With in hand Manipulation Planning using Deep Reinforcement Learning Techniques

by

Chinmay Burgul

Advisor : Jing Xiao
Berk Calli

December 2019

Contents

1 Abstract

Within Hand Manipulation(WIHM) is seen as a good skill in Robotic Manipulation tasks which reduces manipulation complexity in scenarios where object to be manipulated needs to be re-oriented or re-positioned to perform a given task. The WIHM system involves multiple Degrees of Freedom, multiple points of contact and varying friction forces. This makes the kinematic and dynamic modeling of the system a complex task. Inaccurate mathematical model of the system will not suffice for accurately planning a trajectory using conventional control system. Hence, This project explores a model-free approach for planning of WIHM tasks. A well-known algorithm of Deep Q-Network was successfully implemented on a Variable Friction Finger WIHM task for a single goal problem and results are explained further.

2 Introduction

One of the long-standing goal of a roboticist is to create a general purpose robotic system that can perform wide variety of tasks in human-centered environments such as hospitals and homes. For a robot to perform a wide variety of tasks requires high-versatility like Within-Hand Manipulation(WIHM), Dexterous Manipulation in small work volume, Object Recognition in a cluttered environment, Navigating in an unknown environments.

In this paper we focus on versatility in terms of Within-Hand Manipulation. Looking at the ways human performs manipulation tasks. The Anthropomorphic design of human hand has innate capabilities of Within-Hand Manipulation because of the multi-finger mechanism and god-gifted control algorithm. Previously there were several attempts of bio-mimicking this capability. Mimicking the anthropomorphic mechanism was nearly achieved using under-actuated mechanism but mimicking the god-gifted control algorithm is still a far to reach goal.

People are working on to achieve this control goal from long time on anthropomorphic mechanism. In this paper we are approaching the mimicking of WIHM capabilities with simpler mechanism of friction finger which is taken from Yale OpenHand [13] Model VF and the motivation for the design is explained in [25]. [6] has worked on controlling and planning WIHM tasks with combining Graph search algorithm and Visual Servo mechanism. This research focuses on model free Reinforcement Learning algorithm which learns the dynamics of the system by exploration. The advantage of model free approach is it can learn to generalize WIHM for various objects. Few drawbacks of this approach is it requires a high amount of samples and time to explore the environment and learn a particular task. Previous research [28][20] has come up with impressive solution of learning from demonstrations, Multi-goal Reinforcement Learning [2] which increases the efficiency of learning and decreases the need for huge samples.

Generalizing WIHM control algorithm will take us one step closer to achieve

the control goal of versatility in terms of WIHM. This will provide translation, orientation and fine-positioning of the object within the hand work-space. In this research we focus on developing a model-free Reinforcement Learning algorithm for controlling WIHM on friction finger setup and generalize the policy for different objects. We also add Multi-goal Reinforcement learning, and learning from demonstration techniques to decrease the samples requirement and learn the tasks faster.

3 Related Work

3.0.1 Model based trajectory optimization

Dexterous Manipulation tasks have been successfully achieved by using model based trajectory optimization methods [14][19][11] in simulation domains especially when the dynamics can be adjusted and relaxed to make them more tractable. These methods depends on accurate dynamic models and state estimates which are difficult to obtain for contact rich manipulation tasks, in the real world. And these approaches struggle to translate to real world manipulation as learning complex models on real world systems with significant contact dynamics is difficult.

3.0.2 Model Free Reinforcement Learning

Model Free Reinforcement Learning Methods[22][26] and Deep Reinforcement Learning Methods[24][12] do not require dynamic modeling of the system. They will know the configuration of the system by exploring and function approximating a neural network. In-fact, they are function approximating the dynamical model of the system by learning from experiences. But for this they require a huge sample space. Few methods overcome this need by using simple policy representation [17][10]. But to solve real world robotics tasks more complex representation is required to feed rich sensory information.

3.0.3 Imitation Learning

In this, demonstrations of successful behaviour are used to train policies to imitate the expert [23]. One approach of Imitation Learning is Behavior Cloning(BC), which learns a policy through supervised learning to mimic the demonstration state-action pairs. BC has been applied successfully in autonomous driving[18][3], quad-copter navigation[5], locomotion[15][8]. But it suffers from problems related to distributional drift and outside the space of demonstration data. Dataset Aggregation (DAGGER) [21] mixes the learned and expert policies iteratively and augments the dataset to solve the accumulating error problem. It requires the expert to be available during training to provide additional feedback to the agent and makes it difficult to use. Pure Imitation Learning or BC are limited and cannot exceed the capabilities of the demonstrator since they lack the

notion of task performance and environment. They under-perform or perform similar to the demonstrations.

Inverse optimal control or Inverse Reinforcement Learning (IRL) [16] is other approach of Imitation learning in which agent learns the reward function from optimal demonstrations. Few achievements of IRL are navigation [30], autonomous helicopter flight [1] and manipulation [4]. Our focus is not just to use imitation learning rather use imitation learning for bootstrapping the process of reinforcement learning. This bootstrapping helps to overcome exploration challenges of RL and RL learns the policy to improve based on actual task objective.

3.0.4 Leveraging Reinforcement Learning with demonstrations

In this research, we are planning to combine imitation learning with Reinforcement learning. The Imitation learning bootstraps the initial learning process and gives a idea of task objective this overcomes the exploration challenges and the need to explore the whole task space, while RL fine tuning allows the policy to improve based on actual task objective.

Methods based in dynamic movement primitives (DMPs) [17][9][7] have been used to effectively combine demonstrations and RL to enable fast learning. Most of the methods use trajectory-centric policy representations, which is well suited for imitation but not enable feedback on rich sensory inputs. Such methods have been applied to some dexterous manipulation tasks [27], the tasks are primitive and simpler. Using expressive function approximators allow for complex, non linear ways to use sensory feedback, making them well-suited to dexterous manipulation. [28][20] have successfully demonstrated the work on Leveraging Reinforcement Learning with demonstration.

3.0.5 Multi Goal Reinforcement Learning

3.0.6 Techniques to converge policies faster

- BC from planner or demonstration
- Abstraction of : state/action abstraction, automatic skill recovery, planning over a space of learned skills.
- Learning a model and planning : observation-space process models, latent state space models, using planning algorithms.
- Manipulating with tactile/force feedback
- Modifications for training
- Modification for training in demonstration

3.0.7 Learning Dexterous Within hand Manipulation with Reinforcement Learning

The work from [29] has successfully demonstrated Reinforcement learning with demonstration on anthropomorphic Dexterous manipulation in real world.

3.0.8 Friction Finger

The gripper design is taken from Yale OpenHand Project [13]. The motivation for the design is explained in [25]. The translation steps of the object on Friction Fingers is shown in Fig.5 and orientation of the object is shown in Fig.6. Controlling of WIHM object is been done in [6] which is shown in Fig.7 1. Offline Planning using Graph search Algorithm, 2. Visual Servo and Hybrid of (1) and (2). Parameters of the Friction Fingers are shown in Fig.4

3.1 Background

3.1.1 Within Hand Manipulation using Friction Finger

Hardware Environment Setup : It consists of YALE Open-Hand Gripper Model VF, Camera and structure frame as shown in the Fig.1 taken from [6]. The variable friction has two modes to enable and disable friction shown in Fig.2.

Software Environment Setup The Kinematic model of the Variable Friction Finger System done in [6] is used to create the environment for the agent to interact with. The environment is created on pyglet gaming engine and openai functionalities. This environment gives the current state of the system to the agent, and when agent takes an action, it will return a reward and next state to the agent.

4 Thesis statement

Objective:6.

- Within hand motion planning of object using Model-Free Algorithms
- To generalize the policy for variety of objects and asymmetric objects.
- Generalizing the network for planning of Orientation and Translation of object Within-Hand.
- Get new insights from the planned path.
- Explore new learning techniques in Robotics

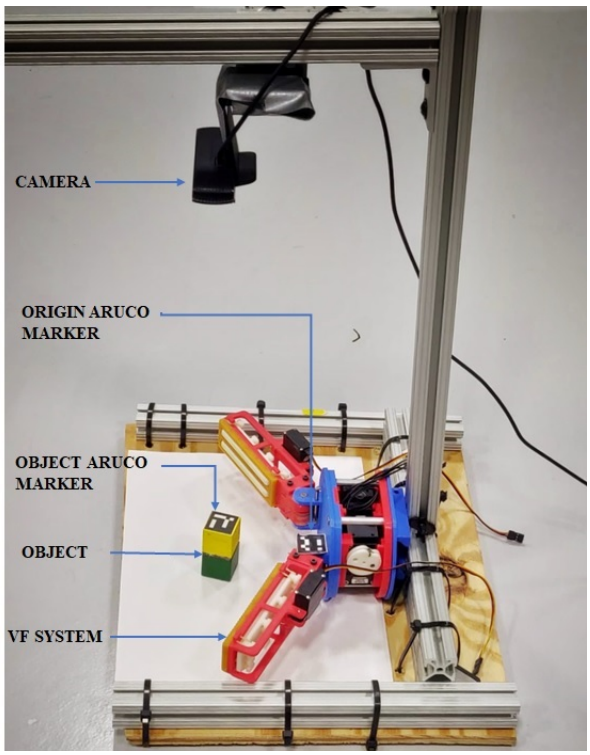

Figure 1: Hardware Setup

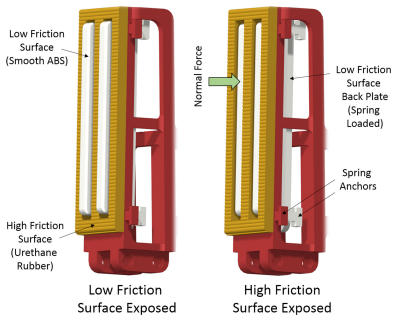

Figure 2: Finger Setup

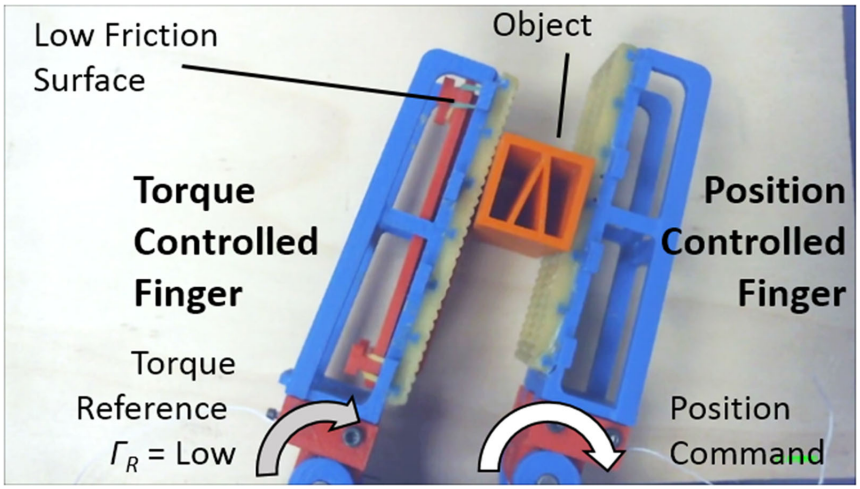

Figure 3: Figure explaining Friction Finger Actions Setup. The Left finger is in torque control mode and Right Finger is in Position Control mode, low friction on left finger and High Friction on Right Finger. When right finger rotates clockwise, the object slides upwards on left Finger Surface.

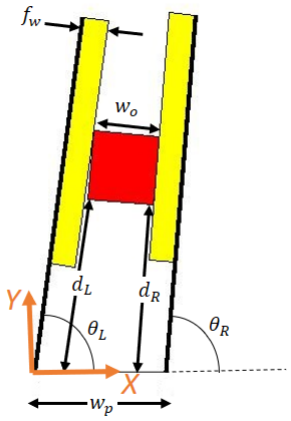

Figure 4: Parameters of Friction Finger shown from Top View

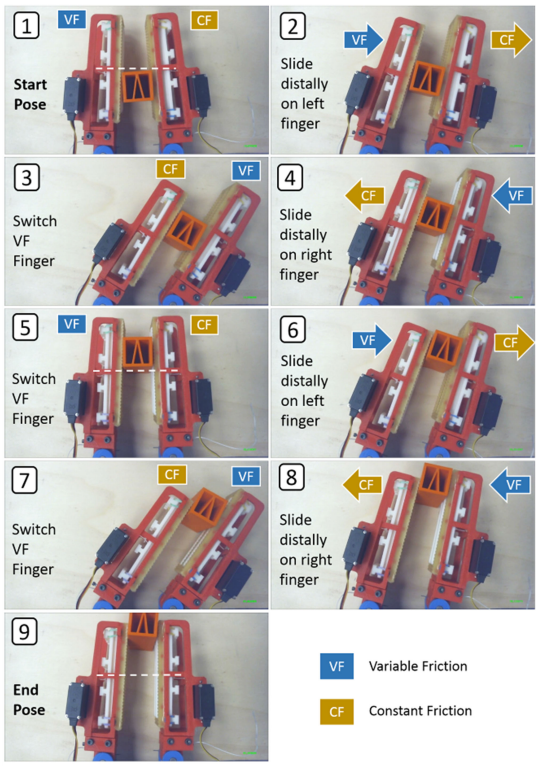

Figure 5: Translation of Object

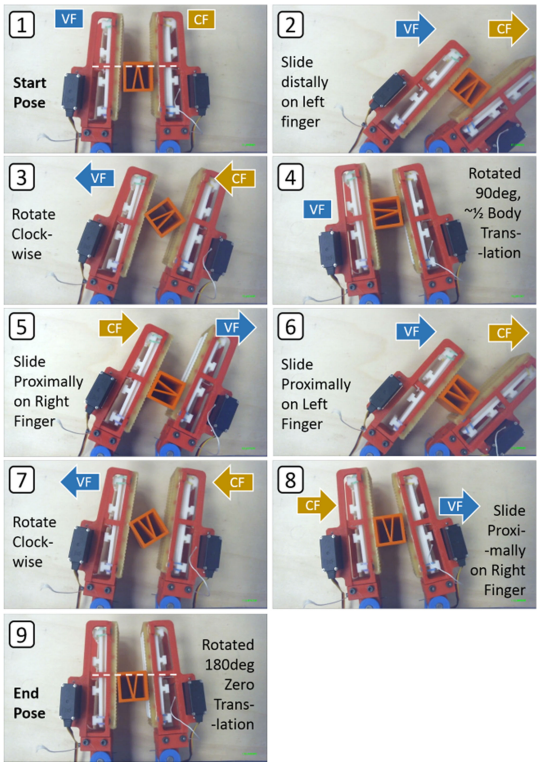

Figure 6: Rotation of Object

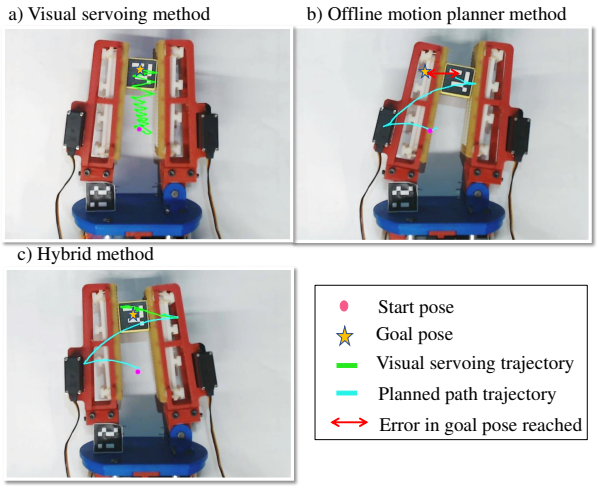

Figure 7: Trajectory tracked by the object from start pose (7,7,0) to goal pose (12,12,0) for different methods a) Visual servo b) Offline motion planning c) Hybrid method

5 Approach/Methodology

The approach of this research is solving the Within-hand Manipulation task using model-free Reinforcement Learning Algorithm of Deep Deterministic Policy Gradient (DDPG) and Deep Q-Networks. DDPG is suitable for continuous action-space and has got good results in manipulation and grasping tasks. DQN is a discrete action-space algorithm and is widely used for solving gaming environments. Both of these algorithms are off-policy and enables us to store the demonstration data and train offline even when it is not collecting the samples. With the offline learning enabled we can use Hindsight Experience Replay and Learning from Demonstration to accelerate the training process as mentioned in the related work.

Experience tuple consists of :

Continuous action space :
State (S) : (θ_{left},θ_{right}, O_x, O_y, G_x, G_y, *on goal*)
$Action(A)$: ($\Delta\theta_{left}$,$\Delta\theta_{right}$)

$$Reward(R) : \begin{cases} 0, & \text{if } \textit{distance(goal, object position)} \leq \textit{10mm} \\ -1, & \text{otherwise} \end{cases} \quad (1)$$

Discrete action space
State (S) : (θ_{left},θ_{right}, O_x, O_y, G_x, G_y, *on goal*)
$Action(A)$: (a_1, a_2, a_3, a_4)

$$Reward(R) : \begin{cases} 0, & \text{if } \textit{distance(goal, object position)} \leq \textit{10mm} \\ -1, & \text{otherwise} \end{cases} \quad (2)$$

Where,
θ_{left} : $Angle\ of\ left\ finger\ and\ X-axis$
θ_{right} : $Angle\ of\ right\ finger\ and\ X-axis$
O_x, O_y : $Object\ Initial\ Position$
G_x, G_y : $Goal\ Position$
$\Delta\theta_{left}$: $Change\ in\ \theta_{left}$
$\Delta\theta_{right}$: $Change\ in\ \theta_{right}$
a_1 : $Sliding\ upwards\ on\ Left\ Finger$
a_2 : $Sliding\ downwards\ on\ Left\ Finger$
a_3 : $Sliding\ upwards\ on\ Right\ Finger$
a_4 : $Sliding\ downwards\ on\ Right\ Finger$

State Space : The state space representation consists of Finger angles, Object's Initial Position, Goal Position and friction mode. This is the information of the robot passed to the agent by the environment.
Action Space : Action space consists of Change in angle of left finger, which is the actuation part. It is a continuous action space. During this action, if the Change in angle is positive the right finger will be in Position control mode and Left finger will be in Torque control mode. And if the Change in angle is

negative the left finger will be in Position Control mode and Right Finger will be in Torque Control mode.
Reward : Sparse reward and Reward Engineering approaches are tried to see the results. In Sparse Rewards, for all the time steps the reward is negative one until the object's initial position is in 10 mm tolerance with the Goal Position. In Reward Engineering the reward at every time step is negative of distance between object position and goal position.

6 Progress Report and Results Fall 19

1. Software Environment : A Simulation Environment framework was created in python script. The environment uses Mathematical Forward Kinematic Equations of the system and displays the visualization effect of action taken on the system at every time step. Using this visualisation we can see the performance of the system and understand the intuition of a path planned by the policy. The environment also plots the trajectory of the object along with Start Point and End Goal Point. Fig[8] shows the Friction Finger Software Environment Visualization.

2. Deep Deterministic Policy Gradient(DDPG) Algorithm was used to train a policy to reach a single goal target point. The Environment had few actions continuous for motor actuation and few actions as discrete for friction mode selection. The DDPG agent was giving all the outputs as continuous and friction selection action was not learning. Hence, we decided to move to discrete action space algorithms like DQN.

3. Deep Q-Network(DQN) Algorithm a discrete action space algorithm was used to train a policy to successfully reach a Single Goal given to the agent. The agent was successful in planning a path to reach a target point. Figure[9] shows a Trajectory of a path. Video of trained policy reaching a given goal point is shown in this link shorturl.at/cdkpv as score Vs episodes plot is shown in Fig[11]

4. Multi-Goal Reinforcement Learning Technique called Hindsight experience replay was used along with DQN to train the model for multi-goal problem. Initially testing of this algorithm was done on Bit Flipping Environment and was successfully trained to flip bits to reach a goal state. The results is shown in Fig[10][12][13]. The same algorithm is tested initially on Friction Finger Environment. The policy was not converging initially and needs more tuning parameters and debugging of code. The debugging of code is in progress.

5. Hardware Setup of the Friction Finger environment was built and software integration was successfully done. Integrating the software environment to train a policy on hardware environment is in progress.

7 Future Goals

- Successfully train a Multi Goal Reinforcement Learning Agent

Figure 8: Friction Finger Environment Visualization. The goal is represented by green box, object by purple box and Left and right fingers are represented by Yellow Rectangular boxes

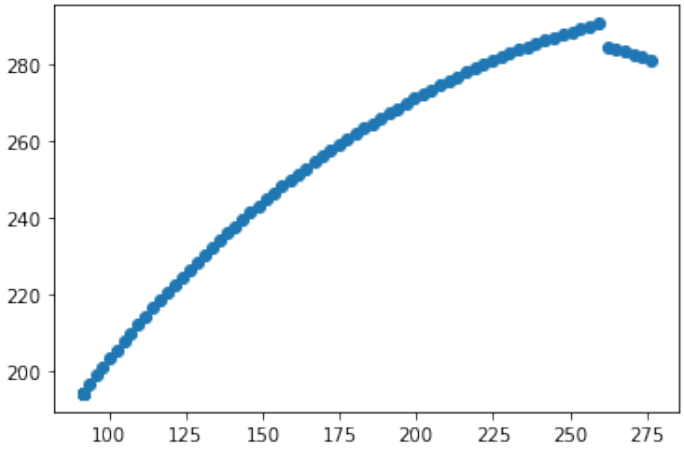

Figure 9: Path Trajectory from Start(Top-Right) to Goal(Bottom Left) and a break in while changing the friction finger mode

```
Initial State:  [0 0 0 0 0 1 1 0 0 1 1 1 1 1 1]
Goal:   [1 0 0 0 0 0 0 0 1 1 1 1 0 0 0]
State at step 0: [1 0 0 0 0 1 1 0 0 1 1 1 1 1 1]
State at step 1: [1 0 0 0 0 1 1 0 1 1 1 1 1 1 1]
State at step 2: [1 0 0 0 0 1 1 0 1 1 1 1 0 1 1]
State at step 3: [1 0 0 0 0 0 1 0 1 1 1 1 0 1 1]
State at step 4: [1 0 0 0 0 0 1 0 1 1 1 1 0 0 1]
State at step 5: [1 0 0 0 0 0 1 0 1 1 1 1 0 0 0]
State at step 6: [1 0 0 0 0 0 0 0 1 1 1 1 0 0 0]
Success!
Press enter...
```

Figure 10: Bit Flipping of 15 Bit Array flipping a one bit at a time step and reaching a goal state

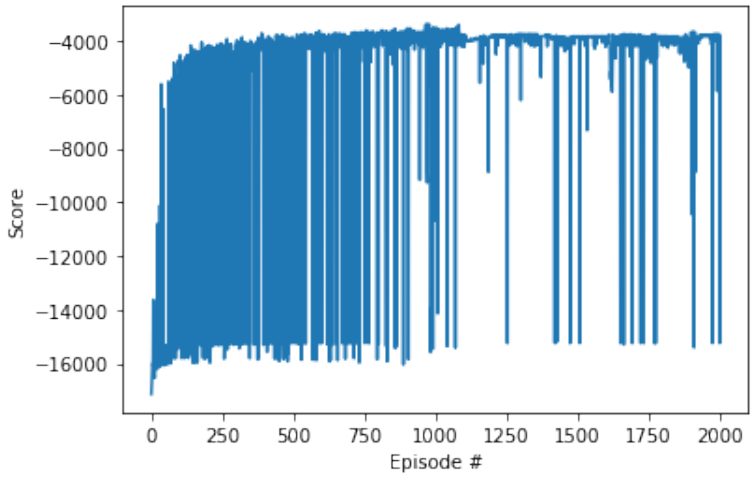

Figure 11: Figure shows increase in rewards Vs episodes in Friction Finger Environment

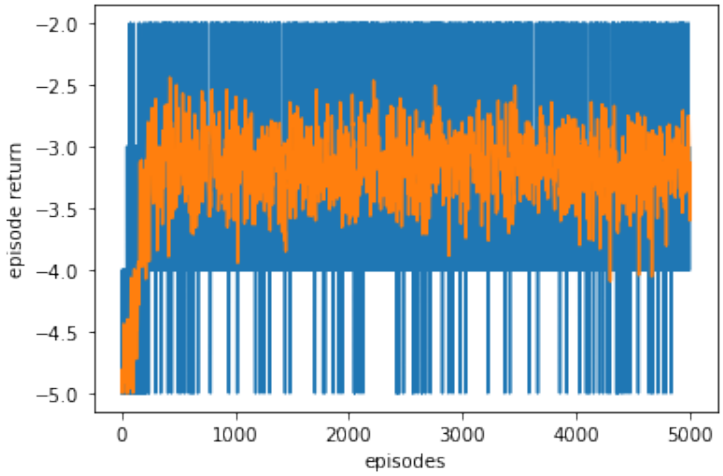

Figure 12: Figure shows increase in rewards Vs Episodes and orange plot is a mean of episode return in Bit flipping Environment

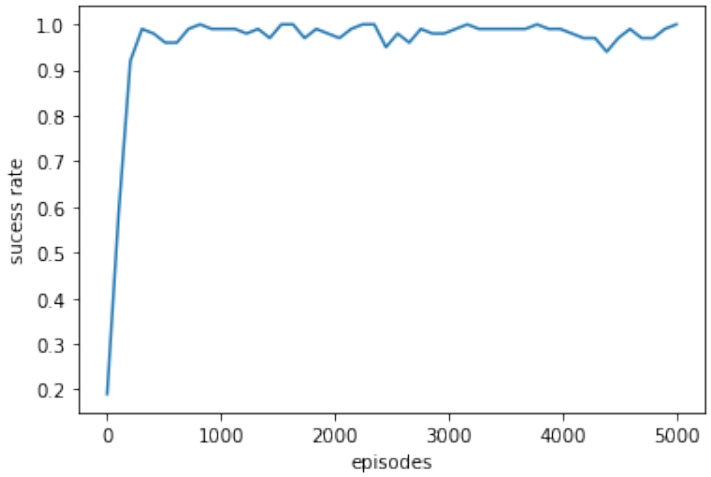

Figure 13: Figure shows Success Rate Vs Episodes in Bit flipping Environment

- Create a Hardware Environment to train and work with real world scenario.
- Generalize the policy for symmetric and asymmetric object
- Train a Policy for Change in Orientation of Goal.
- Train a Policy to combine change in Translation and Orientation of Goal.

8 Work Plan and Time Table

N.o	Tasks	Dates	Status
1	Software Environment setup ready	Oct 25	Done
2	Implement Vanilla DDPG on Software Env	Nov 10	Change
3	Implement DQN in Software Env	Nov 23	Done
4	Implement DQN + HER in Software Env	Dec 12	Done
5	Hardware Setup ready	Dec 12	Done
6	Create a Hardware Environment	Dec 25	To Do
7	Get Demonstration trajectories in Hardware Env	Winter	
8	Train DQN + Demonstrations in Hardware Env	Winter	
9	Create a Change Orientation of Goal in Env and train	Winter	
10	Generalize for multiple objects and shapes	Spring	
11	Tests and Results	Spring	
12	Thesis Defense	-	

9 Conclusion

- Planning a real-time accurate path
- Giving new insights for Path Planning
- Eliminates need for correction methods like Visual servo and hybrid method
- Framework can be used to generalize for different object shapes and non-uniform objects

10 Progress Report and Results Spring 20

1. Exploring different state space representation and reward methods

The initial part of the training needed to play with parameters and there are multiple parameters which makes a huge combination of a unique set of parameters to begin. Hence, I decided to play with necessary parameters like network architecture, reward engineering, state space representation. I followed 4 approaches to train a policy to reach a single goal single start task.

(a). Approach 1: In this approach, the state space representation has theta, object distance, distance between object and goal, and on goal parameter and reward is engineered as negative of distance till object reaches goal and positive one every step after it reaches the goal.

State (S) : $(\theta_{left}$,$\theta_{right}, O_x, O_y, dist_x, dist_y, on\ goal)$

Reward (R) : Engineered

(b). Approach 2: In this approach, the state space representation has theta, object distance, goal position, and on goal parameter. The previous approach had distance between goal and object but here we tried to include goal's position in the state space. Reward is negative of distance till object reaches the goal and positive one every step after it reaches the goal.

State (S) : $(\theta_{left}$,$\theta_{right}, O_x, O_y, G_x, G_y, on\ goal)$

Reward (R) : Engineered

(c). Approach 3: In this approach, the state space representation has has theta, object distance, distance between object and goal, and on goal parameter the difference in this approach from previous approach is sparse rewards. That is negative one reward until object reaches the goal and zero for every step after it reaches the goal.

State (S) : $(\theta_{left}$,$\theta_{right}, O_x, O_y, dist_x, dist_y, on\ goal)$

Reward (R) : Sparse

(d). Approach 4: In this approach, the state space representation has theta, object distance, goal position, and on goal parameter the difference in this approach from previous approach is including goal position in the state space and sparse rewards. That is negative one reward until object reaches the goal and zero for every step after it reaches the goal.

State (S) : $(\theta_{left}$,$\theta_{right}, O_x, O_y, G_x, G_y, on\ goal)$

Reward (R) : Sparse

The score vs episodes plot is shown in Figure [14] and comparing all the results, we can comprehended that model is learning faster with giving goal position in the state space instead of distance between goal and object. The policy is getting stabilised sooner with reward engineering as compared to sparse reward.

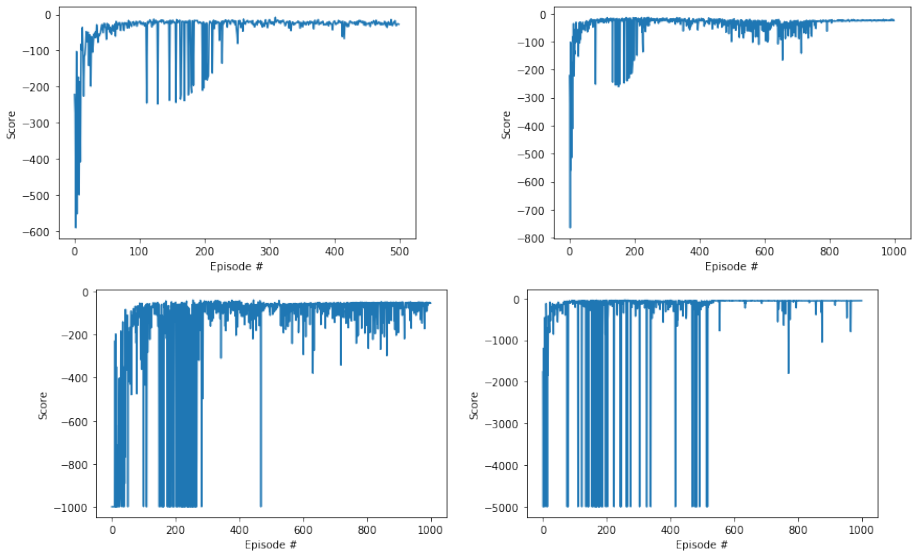

Figure 14: Score vs Episode **(a)** Approach 1 **(b)** Approach 2 **(c)** Approach 3 **(d)** Approach 4

The trajectory generated from the policy for same start and end goal of each approach is plotted in x-axis vs y-axis in order and shown in the Figure [15].

2. Training with DQN + HER with split approach

The second part is trying to train for multi-goal single-start using DQN + HER. The policy was initially not getting trained for multiple goals from a single start point. We followed an approach of divide and train. In this approach the whole work space was divided in several parts and we tried to train a policy for each part to reach the goal. Given a goal point, if the goal point lies in a particular part of the work space we will call the policy trained specific to that part and execute the trajectory. But training for all the parts was happening as expected and the results are shown in Fig [16] for Score vs Episodes and Average loss vs Episodes for a small part I. Further similar results of training in a II part is shown in Fig [17] where the policy was not converging even after 10000 episodes. The figure shows score vs episodes and Average loss vs Episodes. Figure [**??**] shows possible trajectories formed using the above approach.

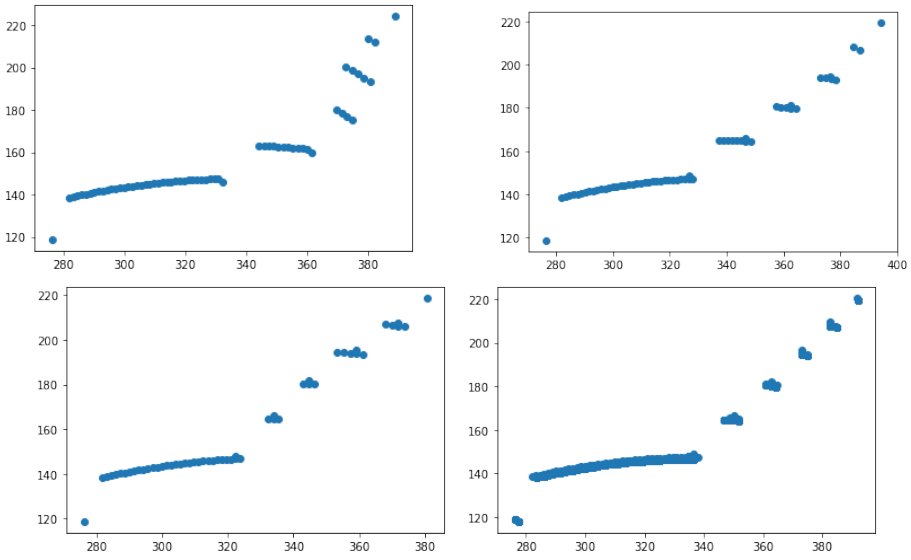

Figure 15: Trajectory plotting in x-axis vs y-axis **(a)** Approach 1 **(b)** Approach 2 **(c)** Approach 3 **(d)** Approach 4

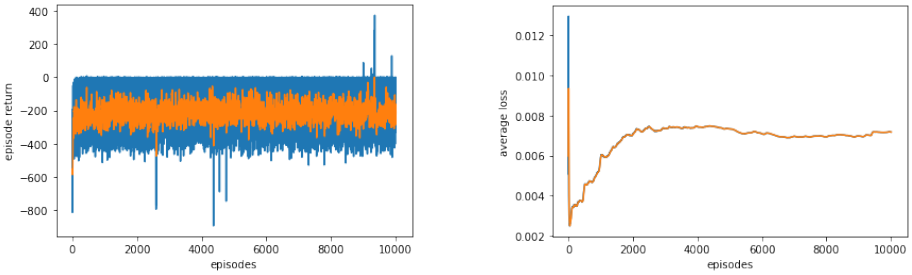

Figure 16: Discrete DQN + HER Part I **(a)** Score vs episodes **(b)** Average loss vs episodes

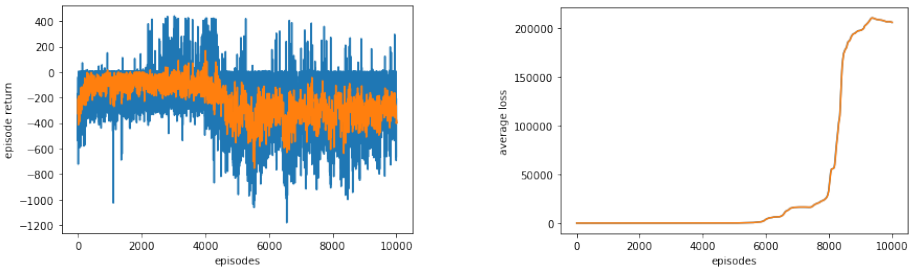

Figure 17: Discrete DQN + HER Part I **(a)** Score vs episodes **(b)** Average loss vs episodes

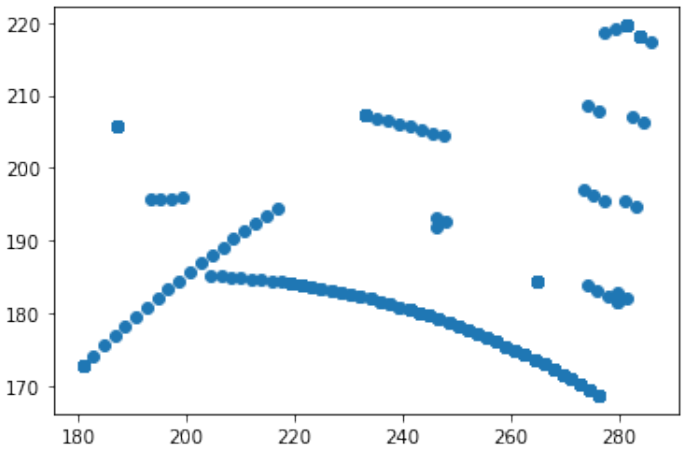

Figure 18: Failure to reach a goal for Multi goal and starting from a Small Part

3. Training with PPO

In this part of the research we used stable baselines to train the policies. Among several algorithms in stable baselines the recent algorithm named proximal policy optimization was used to train on the environment. To start with training we need to fix with initial parameters like network and time steps to train. We explored the training process with different parameters and results of the few parameters are shown in Figure [19]. The training was done for a single goal and single start task.

We comprehend that the Network [256, 256] and 256 time steps is better than the other, as the model converges faster and reaches stability sooner than the other parameters.

4. Single start multi Goal

In this part of the research, we focused to select a network architecture to train a policy for Multi goal single start task using a PPO2 policy. The Figure [20] is a plot of Score vs Episodes for a) network [64, 64] b) network [128, 128] and c) network [256, 256]. Here the motto was to understand and pick which network to be trained for 50,000 episodes. This experiment was conducted for only 500 episodes.

Later on, after the network architecture[256, 256] is selected we trained for 50,000+ episodes with different rewarding system. Figure [**??**] shows plots of Score vs Episodes for (a) Reward of +1 was given for every time step after the object reaches the goal and negative of distance between goal and object was given until the goal is reached. (b) Reward of 0 is given for every time step after the object reaches the goal and negative of distance between goal and object was given until the goal is reached.

The model converges efficiently and a trajectory can be generated to reach

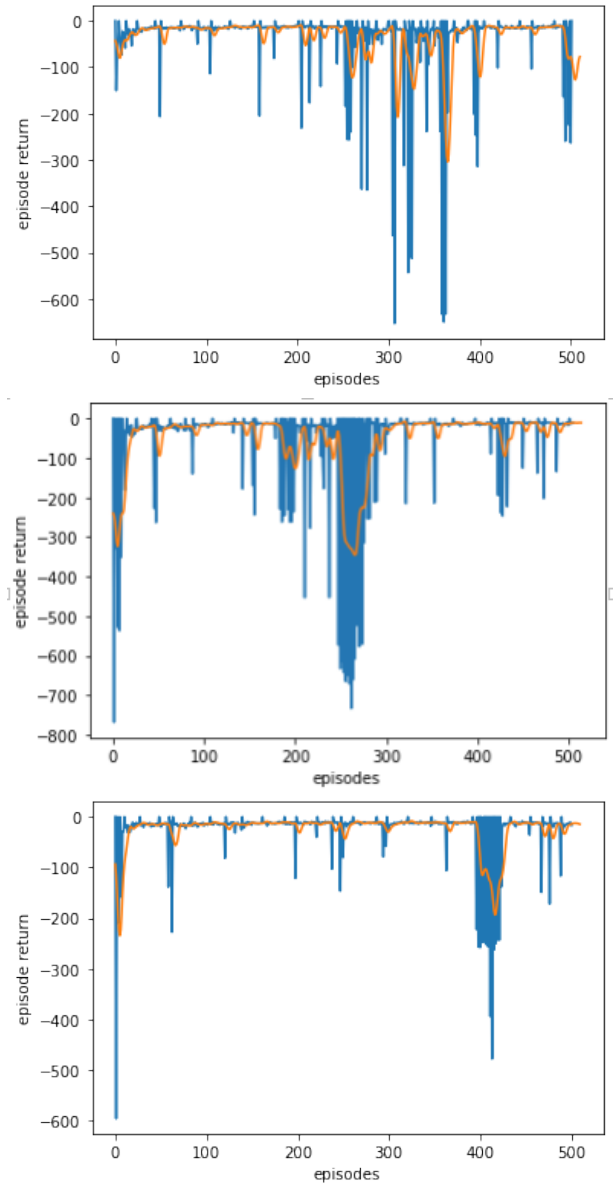

Figure 19: Plots of Score vs Episodes for training with PPO2 to explore optimal parameters for time steps and networks **(a)** Network [128, 128] and 128 time steps **(b)** Network [128, 128] and 256 time steps **(c)** Network [256, 256] and 256 time steps

any part of the work space from a particular start point. The video of the object reaching a goal can be seen on this link http://tiny.cc/mfd2oz

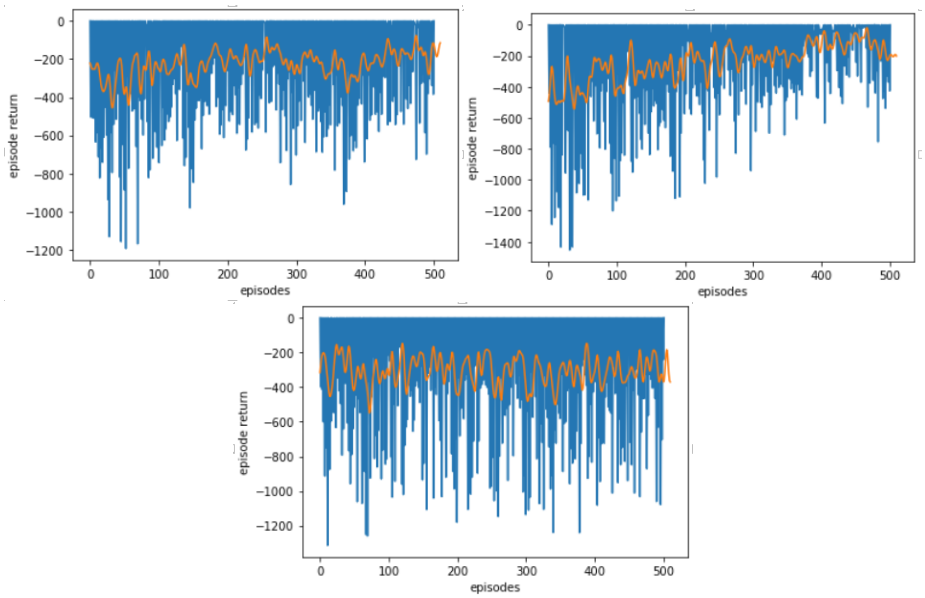

Figure 20: Plots of Score vs Episodes for training with PPO2 to explore optimal network architecture **(a)** Network [64, 64] **(b)** Network [128, 128] **(c)** Network [256, 256]

5. Multi start multi goal

In this part of the project, the motto was to train a policy for multi start and multi goal. We tried training a single policy but it did not converged. Hence, we tried a different approach to split the work space in 6 different parts and train it for multi goal task. If given the start point is within a particular part of the work space then we use the policy specifically trained for that part to generate a trajectory. Here, we divided the whole work space in 6 different parts as following.

Part-I : theta left ranges from 90-110 degrees and dl ranges from 35-55 mm
Part-II : theta left ranges from 90-110 degrees and dl ranges from 55-75 mm
Part-III : theta left ranges from 90-110 degrees and dl ranges from 75-110 mm
Part-IV : theta left ranges from 70-90 degrees and dl ranges from 35-55 mm
Part-V : theta left ranges from 70-90 degrees and dl ranges from 55-75 mm
Part-VI : theta left ranges from 70-90 degrees and dl ranges from 75-110 mm

Figure [22] shows plots of Score vs Episodes for all the parts. All the parts have different areas and different complexities to reach a goal. The training is not same in all the plots and center parts of the work space are bigger hence it

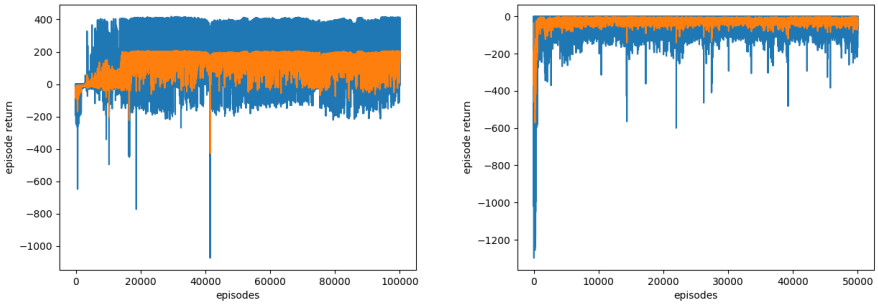

Figure 21: Training with PPO2 for multi-goal successfully with different reward engineering

takes longer time to train a policy for it. The beginning parts are smaller hence it converges faster and reaches stability sooner.

Video of the object reaching a goal can be seen on this links http://tiny.cc/ved2oz http://tiny.cc/idd2oz

6. Orientation task

We tried training for the orientation task with different approaches and different algorithms but it did not worked and policy did not converged. Our plan was to train a policy for orientation and translation separately and create a combined framework to perform translation and orientation tasks.

The common difficulties we faced during training is, there is a very small area in work space where orientation can be performed. When the object reaches that small work space the kinematic formulas cannot solve the unknowns and the episode collapses there. Wherever the kinematic solvers can solve the work space is limiting itself to perform orientation task.

11 Goals Achieved

- Trained a policy to real-time plan a trajectory for multi-start multi goal task
- Got new insights for Path Planning and training approaches.
- Eliminates need for correction methods like Visual servo and hybrid method
- Framework can be easily generalized for different object shapes and non-uniform objects.

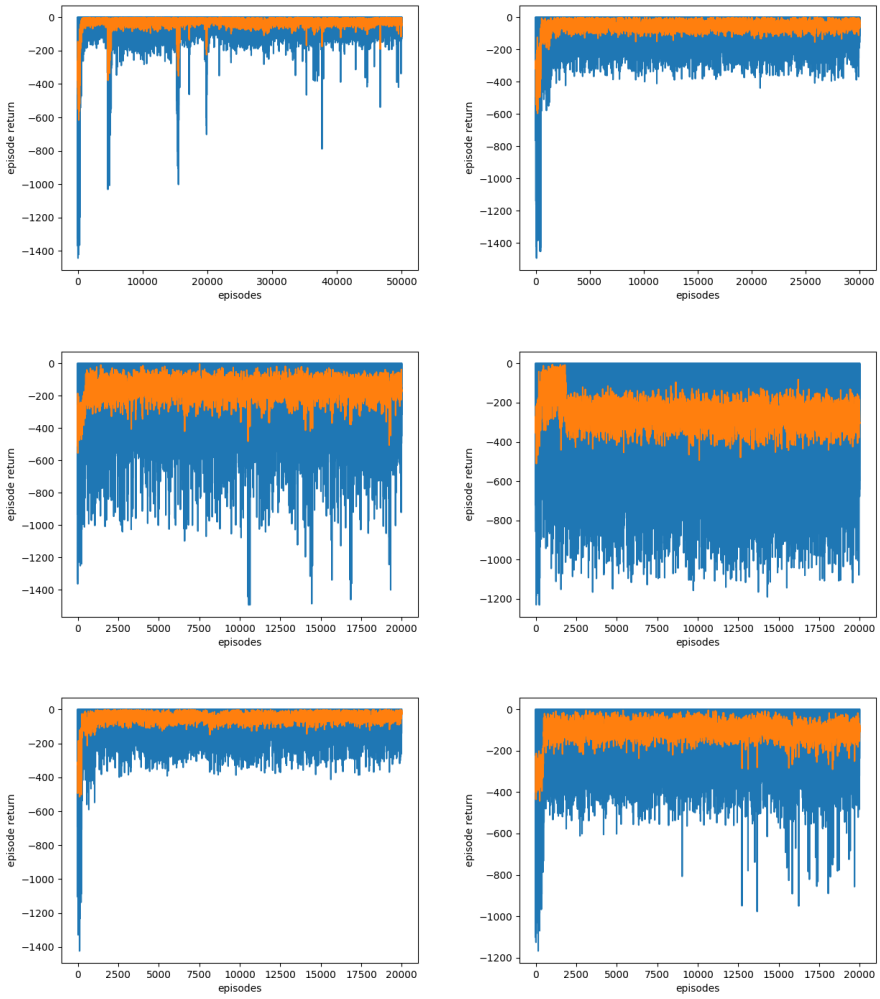

Figure 22: Plots of Score vs Episodes for training with PPO2 for Parts **(1) (2) (3) (4) (5) (6)**

12 Future Work

- Train a policy for Orientation Task
- Work on Hardware training
- Generalize for multiple objects

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016.

[4] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[5] A. Giusti, J. Guzzi, D. C. Cireşan, F. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, July 2016.

[6] Narayanan Gokul, Amrith Raj Joshua, Gandhi Abhinav, Gupte Aditya, Spiers Adam, and Calli Berk. Within-hand manipulation planning and control approaches for variable friction fingers. 2019.

[7] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE, 2002.

[8] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, and Stefan Schaal. Learning locomotion over rough terrain using terrain templates. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 167–172. IEEE, 2009.

[9] Jens Kober and Jan Peters. Learning motor primitives for robotics. In *2009 IEEE International Conference on Robotics and Automation*, pages 2112–2118. IEEE, 2009.

[10] Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856, 2009.

[11] Vikash Kumar, Yuval Tassa, Tom Erez, and Emanuel Todorov. Real-time behaviour synthesis for dynamic hand-manipulation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6808–6815. IEEE, 2014.

[12] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[13] R. Ma and A. Dollar. Yale openhand project: Optimizing open-source hand designs for ease of fabrication and adoption. *IEEE Robotics Automation Magazine*, 24(1):32–40, March 2017.

[14] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 137–144. Eurographics Association, 2012.

[15] Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and autonomous systems*, 47(2-3):79–91, 2004.

[16] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[17] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[18] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[19] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[20] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[21] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[22] R.S.Sutton and A.G. Barto. Reinforcement learning: An introduction. I and II.

[23] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547, 2003.

[24] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[25] A. J. Spiers, B. Calli, and A. M. Dollar. Variable-friction finger surfaces to enable within-hand manipulation via gripping and sliding. *IEEE Robotics and Automation Letters*, 3(4):4116–4123, Oct 2018.

[26] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *journal of machine learning research*, 11(Nov):3137–3181, 2010.

[27] Herke Van Hoof, Tucker Hermans, Gerhard Neumann, and Jan Peters. Learning robot in-hand manipulation with tactile features. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 121–127. IEEE, 2015.

[28] Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[29] Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. *CoRR*, abs/1810.06045, 2018.

[30] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. 2008.