Leveraging Mobility for Robot Grasping Tasks

Samantha Wathugala

May 15, 2019

Acknowledgement

This work was conducted with the mentorship of Ph.D candidate Gregory Kahn, Post-doc Chelsea Finn, and Professors Aviv Tamar and Pieter Abbeel. Thank you to undergraduates Anna Zhu and Sam Lowe that helped last year with the project setup: Anna Zhu especially with toy detection and the baseline navigation script, Sam Lowe especially with the data collection setup, and both with assembly and maintenance of our dear but occasionally disagreeable robot. Finally, thank you to masters candidate Jack Sullivan, for co-conducting the data collection, analysis, and writing for the work done in Fall 2018.

Leveraging Mobility for Robot Grasping Tasks

Samantha Wathugala

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee

[illegible]

Pieter Abbeel
Research Advisor

5/15/19

(Date)

★ ★ ★ ★ ★ ★ ★

Sergey Levine

Sergey Levine
Second Reader

5/15/2019

(Date)

Abstract

Navigation is often required in order to perform complex grasping tasks such as cleaning unstructured household rooms. Although there is extensive prior work in applying reinforcement learning to grasping and navigation, these tasks have largely been separate from each other. Our work's primary contribution is to determine whether learning to navigate in a way that might improve grasping would provide an increase of grasp success.

Exploring this objective and answering this question led to two stages of work, both grounded in the complex task of cleaning up toys scattered in a room. In the first stage, assuming that we could navigate to any pre-grasp orientation, we investigated whether certain orientations were more amenable to successful grasps. We trained probability of success models as a grasping policy, to potentially introduce a heuristic that we could judge graspability off of. We found that, though that heuristic was good enough to produce reasonably effective grasping policies, it was not good enough to be used in lieu of actual grasp success. We then ran those policies on the physical robot with a single toy and observed the actual grasp success rates and images collected. We did find that for each grasping policy, there existed toy orientations that proved to be more difficult to grasp than others.

With this idea validated, we next implement a navigation policy to approach toys at more graspable orientations, and compare its grasp success to the original navigation policy. We use a simple nearest neighbor's approach to determine which pre-grasp orientations of a certain toy were good. In real time, before each potential grasp, we use a Structure-from-Motion model and the robot camera's current image to estimate the angle of the robot's approach to this toy, and then determine whether to grasp or to continue to circle the toy based on how successful past grasps at that approach have been. From implementing and running this policy, we find that this navigation policy indeed improves grasp success from 59% to 80%. This demonstrates the hypothesis that was previously only validated: training a navigation policy based on former grasping experience can and does improve grasp success in tasks that involve both.

Contents

1 Introduction

At present, robots in industrial environments can handle challenging manipulation tasks such as automotive assembly and warehouse automation, while household robotics are limited to "intelligent" vacuum cleaners. The main reason for this gap is the unstructured nature of household domains, which requires complex sensing, reasoning, and acting skills. Over the last decade, deep learning has shown great promise in learning from unstructured data, and has revolutionized computer vision and reinforcement learning. More recently, learning methods have been receiving increased attention in robotics, and hold promise in dealing with unstructured domains.

Specifically, we focus on grasping tasks in unstructured environments. Navigation is often required in order to perform complex grasping tasks such as cleaning unstructured household rooms. Navigating and operating in unstructured environments is challenging as robots cannot rely on having complete knowledge of their environment. Robots instead have to acquire and process the necessary state information to support decision making. A robot in this setting also has to continuously monitor their impact on the environment as it is assumed that their actions won't always be successful. This leads to our investigated objective, which is to determine whether a mobile robot can utilize its mobility to improve its grasping.

Our robotic platform is a TurtleBot 3 mobile base equipped with a Widow-X 6-DoF manipulator arm. This platform has suitable sensing and manipulation capabilities, though the inherent challenge of low cost robots is that the grasping arm is not as accurate as it could be, therefore making the data and performance more unreliable. However, this low cost makes the robots more accessible and deployable, and our results more robust.

We investigate our objective in two parts. In the first half, we investigated whether certain orientations were more amenable to successful grasps in order to determine the utility of learning to navigate in a way that might improve grasping. After all, if navigation did not help, not only would it not be worth the effort of implementation and maintenance, it could actually hurt performance. For example, a robot's task might be to pick up and put away all the toys in a room as quickly as possible. The robot might take longer if it takes a circuitous route to approach a toy at different angles with a learned navigation policy, instead of navigate directly to the toy using a simpler or hardcoded policy and just try its best to grasp from there.

Exploring this objective and answering this question led to training models to predict the probability of success value of a grasp given the current position of the robot arm pose and image of the toy. We conducted experiments that trained multiple models and ran those models on the physical robot to compare their grasp success rates on two sets of different orientations of a single toy, a girl doll. Our first set of orientations is up/down/left/right and our second is horizontal/slant/vertical. Our different models differ by training on different combinations of data sets (e.g. training on data of grasping 9 toys equally vs. training on data that focuses more on grasping the test-time toy). We observe how well our models determine successful grasping regions with comparing heatmap visualizations and the current images of the toy, and running the models at different orientations and observing the grasp success rate for each orientation.

We found that for each model, there existed girl doll orientations that proved to be more difficult to grasp than others. In a vertical/horizontal/slant orientation scheme, model (1) grasps the toy at a "vertical" orientation better, 64% of the time as opposed to 52% or 53% at the others. In a up/down/left/right orientation scheme, model (2) grasps the toy worse at a "down" orientation, with only 39% grasps successful as opposed to its overall grasp success rate of 48%. When observing probability of success heatmap visualizations of our models, we found that the probability of success value magnitudes of their models did not correspond well to whether they ultimately successfully grasped or not. Therefore, we did not use our heatmaps in our analysis of advantageous orientations. Still, the actual grasp success counts were sufficient to validate our hypothesis.

In the second half of this work, we actually change the navigation policy to reflect observations from the baseline navigation and grasping policies, and compare the grasp success of the robot using this policy to the grasp success during the original navigation policy. Once again, we use a single toy, this time a baby doll. In the previous section, orientations were determined after the fact, by hand. For the orientation scheme in this section, we built a Structure-from-Motion model of this toy. In real time, before each potential grasp, we used that model and the robot camera's current image to estimate the angle of the robot's approach to this toy, and then determined whether to grasp or to continue to circle the toy based on how successful past grasps at that approach have been. From implementing and running this policy, we find that this navigation policy indeed improves grasp success from 59% to 80% on this toy. This proves the hypothesis that was merely validated in the first half of this work; training a navigation policy based on former grasping experience can and does improve grasp success in tasks that involve both.

Finally, throughout this process, we created a flexible, robust experimental setup for this robot and task, and collected enough data to train several deep learning models that grasp well, as well as much additional on-policy data (i.e., the data from running model (1) and model (2)) that we have not leveraged yet. Such data will be necessary for future projects with our robot.

2 Related Work

The task of grasping unknown objects is very difficult with the variety of different objects and environments. Robotic grasping has extensive prior work that is comprehensively reviewed by recent surveys such as Bohg et al. [4]. Early work such as Bihi et al. [3] and Mason et al. [15] found analytical solutions to grasping, but these approaches assumed a structured environment where precise information about the environment and its dynamics such as external forces and surface properties are known. Standard supervised learning can be utilized to train grasping tasks on robots [19] but requires extensive initial input and monitoring from human supervisors. The literature has recently favored more flexible, trial and error based approaches such as reinforcement learning.

Similar to our work, many current grasping approaches try to predict a grasp pose by observing the environment (commonly with a depth camera), choosing the best location at which to grasp, and then executing an open-loop planner to reach that location [17]. While most of these studies focus on optimizing the number, percentage, and efficiency of grasp successes [6] and [16], we instead focus on exploring the difference of grasp success when grasping objects at different object rotations.

Levine et al. [13] used images of the robot arm gripper and the object to predict probability of grasp success using a large convolutional neural network of the current image. Our approach is similar in that it uses the image of the gripper and the object with a convolutional neural network, but we also include the robot's arm pose (arm position in the environment) in our state. Another similar work is self-supervised learning of grasp poses by Pinto and Gupta [18]. This approach was trained with self-supervised data collected using a heuristic grasping system based on object proposals to learn a network to predict the optimal grasp orientation for a given image patch. Our method differs by not requiring proposals or crops of image patches and by using a small, low cost robot (3K USD vs their > 22,000K USD), therefore ours would be more practical to deploy in real households.

There has also been extensive related work focusing on applying reinforcement learning to robot navigation [11], [22], [1]. Many previous navigation works attempt to create a map of the environment [7], but having only the knowledge of the camera field of view has been preferred as the robot does not have to store as much information and can react to obstacles and changes in an unstructured environment [2]. This work uses mobility and image recognition to navigate in its unstructured environment instead, to hypothesize about how this navigation data could be leveraged to improve grasp success, as opposed to learning a more complicated navigation policy (e.g. by reinforcement learning) as a virtue in itself. Reinforcement learning's effectiveness for learning to navigate to achieve certain goals, e.g. collision avoidance [11], does suggest that it could be promising to train navigation for a heuristic that improves the probability of grasp success.

This work explores the potential of training navigation to improve grasp performance rather than only train grasping to improve grasp performance. Kalashnikov et. al.'s expensive stationary graspers learn to adjust the object - by knocking it over, for example - to make it easier for them to grasp [12]. Our robot also seeks to optimize the way that the object's position relates to the its own, but using its mobility to do so. There are many tools and algorithms for scene reconstruction, feature extraction, and object homography that this work uses or at least takes inspiration from in the process of doing so, including SIFT feature extraction, RANSAC, and Structure-from-Motion scene reconstruction as implemented by COLMAP [14], [8], [20].

3 General Task Setup

3.1 Task Steps

Our motivating task is to declutter an unstructured room filled with toys. For each toy, the robot needs to

1. navigate towards the toy
2. attempt to grasp the toy
3. check whether it successfully grasped the toy
4. upon successful grasp, perform the goal action that represents putting the toy where it belongs.

where the robot repeats steps 2-3 until the grasp is successful, and then returns to step 1 on a different toy. The overall task, therefore, is to successfully grasp and perform the goal action on all of the toys in its environment.

3.2 Robot

Our robot is a WidowX MKII Arm mounted on a Turtlebot3 Waffle base. She has two cameras: an Intel Realsense camera facing horizontally, and a Genius WideCam F100 facing downwards. This setup is relatively inexpensive, so successful policies have to be more tolerant of equipment error than if we used a higher-end robot.

3.3 Success Evaluation

We judge success or failure in a self-supervised fashion. After closing its gripper, the robot judges the grasp a success if both of the following are true, and a failure otherwise: (a) the Realsense image contains a toy when the arm holds its gripper near it, and (b) the Realsense image does not contain a toy when the arm lifts its gripper high out of that camera's view.

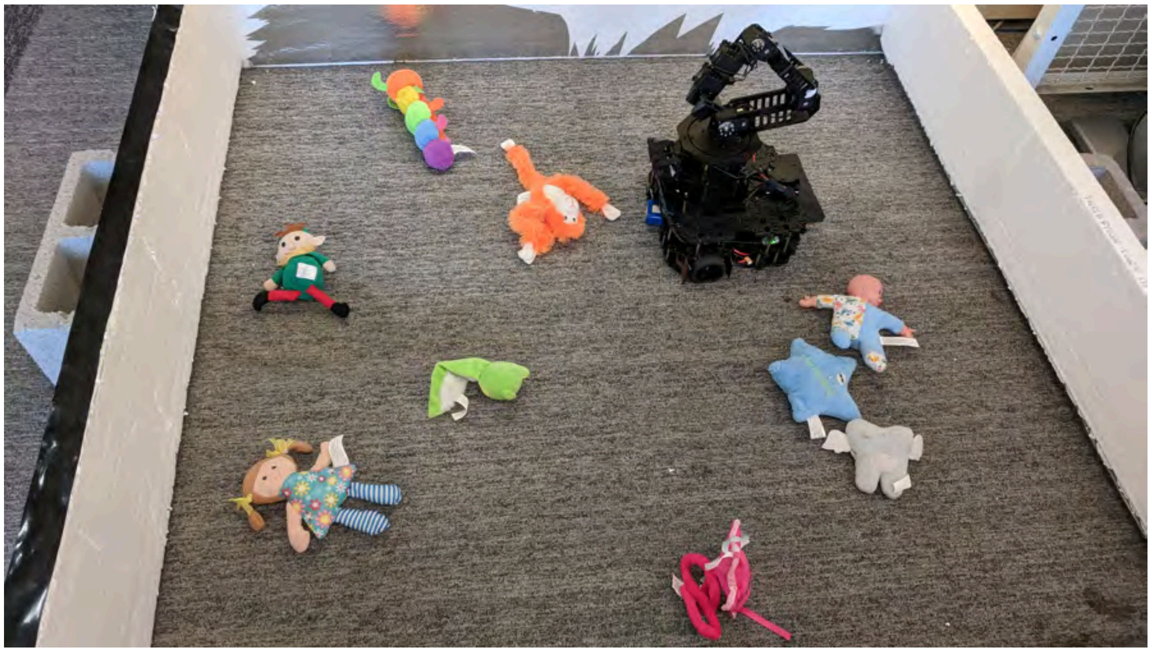
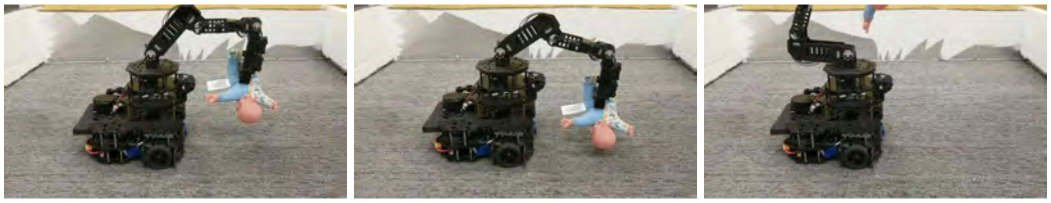


Figure 3.1: Robot in room with all nine toys



(a) Lift the toy off the ground. (b) Check that it is in view. (c) Check that it is out of view.

Figure 3.2: Three steps to judge a success with the Realsense camera.

3.4 Baseline Navigation Policy

The baseline policy's navigation script locates the nearest toy bounding box in the Realsense and Genius camera images and moves forward, left, or right based on what might move that toy's box to the bottom center of the camera images. For toy detection, we used Tensorflow's Object Detection API with ResNet-101 feature extractors pretrained on the COCO dataset and then finetuned with our set of toys through hand-labeled images from our cameras in which we manually marked bounding boxes for all toys and non-toys [10].

This navigation policy gives us the ability to see toys from different angles over time, and to run the setup with minimal supervision over all of its toys.

3.5 Baseline Grasping Policy

When the robot has approached the toy based on the toy's bounding box in first the Realsense camera and then in the Genius camera, the robot uses the Genius camera image's bounding box coordinates to estimate a center of the bounding box (x, y) and height of the toy (z). Then, using MoveIt! inverse kinematics [5], the WidowX arm aims for that estimated (x, y, z), closes the gripper, and then evaluates success. If that grasp fails, the robot lowers the z value and tries again with the same (x, y). It continues to try successively lower z values until it either succeeds or reaches the ground, at which point it gives up on this approach toward this toy and returns to navigation after some kind of adjustment or random spin.

4 Preliminarily Determining Whether Angle of Approach Matters

4.1 Preview

In Fall 2018, our experiments used the baseline navigation policy, but trained new grasping policies via reinforcement learning from experience that the robot collected from using the baseline navigation policy with the baseline grasping policy. The grasping policies trained were probability of success models, because that could potentially introduce a heuristic that we could judge graspability off of. We found that, though that heuristic was good enough to produce reasonably effective grasping policies, it was not good enough to be used in lieu of actual grasp success. We then analyzed the affect of toy orientation using the robot's actual grasp success rates and Genius camera images from running the baseline navigation policy and those trained grasping policies. In our analysis we found reasons to validate our hypothesis that angle of approach can indeed help or hurt the robot's chances of grasping successfully.

4.2 RL Setup for Training Grasp Policies

4.2.1 RL Model

As our robot explores and attempts to achieve its task, every 0.25s, we record the image from its genius camera, the pose of its arm, and the result of checking grasp successes. For our model, we only look at the data during the arm's descent towards a toy for each grasp. The observation is (genius image, current arm pose), the action is moving the arm to the next pose in the rollout, and the reward is 1 if the next pose is the end of a successful grasp, -1 if the next pose is the end of a failed grasp, and 0 if the grasp is not yet completed. We made each rollout one step long, where the state is any place (in image and pose) along the descent trajectory, and its action consists of moving the arm towards the final place along that descent trajectory. This meant that data from one H-step grasp trajectory (where $roll_t$ denotes a rollout at step t, im_t denotes an image at step t,

$pose_t$ denotes the arm pose at step t, 0 denotes a non-terminal state, and $success$ is ±1 to indicate grasp success (+1) or failure (-1)):

$$(im_1, pose_1, 0), (im_2, pose_2, 0), \dots, (im_H, pose_H, success)$$

would produce $H - 1$ rollouts with horizon 1 that look like this:
$roll_1$ = ((state$_1$,action$_1$,reward$_1$)) = $(((im_1, pose_1), pose_H, success))$
$roll_2$ = ((state$_1$,action$_1$,reward$_1$)) = $(((im_2, pose_2), pose_H, success))$
...
$roll_{H-1} = (((im_{H-1}, pose_{H-1}), pose_H, success))$

Notice that each rollout's action moves the arm directly to the final pose. Equipment limitations made this approach more appealing than one in which the we trained smaller actions. The intermediate rollouts allow us to get more out of our dataset, and improved performance when we put policies into practice and the robot needed to make decisions from halfway down towards the toy rather than its initial pose. The video stills in Figure 4.2 illustrate this process.

4.2.2 Baseline Policy

Collecting data from our low-cost robot doing its task in realtime is expensive, as opposed to a simulated task, or a setup with a powerful robot plugged into the wall. Therefore, we wanted the baseline policy to have certain qualities for collecting off-policy data:

- Judging success or failure in a self-supervised fashion, as described before.
- Occasionally achieving success and promoting exploration. Our reward setup, +1 for a successfully completed grasp and -1 for a failed completed grasp, is too sparse to start with a completely random approach. So, instead, we designed a fairly competent baseline, based on navigating towards and grasping towards our aforementioned toy detector's bounding boxes. These calculated grasps were about 22% successful, so they generated data with both positive and negative rewards. To still explore more of the state space than this deterministic method, we also occasionally added noise to these grasps.

4.2.3 Probability of Success Model

The probability of success model we trained was a supervised learning model from (image, current pose, action pose) to the probability of grasp success.

Let us denote each rollout for step t of grasp g as $roll_t^g$ (where all quantities with g superscripts denote grasp g):

$$roll_t^g = ((im_t^g, pose_t^g), pose_H^g, success^g \in \{-1, 1\})$$

From each rollout $roll_t^g$ we created our model input x:

$$x = (x_{im}, x_{pose}, x_a) = (im_t^g, pose_t^g, pose_H^g)$$

and our model output y:

$$y = max(0, success^g)$$

Using the parameters and architecture tuned in Kahn et. al. [11] for collision probability, the model passes the genius camera image through a convolutional neural network (CNN), the observation vector through a fully connected network (FC), the action vector through a FC, and then those three inputs combined through an FC to get the final output probability of success.

The model with parameters $\theta = (\theta_o, \theta_{im}, \theta_{pose}, \theta_a)$ outputs a probability of successful grasp for $x = (x_{im}, x_{pose}, x_a)$:

$$p_\theta(x) = \mathrm{FC}_{\theta_o}(\mathrm{CNN}_{\theta_{im}}(x_{im}), \mathrm{FC}_{\theta_{pose}}(x_{pose}), \mathrm{FC}_{\theta_a}(x_a))$$

where θ was trained to minimize the mean cross-entropy loss over each training rollout's x with success label y (1 for successful, 0 for unsuccessful):

$$L_\theta(x, y) = -(y \log(p_\theta(x)) + (1 - y) \log(1 - p_\theta(x)))$$

4.2.4 Data Collection Configurations

We used data from three task and policy configurations to train the probability of success model:

1. The baseline policy, including grasp exploration, on 9 soft toys: 9202 grasps (24.2% successes)
2. A deep Q-learning grasping policy trained from (1), with the states, actions, and rewards as described in the horizon-1 RL model, on 9 toys: 614 grasps (19.9% successes)
3. The baseline policy, including grasp exploration, on 1 toy: 336 grasps (33.6% successes)

where the baseline policies occasionally introduced noise into the grasp, to increase exploration.

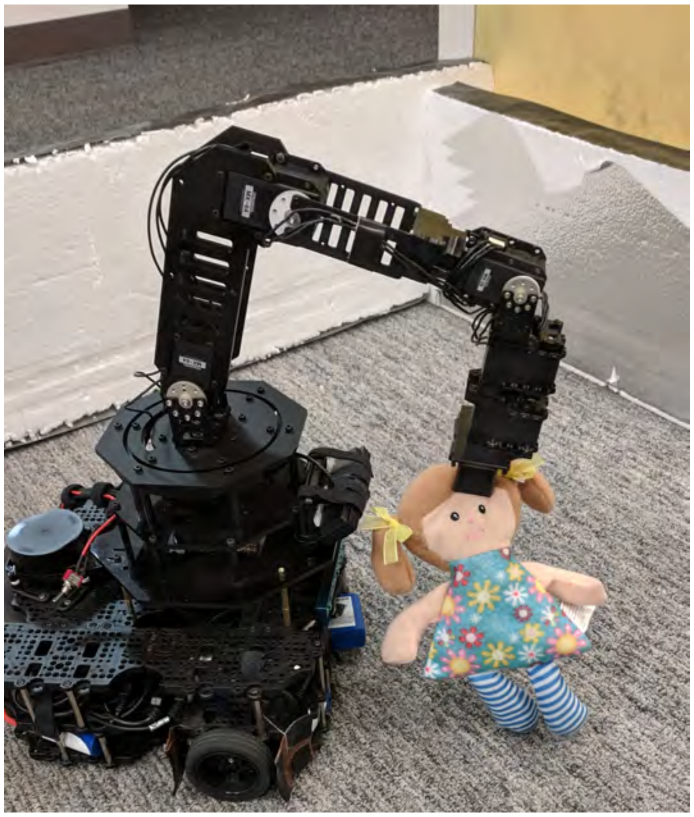


Figure 4.1: Robot with girl toy, the toy used to collect dataset (3)

All three configurations used the baseline policy's hardcoded navigation script. In addition to the aforementioned Genius camera images, we also collected Realsense camera data and Turtlebot command velocity data from this, which we could possibly use for future work with this setup if this paper's results suggest training navigation in a certain way might improve the grasping task's performance.

We chose to focus a portion of our dataset on only one toy to see if doing so would allow the probability of success model to train more definition around the toy. We chose the girl toy for being asymmetric, soft in all areas (some other toys had hard portions that posed a danger to our gripper), and colorful (some other toys were grey and caused the the checking routine to return false negatives).

4.3 Experiments

Our experiments will help us determine whether a mobile robot can utilize its mobility to improve its grasping by training multiple models and comparing their grasp success rates on different orientations of a single toy. Our experiment will train probability of success models to compare visualizations and grasp success rates for different orientations across different probability of success models.

4.3.1 Probability of Success Models Trained

The different experiment models used will be all of the type probability of success (described in the *Task Setup* section) and thus have the same network architecture. The only difference is the data that we use to train the models. All data used to train is not orientation specific (i.e. contains a random proportion of different toy orientations). The combined datasets are referenced from the *Data Collection Configurations* section:

1. Combination of (1), and (2)
2. Combination of (1), (2), and (3)
3. Model (1) finetuned on dataset (3)

4.3.2 Predicted Grasp Success

We can observe how well our models determine successful grasping regions with comparing heatmaps visualizations and the current image of the toy and running the models at different orientations and observing the grasp success rate for each orientation.

Heatmap Visualizations

We create heatmap visualizations of a model's probability of success values for each genius image to observe which region of the image the model thinks is successful. Specifically, for a given robot gripper height (z dimension is fixed), we evaluate the model's probability of success at every spot in its feasible rectangular region (all horizontal and vertical arm positions x's and y's).

These heatmaps use the probability of success model as a source of truth to draw conclusions about this inexpensive setup's ability to grasp toys at different angles. To see if the probability of success model itself is realistic, we observed the robot's performance using a policy that uses the cross-entropy method to choose actions with a high predicted probability of success by trying different actions and observing which actions maximizes probability of success.

Grasp Success

For each policy, we calculate the grasp success rate for all four orientations (up, down, left, and right). We record the number of successes and failures in addition to the grasp success ratio. The null hypothesis of our experiment is that there is no or negligible difference between the success rates of the different orientations.

We do this with the trained policies and not the baseline policy because we wanted to have a sense of "on policy" grasp success.

4.4 Results

Two notable and non-trivial preliminary results that made testing this hypothesis possible were collecting sufficient amounts of data from different policies and getting other reinforcement learning methods such as deep Q-learning working with reasonable grasp success. Because our robot is mobile, data collection was time consuming as it was dependent on us having to frequently charge and change batteries. We have over 10,000 grasps that we can use for future projects. The deep Q-learning policy trained, with a grasp success rate of 19.9% on all toys, worked around as well as the baseline policy (24.3% with occasional noise on all toys, 22.2% definitely noiseless on all toys). The probability of success policies, one achieving 56% grasp success on the girl toy and another 48%, did better than the baseline's 33.6%. The fact that we were able to train grasping policies that worked that well shows that our baseline policy collects useful data and that we have amassed a useful amount of it. This gives us confidence that we could try other RL models or explore other related problems with at least this data, and if they are not successful, it would not be for lack of data.

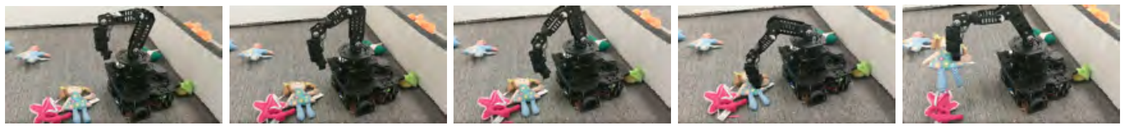

(a) initial pose (b) step 1 (c) step 2 (d) step 3 (e) success!

Figure 4.2: Model (1) in action: In steps 1 and 2, equipment limitations prevent our robot from going all the way to her destination, so she goes partway each time, and the policy plans from there. In step 3, she completes her journey to the toy, and then lifts the toy to check for success.

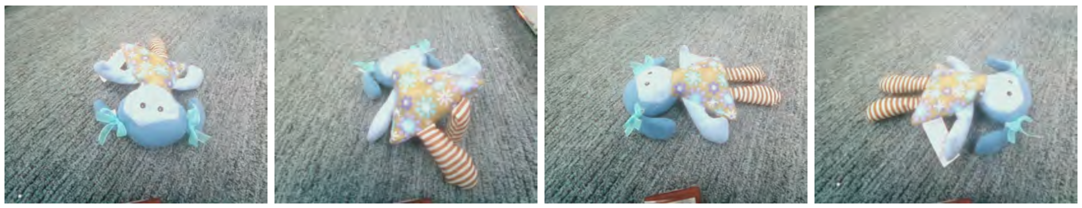
We ran model (1), model (2), and model (3) on the robot, since a reasonable probability of success model should be able to perform reasonably well, reasonable meaning that it should succeed at least some of the time, so that we can compare nonzero success rates across different angles of the toy and trust the model's confidence in itself when we generate heatmaps.

Model (1) had reasonable performance on the one girl toy: it made 133 successful grasps out of 238 total, for a grasp success rate of 56%. Models (2) and (3) were expected to do at least as well on the one girl toy, as they had additional data on just that toy. (3) did not; the grasps that the robot attempted under that policy were nowhere near the actual toy (success rate 0%). 336 grasps was evidently not enough data to train on, and the new model finetuned on just those grasps overfit that limited data. Therefore, we do not present heatmaps or grasp success rates in the following sections for that policy. Model (2), trained on all the data at once, including the extra girl toy data, did perform reasonably well on the girl toy: it made 185 successful grasps out of 383 grasps, for a grasp success rate of 48%. Considering that the baseline policy's grasp success rate was

A typical successful grasp can be seen in the video stills in Figure 4.2.

4.4.1 Grasp Success

We observed 238 grasps running model (1) on the robot, and 383 grasps running model (2) on the robot. We sorted these grasps by orientation, by two schemes, illustrated in Figures 4.3 and 4.4. All images were sorted by hand, even though some pictures were ambiguous to sort; e.g., (c) in Figure 4.4 might fit into either down or right for orientation schema (1); (b) in Figure 4.3 might fit into either vertical or slant for orientation schema (2). Despite these ambiguities, there were only three grasps omitted from sorting, only because they for some reason did not produce a picture of the toy before the toy was grasped.



(a) down (b) up (c) left (d) right

Figure 4.3: Orientations of girl toy in orientation schema (1)

Tables 4.1 - 4.2 show the grasp successes for each model for each orientation.

We can see from this simple sorting that we do see dramatic differences in grasp success by angle for each model. Model (1) does much better for toys at "vertical" orientations (so, ones where

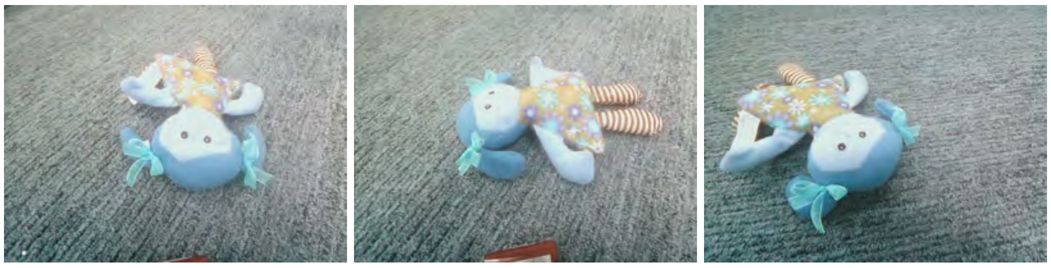

(a) vertical (b) horizontal (c) slant

Figure 4.4: Orientations of girl toy in orientation schema (2)

the toy varies less in the robot arm's y direction) than for toys at others. Model (2) actually does a bit worse if the toy is at a "vertical" angle, and much worse for those at a "down" orientation.

Table 4.1: Horizontal/Slant/Vertical Grasping Successes

	Model (1)		Model (2)	
Orientation	# grasps	% success	# grasps	% success
horizontal	79	52%	125	47%
slant	87	53%	175	51%
vertical	72	64%	80	41%

Table 4.2: Up/Down/Left/Right Grasping Successes

	Model (1)		Model (2)	
Orientation	# grasps	% success	# grasps	% success
up	63	60%	77	47%
down	43	58%	76	39%
left	86	51%	116	54%
right	46	57%	111	50%

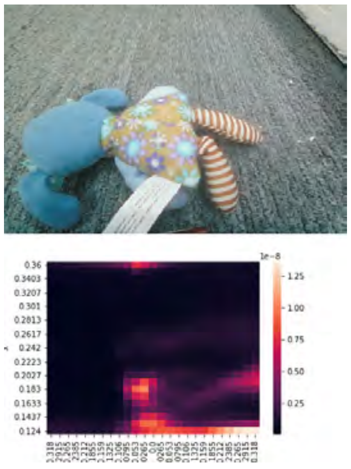
4.4.2 Heatmaps

Counting actual grasp successes and failures by orientation did provide a quantitative measurement how graspable a toy is by angle. We also generated heatmaps in hopes of noticing another measurement of graspability per image, perhaps through the magnitude of the model's predicted probability of success for its best grasp action.

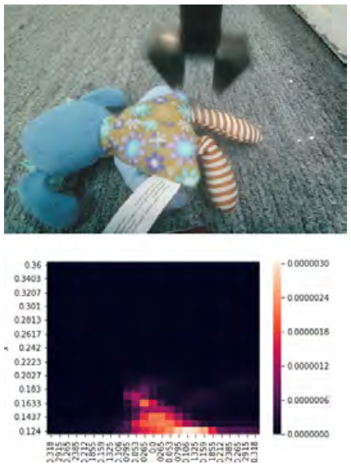
We found that the probability of success value magnitudes of their models did not actually correspond well to whether they ultimately successfully grasped or not. Considering that their policies did perform reasonably well, as discussed earlier, the relative magnitudes between poses was enough to help the robot make good decisions. However, the models alone clearly did not have a good idea whether doing its best would be good enough to result in a grasp based on our produced heatmap visualizations.

An example of the probability of success heatmaps changing drastically between images in the same rollout can been seen in Figure 4.5. Within one grasp, the model's prediction of its best chance change from 0.00000125% to 0.00030% to 100%. This is the same toy, and the robot has basically the same options about where to put its arm next between (a) and (b) regardless of its current pose (the only input to the model other than the image). Thus, the probability of success magnitudes cannot verify whether a grasp will ultimately be successful.

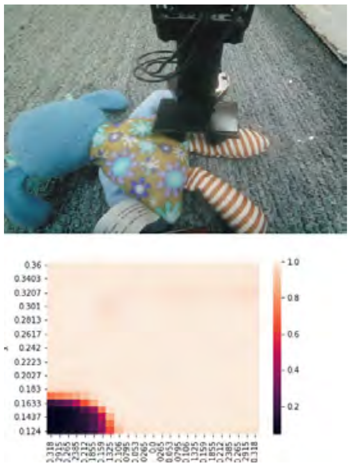
On the other hand, the map does not lead the robot too astray where it matters for the policy; in Figure 4.5(a), one can see the definition of the doll's legs. Figure 4.6 illustrates more examples of the accurate shapes shown in the heatmaps from model (2).



(a) 1.25e-8



(b) 0.0000030



(c) 1.0

Figure 4.5: Model (2) in action, with the images it sees and the model visualized as heatmaps. Each heatmap scales from 0 to a different number.

The heatmaps from model (1) were similarly flawed, and their heatmaps did not have as high of a definition. So, although the heatmaps did explain the robot's decisions as it ran through the policy, they do not give any hints as to how toy orientation affects graspability; only the actual results of running the policy do that.

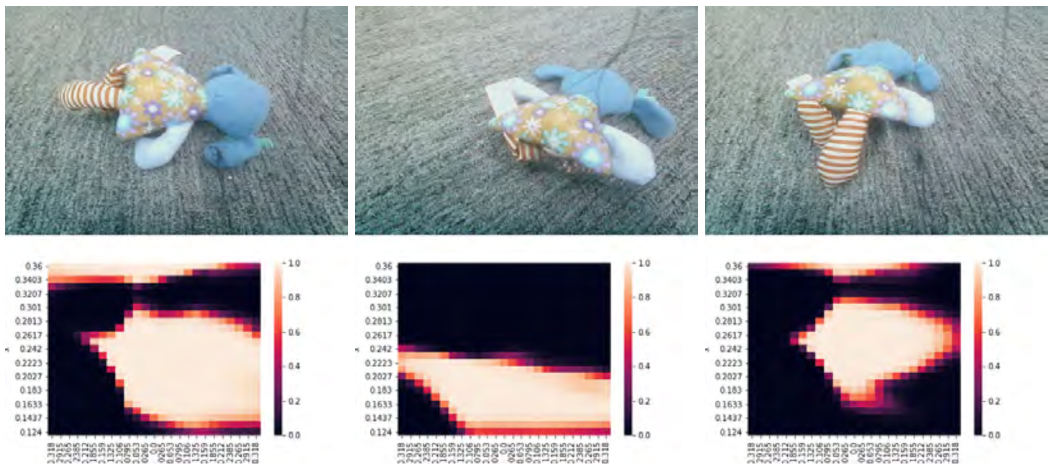

(a) (b) (c)

Figure 4.6: More heatmaps from model (2) in action, where the shapes roughly correspond to the doll's position. Note that the perspective of the camera is angled forward, while the axis of the heatmap is for a constant height (z-dimension) near the ground; therefore, the heatmaps are more top-down than the corresponding images.

4.5 Conclusion

We found that for each model, there existed girl doll orientations that proved to be more difficult to grasp than others.

The question we pose is not a controllability problem; we are not asking what the robot is capable of. If that were the question, then the arm's six degrees of freedom are plenty to pick up a toy whatever way it is flopped over. The question is not what the robot *can* do but what it *would* do given a reasonably good policy. The baseline policy, the deep Q-learning policy, model (1)'s policy, and model (2)'s policy are all reasonably good, achieving at least the baseline policy's performance; and, in the case of our probability of success models model (1) and model (2), achieving twice as much grasp success.

For both of these reasonable policies, we found that there were angles that that policy might want to avoid or pursue. Model (1) might want to approach a toy so that it fits a "vertical" orientation; model (2) might want to approach a toy so that it does not fit a "down" orientation or even a particularly "vertical" one. These are just a few handpicked and handsorted orientations. In practice, to learn what orientations - or, more broadly, kinds of images of a toy - to prefer or avoid, one would want to use some kind of unsupervised learning technique to make these distinctions.

As this exploration so far shows promising results, future work should be to implement something that uses the ideas validated here to improve our overall task completion time. First in doing that, a less manual pipeline to categorize images into good and bad angles is necessary, both to make this scalable considering that different models have different preferences, and so that there's a task-time way for the robot to categorize its current situation. We deliver on that goal in the next chapter. Then, there are many roads to exploit this. The robot could rotate and transform a current image of a toy so that it looks more like angles that we in general successfully grasp, predict a grasp from that, and transform it back: this leverages mobility in that it uses data that the robot only got through its navigation policy. Another approach might involve directly training navigation: When close to a toy, the robot can adjust the way it approaches it to make it fit one of its better angles, e.g. rotate a bit to the left or right so that the toy is more "vertical" in our view. This could involve a video prediction model. Again, one avenue for implementing these ideas, and testing them more concretely, is indeed explored in the next section.

Finally, in completing this preliminary work, we created a flexible, robust experimental setup for this robot and task, and collected enough data to train several deep learning models that grasp well, as well as much additional on policy data (i.e., the data from running model (1) and model (2)) that we have not leveraged yet. This data is bound to be useful whatever direction anything involving this robot and setup takes.

5 Implementing a Navigation Policy with Grasping in Mind

5.1 Basic Structure

The basic structure for a navigation policy that considers the graspability of toy, as implemented in this chapter, is such:

1. Explore the space and navigate until spotting a toy in either camera.
2. Navigate towards the toy as is sufficient to evaluate the toy's position.
3. Evaluate the toy's position from the image: Should we grasp now, or explore more and grasp later?
    a) If the former, attempt grasp. After grasp attempt, return to (1).
    b) If the latter, abandon this approach to this toy. Adjust our position relative to the toy and return to (3).

The difference between this and the baseline navigation policy is that, where this policy sometimes makes the decision to not attempt a grasp in (3), the baseline navigation policy always tries to grasp.

5.2 Methods

5.2.1 Grasp Policy

In this work, we focus on developing the navigation policy to optimize a grasp routine while otherwise keeping that routine constant. We use the Baseline Policy as that routine. In this work, we define a grasp attempt as the series of grasps (subgrasps) that ultimately end in the lowest z. That way, a grasp attempt (and therefore, this approach) is a failure only if all of that series of subgrasps fail; if the final subgrasp succeeds, then this grasp attempt is a success. This is different from the previous section, where each of those subgrasps was considered individually. If considered individually, then this approach does not work - the robot's baseline policy, in being cautious as to not squash the toy more than necessary or safe, usually initially reaches slightly too high, and only achieves success after trying one or two slightly higher grasps. For a project where we are not training a grasp policy, the subgrasps cannot be considered independently, for any subgrasp that succeeds usually only succeeds because of the subgrasp(s) preceding it.

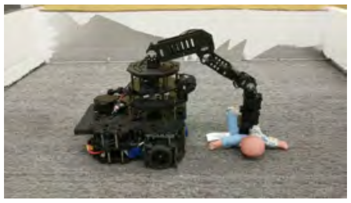
Video stills from one successful grasp attempt, comprised of two subgrasps, are shown in Figure 5.1.

(a) First subgrasp

(b) Close gripper

(c) Lift - ah, it failed.

(d) Subgrasp 2: lower z


(e) Squeeze the toy

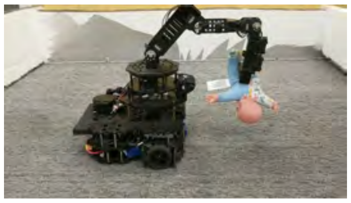

(f) Success!

Figure 5.1: The robot only tries the successful lower z after trying an unsuccessful higher z.

5.2.2 Toy Classifier

The toy classifier is a simple classifier finetuned for our nine toys on top of a partially frozen Keras MobileNet architecture pretrained on Imagenet [9]. This classifier's input data is the first image of a toy taken at the beginning of the grasp routine during months of running the robot. While the hardware, location, and set of toys in the images are the same in all the images, the images had a variety of lighting conditions and often multiple toys in frame (even though the robot was aiming to grasp just the closest one). Despite these conditions, the classifier trained for one epoch on 679 labeled training images achieved 82% accuracy on an 80-image test set.

This classifier was then used on the following data: images from running the baseline navigation policy and the baseline grasping policy that are unambiguously noiseless. (Recall that for training in the previous experiments, we added exploration noise to the baseline policy's grasp decisions, which was okay then because that data was off-policy. For this experiment, we require on-policy data for the grasping policy.) This is necessary for the pose evaluator.

A classifier such as this could also be used at task-time for experiments where multiple toys are in the robot's view. In such experiments, the robot will need to decide which toy it is seeing in order to decide which pose evaluator to use. For that, the images don't need to only come from that limited dataset described. So far in this work, we only use one toy, the baby doll toy, so we do not use any classifier for that purpose.

5.2.3 Pose Evaluator

Developing our pose evaluator had three steps:

1. We took a clear video of the toy and trained a Structure-from-Motion (SfM) model on it using COLMAP [20], [21].

2. We ran COLMAP on the images that our toy classifier labeled as that toy. This meant attempting to incorporate each image from that training set into the SfM model.

   For an incorporated image, COLMAP gives the projection from world to camera coordinate system using a quaternion defined using the Hamilton convention and a translation vector T_{vec}. The (x, y, z) coordinates are given by

   $$(x, y, z) = -R^T * T_{vec}$$

   where R^T is the inverse/transpose of the 3x3 rotation matrix composed from the quaternion.

   Then we calculates the angle θ as simply

   $$\theta = \tan^{-1}(\frac{x}{y})$$

   and label that image with that angle.

   Thus, this step sorts the images by angle and by ultimate grasp attempt success.

3. We sorted the images into eight angle bins and calculated the grasp success rate at each bin. We labeled the "good" bins as ones that perform better than the average grasp success of the baseline policy, as determined over the incorporated training set images.

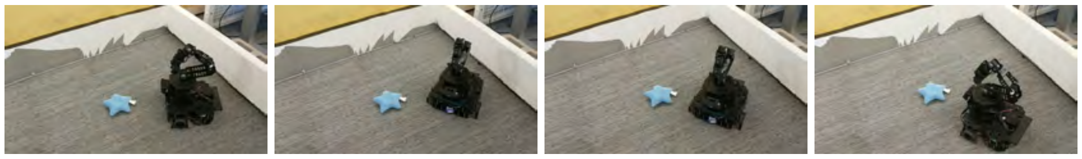
At test time, the pose evaluator estimates the angle bin of the toy from the image. This is done through the same process that binned the training data. If the image can be incorporated and angle determined, and that angle is in one of the "good" bins, it tells the robot to grasp now. Otherwise, it tells the robot to move on.

5.2.4 Navigation Policy

The navigation script was designed to "circle" the toy until succeeding. Upon seeing the toy, the robot would:

1. Approach the toy until able to call the pose evaluator (e.g. close enough).
2. If the pose evaluator returns not to grasp, then back up, turn to the right, go forward, turn back to the left, and return to step (1) to approach to toy again.
3. If the pose evaluator returns to grasp, then perform the grasp routine.

This pattern is illustrated in Figure 5.2.



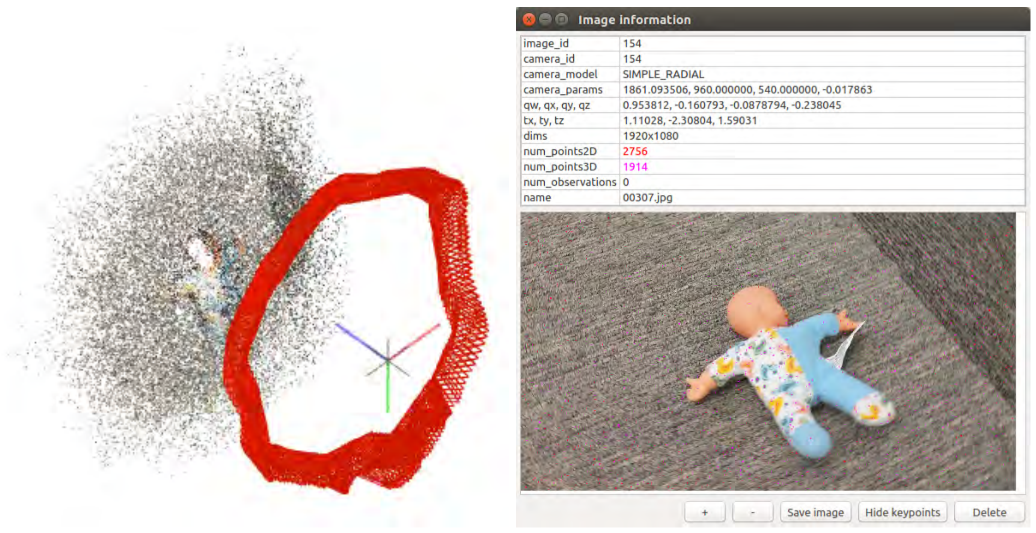
(a) approach toy (b) turn away (c) move forward (d) turn back to toy

Figure 5.2: Video stills from the circling navigation policy in action. Each time the robot faces the toy, she evaluates her image to decide whether to attempt to grasp or continue to circle without grasping.

5.3 Results

5.3.1 SfM-Assigned Angle Bin Performance

199 video stills were used to create the SfM model in COLMAP software shown in Figure 5.3. Originally, 398 images were used, but this made incorporating one additional image take much longer. With 398 images, the vocabulary tree matching stage took 30 seconds, which is unacceptably slow considering that this is part of the task-time policy. Ultimately our goal is to improve performance on the task and clean the room faster. A higher rate of grasping success that comes at



image_id	154
camera_id	154
camera_model	SIMPLE_RADIAL
camera_params	1861.093506, 960.000000, 540.000000, -0.017863
qw, qx, qy, qz	0.953812, -0.160793, -0.0878794, -0.238045
tx, ty, tz	1.11028, -2.30804, 1.59031
dims	1920x1080
num_points2D	2756
num_points3D	1914
num_observations	0
name	00307.jpg

Figure 5.3: COLMAP GUI screenshots, including the full model and an image with SIFT features [14]

that cost of latency does not further that aim. This smaller model worked just as well to incorporate training data images and was three times as fast.

The classifier classified 136 of the 1202 on-policy Genius camera images as baby doll toy images, so those were the images (along with their grasp success/failure marks) that were passed through the SfM binning script. That script was able to incorporate and bin 44 of those images were incorporated, as in Figure 5.4.

(a) 0°

(b) 90°

(c) 180°

(d) 270°

(e) 135°

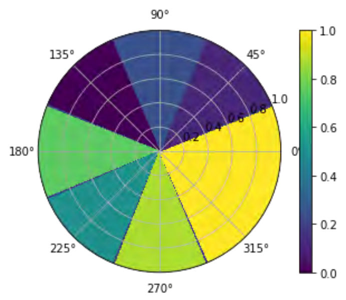
(f) 45°

(g) 225°

(h) 315°

Figure 5.4: Baseline navigation policy images classified as the baby doll in SfM-assigned bins

Each of those images is the first image in a grasp attempt, which each ultimately succeeded or failed to grasp the toy. Thus, we can calculate the baseline policy's grasp success, from these images, as 59%, and we can consider the bins that performed better than 59% as our "good" bins.



(a)

Bin	Number of Grasps	Success Rate
0°	1	1.0
45°	9	0.111
90°	4	0.25
135°	1	0.0
180°	4	0.75
225°	8	0.5
270°	8	0.875
315°	9	1.0
Total°	44	0.59

(b) white rows are the "good" bins; dark are not

Figure 5.5: Bin Angle Performance in Baseline Navigation Policy

5.3.2 Test-Time Performance

A navigation policy that only grasps at angles from {0°, 180°, 270°, 315°} is, in a sense, drawing from a population of 22 grasps, with a 91% chance of success, as opposed to the baseline navigation policy represented by these 44 grasps that have a 59% chance of success. This difference is significant with $p < 0.05$, so we have reason to hope that the robot's circling navigation policy will significantly outperform its baseline navigation policy in grasp success.

To test this hope, we actually implemented and ran the circle navigation policy. Our actual numbers from running that policy one baby doll toy are indeed much better than the baseline policy, with 80% success in 20 grasps so far, shown in Figure 5.6.

Bin	Number of Grasps	Success Rate
0°	3	1.0
180°	2	1.0
270°	6	0.66
315°	9	0.77
Total	20	0.8

Figure 5.6: Bin Angle Performance in Circle Navigation Policy

5.4 Conclusions and Future Work

We validated the hypothesis of Fall 2018 by actually doing it: we implemented a navigation policy based on its grasping task experience, and running that policy indeed improved its grasp success.

There are several directions to naturally extend this work. First and foremost, this method should be generalized to other toys. To do this, pose evaluators for other toys must be developed, and then tested in single-toy environments as we did for this baby doll toy. Then we will also have the opportunity to try out similar navigation schemes with multiple toys in the scene, where the robot would use a classifer at test-time as well. We began to explore this option with other toys, and have found already that, encouragingly, the object detector responds well to toys it was not specifically trained on. This means that the baseline policy is more generalizable than expected, and will make testing this method on additional suitable toys easier. Once more toys are incorporated into this idea, we can evaluate not only by grasp success, but by a truer measure of success for this kind of task - task completion time for the task of decluttering the entire room of toys.

Another area to develop is developing pose evaluation that can handle different distortions of the baby toy, to help with higher image incorporation rates. Incorporating more images is important because that allows the navigation policy to learn from more of its training data, and it improves performance at task time, since potentially good angles are automatically dismissed when the angle cannot be determined since the image could not be incorporated into the SfM model. Currently, if the baby toy is turned on its other side, the current SfM model cannot incorporate it into its model. One way to solve this could be having a classifier that distinguishes not just between the nine different toys, but also between their different distortions. Such a classifier would distinguish between the face-up and face-down toy, choosing between the SfM model that was created in this work and a hypothetical model created based on a video of the baby in a different pose. Another option would be a more flexible pose evaluation method; for instance, one that does not use floor features at all. We explored creating SfM models from images with the backgrounds segmented out, but SfM was unable to converge using only the features on the toy with the number of images that we had, so we would need more images, or a different featurizer than SIFT, to make that work.

Third, this work simply used the baseline grasping policy, but this method should work for other grasping policies – for instance, one of the RL grasping policies trained in the previous chapter – as well. Theoretically, this method should work for any grasping policy as long as the data used to teach the pose evaluator uses the same policy as the pose evaluator is returning results for.

Our experiments so far fit into the following reasonably generalizable story:

1. Train a bounding box object detector on the toys in your environment that you want your robot to handle.
2. Explore your environment with a naive navigation policy and grasping policy that uses that bounding box, collecting image and success data.
3. Take pictures of the toys in your environment and train a toy classifier that your robot can use to train and choose the desired pose evaluator.
4. Create a pose evaluator, using a little more knowledge of each toy, as well as results of the exploration in (2). For this baby doll toy, this involved taking a video of the toy, creating a SfM model for this toy from this video, and then using it to analyze the training data collected in (2), classified by (3), on the grasping policy that used and plan to use with the modified navigation policy. Ultimately, this evaluator must be able to, at test time, take the toy image and return a boolean of whether to attempt this grasp.
5. Use that evaluator in a navigation policy: Navigate to the toy, classify the toy, use the evaluator assigned to that toy, and either grasp or return to navigation based on that result.

Our positive results point to our larger overall theme that keeping our skills less disparate and better leveraging our flexibility allows us to achieve more success.

6 Acknowledgments

This work was conducted with the mentorship of Ph.D candidate Gregory Kahn, Post-doc Chelsea Finn, and Professors Aviv Tamar and Pieter Abbeel. Thank you to undergraduates Anna Zhu and Sam Lowe that helped last year with the project setup: Anna Zhu especially with toy detection and the baseline navigation script, Sam Lowe especially with the data collection setup, and both with assembly and maintenance of our dear but occasionally disagreeable robot. Finally, thank you to masters candidate Jack Sullivan, for co-conducting the data collection, analysis, and writing for the work done in Fall 2018.

Bibliography

1. C. U. et al. “Autonomous driving in urban environments: Boss and the urban challenge”. In: *Journal of Field Robotics*, 2008.
2. S. Bazeille, E. Battesti, and D. Filliat. “A Light Visual Mapping and Navigation Framework for Low-Cost Robots”. In: *Journal of Intelligent Systems, 2015, pp.27.* 2015.
3. A. Bihi and V. Kumar. “Robotic grasping and contact: a review”. In: *IEEE International Conference on Robotics and Automation. Vol. 1. IEEE, pp. 348-353*, 2002.
4. J. Bohg, A. Morales, T. Asfour, and D. Kragic. “Data-driven grasp synthesis—a survey.” In: *IEEE Transactions on Robotics*, 2014.
5. S. Chitta, I. Sucan, and S. Cousins. “MoveIt! [ROS Topics]”. In: *IEEE Robotics Automation Magazine* 19:1, 2012, pp. 18–19. ISSN: 1070-9932. DOI: 10.1109/MRA.2011.2181749.
6. F. Chu, R. Xu, and P. A. Vela. “Real-world Multi-object, Multi-grasp Detection”. In: *arXiv:1802.00520v3*, 2018.
7. M. Cummins and P. Newman. “Fab-map: Probabilistic localization and mapping in the space of appearance.” In: *The International Journal of Robotics Research, 27:647–665*, 2008.
8. M. A. Fischler and R. C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24:6, 1981, pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: http://doi.acm.org/10.1145/358669.358692.
9. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. In: *CoRR* abs/1704.04861, 2017. arXiv: 1704.04861. URL: http://arxiv.org/abs/1704.04861.
10. J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. *Speed/accuracy trade-offs for modern convolutional object detectors.* 2017.
11. G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine. “Self-supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation”. In: *arXiv:1709.10489*, 2017.
12. D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. “QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation”. In: *CoRR* abs/1806.10293, 2018. arXiv: 1806.10293. URL: http://arxiv.org/abs/1806.10293.

13. S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. “Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection”. In: *arXiv:1603.02199v4*, 2016.

14. D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60:2, 2004, pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

15. M. T. Mason and J. K. Salisbury. “Robot Hands and the Mechanics of Manipulation”. In: *MIT Press, Cambridge, MA.* 1985.

16. D. Morrison, P. Corke, and J. Leitner. “Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach”. In: *arXiv:1804.05172v2*, 2018.

17. L. U. Odhner, R. R. Ma, and A. M. Dollar. “Open-Loop Precision Grasping With Underactuated Hands Inspired by a Human Manipulation Strategy”. In: *IEEE Transactions on Automation Science and Engineering, Vol. 10, No. 3*, 2013.

18. L. Pinto and A. Gupta. “Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours”. In: *arXiv:1509.06825v1*, 2015.

19. A. Saxena, J. Driemeyer, J. Kearns, C. Osondu, and A. Ng. “Learning to Grasp Novel Objects Using Vision”. In: *Vol. 39 of Springer Tracts in Advanced Robotics. Springer, Berlin, Heidelberg, Ch. 4, pp. 33-42*, 2008.

20. J. L. Schönberger and J.-M. Frahm. “Structure-from-Motion Revisited”. In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2016.

21. J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. “Pixelwise View Selection for Unstructured Multi-View Stereo”. In: *European Conference on Computer Vision (ECCV).* 2016.

22. C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer. “Vision and navigation for the Carnegie-Mellon Navlab”. In: *TPAMI*, 1988.