Dataset and Features

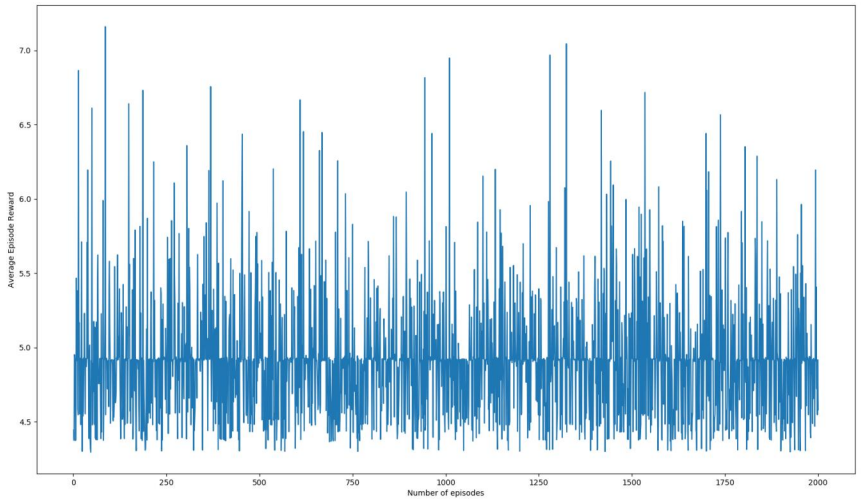

Figure 3: Reward curve from agent trained using state representation.

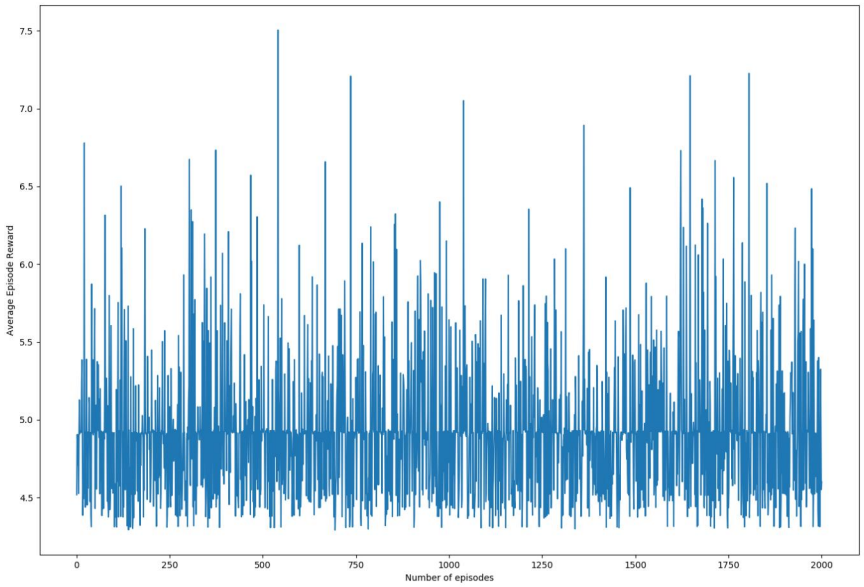

Figure 4: Reward curve from agent trained using state representation without Tacto.

The data is generated from rollouts of a heuristic peg insertion policy and a random peg insertion policy within our simulated environment. Tables 1 and 2 detail the recorded datapoints along with their dimensions. These datapoints are recorded once at every step of the policy. The Tacto color and depth images each consist of two images, one for the left and right fingers of the gripper respectively making four Tacto images in total.

Data Point	Size
RGB Color Image	224x224x3
Depth Image	224x224x1
Tacto Color Images	2x120x160x3
Tacto Depth Images	2x120x160x1
Proprioception	8

Table 1: Inputs to State Representation Encoder

Data Point	Size
Optical Flow	224x224x2
Action	3
Contact Next (boolean)	1
Next end-effector position	3

Table 2: Datapoints for self-supervised training

Before being passed into the encoder, all color images are scaled between zero and one (by dividing each individual value by 255). The pixels of depth images are filtered to fit in the interval $(10^{-7}, 2)$ with any values outside this interval set to zero.

4 Experiments

4.1 Simulated Environment

Both our data collection and our experiments are completed in a PyBullet simulated environment. The simulated robot is a Rethink Sawyer with a WSG50 gripper. Embedded into the fingers of the gripper are two visuo-tactile sensors, called Tacto sesors, which are simulated using the Tacto library [Wang et al., 2020]. The simulation is packaged by Perls2, a robotics framework [Kulkarni et al., 2020], into a repeatable environment for reinforcement-learning experiments. The environment contains the robot, a peg, as well as a box with a hole into which the peg is to be inserted. The environment is initialized with the box at a random position and the peg at a random position at the top of the box. At each timestep, the agent must take an action which determines the motion of the robot arm. In our case, the action specifies a delta to move the end effector in Cartesian space $[\Delta x, \Delta y, \Delta z]$. Once the robot completes the peg insertion task or fails at completing the task within given time, the environment is reset and the peg and box positions are randomly reinitialized.

4.2 Results

We trained the state representation encoder on our dataset of examples from both random and heuristic policies for 50 epochs before training the RL agent to solve the peg-insertion task on our state representation. Contrary to our expectation, the agent did not appear to learn much, if anything, over the course of training.

5 Conclusion

Though our results were disappointing, it has informed us that the reinforcement learning problem is more difficult than we initially estimated. Our current hypothesis as to why our reinforcement learning agent failed to learn is that the existing model that we modified from [Lee et al., 2019] was not tuned to work with our data or our environment. In the future, we plan to rebuild our own self-supervised state representation from first-principles rather than simply try to retrofit an existing state representation. This will also entail re-evaluating the data we collected and perhaps collecting new data. Despite the results not being favorable, this serves as a suitable starting point for our future research.

6 Contributions

Mason Llewellyn worked on this project under the guidance of Kaylee Burns. This is a part of a larger research project which investigates learning multimodal state representations for robot manipulation tasks.

7 Code

The code for this project can be found within two GitHub repositories a repository for the multimodal state representation encoder and a repository for the peg insertion task environment.

References

Micah K. Johnson, Forrester Cole, Alvin Raj, and Edward H. Adelson. Microgeometry capture using an elastomeric sensor. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 30(4):46:1–46:8, 2011. doi: http://dx.doi.org/10.1145/2010324.1964941.

Rohun Kulkarni, Daniel Xu, Marcus DK, and Lyron Co Ting Keh. Perception and robotic learning system v2. 2020.

Michelle A Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019. URL `https://arxiv.org/abs/1810.10191`.

Rui Li, Robert Platt, Wenzhen Yuan, Andreas ten Pas, Nathan Roscup, Mandayam A. Srinivasan, and Edward Adelson. Localization and manipulation of small parts using gelsight tactile sensing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3988–3993, 2014. doi: 10.1109/IROS.2014.6943123.

John Schulman, Sergey Levine, P. Abbeel, Michael I. Jordan, and P. Moritz. Trust region policy optimization. *ArXiv*, abs/1502.05477, 2015.

Shaoxiong Wang, Mike Lambeta, Lambeta Chou, and Roberto Calandra. Tacto: A fast, flexible and open-source simulator for high-resolution vision-based tactile sensors. *Arxiv*, 2020. URL `https://arxiv.org/abs/2012.08456`.

S. Dong N. Sunil A. Rodriguez Y. She, S. Wang and E. Adelson. Cable manipulation with a tactile-reactive gripper. In *Robotics: Science and Systems (RSS)*, 2020.