Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Hierarchical Inverse Reinforcement Learning from Motion Capture Data

Rohit Prakash

Course of Study:	INFOTECH
Examiner:	Prof. Dr. rer. nat. Marc Toussaint
Supervisor:	Dr. Jim Mainprice, Philipp Kratzer, M.Sc.
Commenced:	April 29, 2019
Completed:	October 29, 2019

Abstract

A human motion generally consists of multiple low-level tasks which are performed in a defined order or in parallel to achieve a high level task. For example, making pizza dough consists of several low-level tasks such as measuring water, adding yeast, measuring flour, etc. And these activities must be performed in a definite order to make the dough. For a system to imitate these sequence of activities of a long horizon task with one global reward function is a lot of work. This process can be made easier if there is hierarchical state representation of the task and learning of local rewards for the hierarchies. In this thesis, we have learned to imitate a general day to day human activity of 'setting table for one person'. This work adopts a framework called Hierarchical Inverse Reinforcement Learning (HIRL), which is a model to learn sub-task structure from demonstrations. With this framework, the activity is decomposed into multiple lower level tasks which are performed in a sequence using learned policies. In this work, Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL) is used to learn local rewards for the sub-tasks. Together with hierarchical state space representation and local reward functions, the model encodes the high level task objective based on human demonstrations of full body motion performing the high level task. The model achieves a success rate of 84% on average for middle levels and 83% for the top level in the cross validation tests. For visualization, the model is simulated in a 2D representation that takes current environment state as input and runs till the completion of the task.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

DQN Deep Q Network. 24

HAM Hierarchical Abstract Machines. 20

HIRL Hierarchical Inverse Reinforcement Learning. 3

HRL Hierarchical Reinforcement Learning. 15

IOC Inverse Optimal Control. 16

IRL Inverse Reinforcement Learning. 16

MaxEnt-IRL Maximum Entropy Inverse Reinforcement Learning. 3

MDP Markov Decision Process. 17

SMDP Semi-Markov Decision Process. 20

1 Introduction

Recently, there has been a number of advancements in the field of Reinforcement Learning. The coming of age Alpha Go AI by DeepMind proves how far we have come in development of an artificial intelligence. Alpha Go is arguably the best Go player in the world. It went on to defeat the Go world champion in global arenas[Dee19]. These achievements are the direct result of years of research and application in various scenarios using reinforcement learning. From learning and mastering of Atari games to performing complex real world tasks, reinforcement learning has showed its usefulness and capabilities on various situations.

However, reinforcement learning has its limitations when it comes to complex environments. In a complex environment, state and action space is huge and widespread. Therefore, scaling of task from initial state to terminal state will take forever. The length of the path would be a very long combination of sequences of states and actions. As a result, the reward for an activity is delayed since it takes a long time for completion of an activity. Hierarchical Reinforcement Learning (HRL) tries to reform this problem in a much smaller state space with the use of hierarchies. A complex task can be broken down to hierarchies and provide state and temporal abstractions which could simplify the problem. The use of HRL to solve classic Atari games is an example of such application. The figure 1.1 shows two classic game domains solved using HRL. The first one is the Montezuma's Revenge, where the agent learns the complex task of acquiring the key shown within the blue boundary and leaving out the door shown within the red boundary. For the agent to leave to next stage, he must first go down the ladder, jump to the right, down the ladder again, over the obstacle, up the ladder on the left and acquire the key. Then he must retrace his steps back to the top. And then finally, walk to the door.

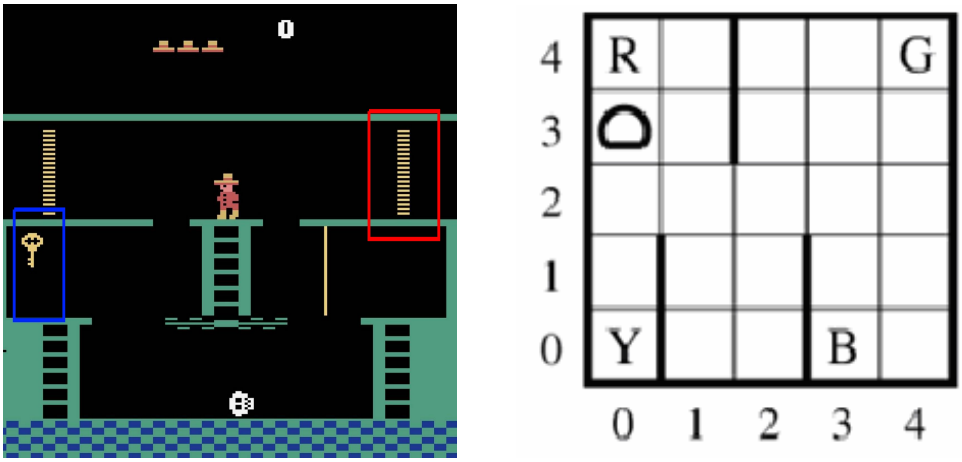

Figure 1.1: Montezuma's Revenge and Taxi domain

The other example is the Taxi domain, where the agent picks up the passenger from one of the four locations(red, green, blue or yellow), goes to the destination location and drops the passenger.

The hierarchies defined has to work in tandem between the levels for the agent to reach its goal. For instance, given the task of cutting a cucumber, human may perform lower level actions such as picking the cucumber, cleaning it and holding knife in stipulated fashion and then cutting it. This is quite analogous to HRL framework and what this framework is trying to achieve by simplifying a long horizon task by breaking down into hierarchies.

In spite of this success, still the learning of the tasks takes a huge number of trials before an agent learns the optimal path or behaviour. Even though the framework could be a benchmark in learning complex scenarios, this is quite inefficient when compared to a human for the same activity. One drawback is that the reward of the states is not already known and agent learns this with various trials of exploration and exploitation like in case of Montezuma's revenge. If the reward could be already known beforehand, our problem reduces to finding an optimal path in the given state space. Also, the reward which the agent receives by performing desired behaviour is defined by experts, such that agent learns from it. This is an additional overhead when defining an environment for a reinforcement learning task. In case of the Atari games or tasks with lower complexities, defining rewards in the environment is easy. But for a real world long horizon task, it is a lot of work. So how will an expert define rewards for a task of self driving in cars?

Inverse Reinforcement Learning (IRL) is another sub field of reinforcement learning, where reward is learned from expert demonstrations. It is also known as Inverse Optimal Control (IOC). The Human expert is a benchmark we are trying to keep up with, and a demonstration of an activity by him can be considered to be viable path to a target state. So learning the reward from a demonstration from a human would make sense as human in most cases is expected to choose states and actions which is most or near efficient.

The objective of this thesis is to learn reward from human demonstration for various activities in the environment defined and learn to plan the same activities in an efficient manner with hierarchical states using the learned rewards and features of the environment. This work attempts to combine HRL and IRL techniques to imitate a human demonstration.

Chapter 2 gives insight into the background literature for the techniques used in this thesis. In chapter 3, existing works present in the domains of human motion prediction, hierarchical learning and hierarchical inverse learning are discussed. The following chapter 4 discusses the core implementation of the model proposed. All the methods and steps involved in the development of the model is explained in detail. In chapter 5, the results of the thesis are discussed with respect to cross validation tests and comparison with ground truth. In the chapter 6, some of the pointers for future works and improvements are mentioned and finally chapter 7 presents the conclusion of the thesis.

2 Background

2.1 Markov Decision Process and Reinforcement Learning

Markov Decision Process (MDP) [Bel57][SB98] is defined as a mathematical framework that models decision making in the environment of an agent. The MDP can be defined using four-tuple M = (S, A, $P_a(s|s')$,$R_a(s,s')$).

- S is the finite set of states in the environment.
- A is the finite set of actions in the environment.
- $P_a(s|s')$ = $Pr(s_{t+1} = s'|s_t = s, a_t = a)$ is the transitional probability that an action a in state s at time t leads to a state s' at time t+1,
- $R_a(s,s')$ is the immediate reward (or expected immediate reward) received after transition from state s to state s', due to action a.

Policy π, can be defined as a mapping of states to actions which informs what action to perform in a given state s.

In an MDP, there is a decision maker called the agent who interacts with the environment. Given a state s, from the set of states S of the environment, the agent selects an action a, from set of actions A. The environment then transitions into a new state and agent receives a reward r, as a result of the action. This process happens sequentially and agent jumps from one state to another creating a trajectory. If the states and actions involved are finite, the MDP is defined as a finite MDP. An important property of MDP is that agent only has to consider the current state it is in for selecting an action. This property is known as Markov property.

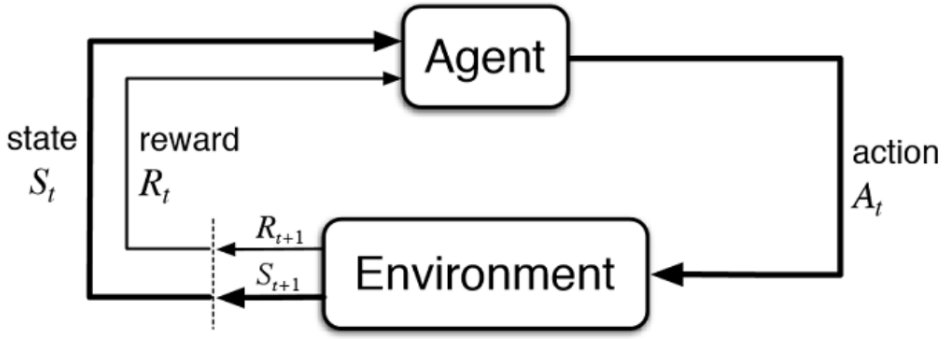

Figure 2.1: Agent-environment interface

Based on the rewards in the environment, the objective of the agent is always to maximize cumulative rewards over time. We can define return G at time t as

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T \tag{2.1}$$

where T is the final step. If T is finite, then this definition works well. However if T is ∞, a discounting factor γ is introduced since return is no longer guaranteed to be finite. And we can rewrite the above equation as

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \tag{2.2}$$

where discount rate γ is between 0 and 1.

An important thing for an agent in the environment is to understand how good it is to be in a given state or to know how good it is to perform a given action in the given state. This notion is made clear with the value functions. There are two types of value functions used in reinforcement learning: the state value function, denoted as V(s), and the state action value function, denoted as Q(s,a).

The state value function is the expected return when starting from state s acting according to our policy π:

$$V^{\pi}(s) = E_{\pi}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...|s_t = s] = E_{\pi}[\sum_{i=0}^{\infty} \gamma^i r_{t+i+1}|s_t = s] \tag{2.3}$$

This equation can be rewritten as

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}[r_{t+1} + \gamma \sum_{i=0}^{\infty} \gamma^i r_{t+i+2}|s_t = s] \\ &= E_{\pi}[r_{t+1}|s_t = s] + \gamma E_{\pi}[\sum_{i=0}^{\infty} \gamma^i r_{t+i+2}|s_t = s^{'}] \\ &= \sum_{s'} P(s'|s,\pi(s))[R(s'|s,\pi(s)) + \gamma V^{\pi}(s')] \end{aligned} \tag{2.4}$$

If the policy is a stochastic policy,

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)[R(s'|s,a) + \gamma V^{\pi}(s')] \tag{2.5}$$

Similarly, the state-action value function or Q function gives the expected return for performing action a in state s when following our policy π. It can be derived to the following equation,

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)[R(s'|s,a) + \gamma Q^{\pi}(s',\pi(s'))] \tag{2.6}$$

If the policy is a stochastic policy,

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)[R(s'|s,a) + \gamma \sum_{a'} \pi(a'|s')Q^{\pi}(s',a')] \tag{2.7}$$

A policy π is optimal if the expected return for π is greater than or equal to the expected return of π' for all states. In other words,

$$\pi \geq \pi' \text{ if and only if } V_{\pi}(s) \geq V_{\pi'}(s) \text{ for all } s \in S.$$

The optimal state value function with their corresponding optimal policy can be written as below,

$$V^*(s) = \max_a [\sum_{s'} P(s'|s,a)(R(s'|s,a) + \gamma V^*(s'))] \tag{2.8}$$

$$\pi^*(s) = \operatorname*{argmax}_a [\sum_{s'} P(s'|s,a)(R(s'|s,a) + \gamma V^*(s'))] \tag{2.9}$$

The optimal state action value function and corresponding optimal policy can be written as below,

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)[R(s'|s,a) + \gamma \max_a Q^*(s',a')] \tag{2.10}$$

$$\pi^*(s) = P(a|s) = \operatorname*{argmax}_a (Q^*(s,a)) \tag{2.11}$$

Any basic reinforcement learning problem can be modeled as MDP. In classic reinforcement learning, the learning takes place based on a feedback system. Unlike supervised learning, where we have the solution in the dataset while training, reinforcement learning algorithm learns via interaction with the environment. There are two broad possibilities of the policy learning algorithms: Model-free algorithm and model-based algorithm.

In model-free algorithm, the agent is not aware of transitional probabilities and learns optimal behaviour with trial-error method with application of different action. Q learning is an example of model-free method. Q learning is an off policy algorithm that evaluates which action to be taken in a state based on action-value function. Learning takes place with exploration of the environment and action-value for a state is updated as algorithm goes along.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \max_a Q(S_{t+1},a) - Q(s_t,a_t) \tag{2.12}$$

At each step, Q table is updated with new approximated value for best action in a state. Function approximation can also be used for the same purpose. This is known as Deep Q network. This is widely used nowadays for a number of application. It can also be extended with hierarchies for HRL tasks.

Value Iteration is a model based algorithm which uses transitional probabilities. The algorithm estimates the value function for all the states. Thus, policy for each state would be the action whichever gives the highest value for a state. Value iteration can be considered as an update rule which converges to optimal V^*. With reference to Bellman optimality equation, it can written as below:

$$V_{k+1} = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_k(s')] \tag{2.13}$$

2.2 Hierarchical Reinforcement Learning

The notion of Hierarchical Reinforcement Learning was developed to address some of the problems of the reinforcement learning in a complex environment. Solving a reinforcement learning problem in a high dimensional state space and action can be problematic due to very long sequences of actions to reach a defined goal. The idea of HRL is basically to subdivide a problem into hierarchy of tasks and solve them on different levels of temporal abstraction.There are broadly three approaches for HRL, namely Options Framework, Hierarchical Abstract Machines and MAXQ Value Function Decomposition.

Options Framework [SPS99] is a well known HRL framework. In this framework, high level actions or macro-actions are options. An option consists of an initiation set, I_o, which describes the low level state in which option can be executed; an option policy π_o, which is the policy that gets executed and a termination condition, β_o, which as the name suggests, provides the condition for termination of the particular option policy. Thus here, the classical definition of MDP needs to be altered by replacing the agents' low level action with options. This is known as Semi-Markov Decision Process (SMDP) which can be described by the tuple:

$$M = (S, O, P, R, \gamma)$$

where S is the original state space, O is the set of options defined, P is the transitional probability for jumping to a state after execution of an option and R is the reward received after executing an option.

MAXQ decomposition [Die00][Yan19] is a framework where hierarchy is created by decomposing the value function of an MDP into combinations of value functions of smaller MDPs. It can be represented as sum of two components $Q(p,s,a) = V(a,s) + C(p,s,a)$ where V(a,s) is the total expected reward gained when executing an action a in state s and C(p,s,a) is the reward expected from the performance of the parent task after taking action a.

Hierarchical Abstract Machines (HAM) [PR98][Yan19] is framework that constitutes of non-deterministic finite state machines whose transitions lead to lower level machines and back again. There are four machine states: action, call, choice and stop. In this approach, the states are complex to design and implement. Hence it is the least popular among the frameworks.

2.3 Inverse Reinforcement Learning

The initial basic algorithm for IRL was introduced in Ng and Russell [NR00]. Many of the future works such as Ziebart et al. [ZMBD08] and Ramachandran and Amir [RA07] on IRL is based on this approach. The main issue with IRL is that a policy can be optimal for multiple reward functions,i.e, even if we have a actions from an expert there might exist multiple reward functions for the particular action. The IRL problem is modeled as an optimization problem. In the paper, it is stated that a policy π given by $\pi(s) \equiv a_1$ is optimal if and only if all the actions and reward function satisfies the condition:

$$(P_{a1} - P_a)(1 - \gamma P_{a1})^{-1} R \geq 0 \tag{2.14}$$

Algorithm 2.1 Abstract algorithm for IRL

procedure
Input : feature matrix, expert trajectories D = $\tau_i{}_{i=1}^{N}$
Initialize reward function, r
repeat
compute soft value function, V
compute policy, π_0
evaluate state visitation frequency μ for the current policy π_0
evaluate objective function L and its derivative $\nabla_\theta L$ by comparing μ and state action distribution implied by D.
update reward parameter θ
until gradient $\nabla_\theta L$ converges
return reward function,r for the optimum policy,π_0
end procedure

The problem is reduced to simple optimization using linear programming algorithms. However, there is a problem which is quite apparent. R = 0 is always a solution and there could be many options for R that could meet the condition. An additional constraint is introduced to possibly reduce the number of solutions to the above equation. The solution would be chosen that maximizes the following condition.

$$\sum_{s \in S} (Q^{\pi}(s, a_1) - \max_{a \in A \setminus a_1} Q^{\pi}(s, a)) \tag{2.15}$$

The intuition is that any deviation from optimal policy should eventually result in reduction of the total reward. This work also introduces the concept of having reward function as linear combination of feature vectors. This was mainly done for approximation in large state spaces. An abstract algorithm for IRL is described in algorithm 2.1[OPN18].

2.4 Maximum Entropy Inverse Reinforcement Learning

MaxEnt-IRL [ZMBD08] is a methodology to learn the reward function from observed demonstrations of an optimally behaving agent. In addition to Abbeel & Ng's [AN04] approach, along with matching feature counts of the observed path and optimal path, principle of maximum entropy [Jay57] is used. With having only the matching of feature counts, the problem is ambiguous since there can be multiple policies for the observed feature counts. In this work, they adopt a probabilistic approach to resolve this ambiguity. Thus, in process handling uncertainty and noise as well. This yields soft maximum version of Markov decision process(MDP). The softened version of MDP equation is as follows[ZRG+09]

$$Q^{\approx}(s, a) = R(s, a) + V^{\approx}(T(s, a)) \tag{2.16}$$

$$V^{\approx}(s) = \text{softmax}_a Q^{\approx}(s, a) \tag{2.17}$$

The reward function is defined as a linear sum of weighted combinations of feature responses $f(s) = [f_1(s)...f_k(s)]^T$. θ is the set of reward weights for the corresponding features. The reward function is applicable to any configuration of environment.

$$r(s;\theta) = \theta^T f(s)$$

The policy for the MDP is probabilistic with distribution: $\pi(a|s) = e^{Q^{\approx}(s,a)-V^{\approx}(s)}$. From all the observed paths from train set, the empirical feature count is calculated as average over total feature counts by number of demonstrations, $\tilde{f} = \frac{1}{M}\sum_{m}^{M} f(s)$ where M is the total number of demonstrations. The gradient is calculated as difference of empirical feature counts $\tilde{f}$ and expected feature counts $\hat{f_\theta}$, which is average features accumulated by trajectories. The expected feature counts are calculated in two step process - backward pass and forward pass. And the parameters θ are learned with each iteration until the feature counts are matched.

3 Related Work

3.1 Human Motion Prediction

There has been quite a lot of work on prediction of motion of an agent in an environment. Ziebart et al. [ZRG+09] is a follow-up of Maxent-IRL. This work presents an uncertainty based planning approach to assign cost function for different features of each state and action in the space. The method employs softened version of MDP and gradient based optimization to come up with cost function for the states of the environment. Activity forecasting [KZBH12] is an extension of Ziebart et al. [ZRG+09]. The objective of the paper is to forecast future action of people from a noisy input. The destination forecasting and understanding of user preferences with respect to physical environment is one of the major focus of the paper. The scene of the environment is semantically extracted to produce features such as cars, pavements, grass, sidewalk, curb, etc. The observed trajectory of the person in the scene is recorded, which along with the physical scene features gives the empirical feature count to be matched with, for IRL part of the problem. Using MaxEnt IRL, the reward weights for each semantic features are learned. And naturally those physical attributes in environment which human tend to prefer is calculated as higher reward weight than others. For instance, sidewalk has higher reward than a car in the scene. This understanding of reward for the semantics of environment helps to predict future probable motion of a person. Even if there are changes, like rearrangement of objects in physical environment, the learned reward weights are still valid since they are separately learned for each feature responses. Hirakawa et al. [HYY+17] also adopts activity forecasting model for predicting non goal directed trajectories of sea bird. There has also been work on the motion prediction of human in a group. Rudenko et al. [RPLA18] presents an approach by considering constraints from the environment and social forces. The work employs the group social force model [MPG+10] to come up with future paths taking in repulsive forces from other people in the surroundings and intra-group forces. There has also been work done to combine the social forces with IRL to get a path in a dynamic environment. Vasquez et al. [VOA14] adopts IRL techniques with features that change over time. Thus re-planning is done in an optimal fashion to factor in on the changing features. This approach uses density features, speed-orientation, velocity and social forces to plan a path for a robot in a dynamic environment.

3.2 Hierarchical Learning

There has been quite a number of works in HRL for learning favourable behaviour in a complex environment. Many of these works are inspired from the basic HRL architectures explained in section 2.2. Kulkarni et al. [KNST16] adopts a hierarchical DQN framework to solve the Atari game called Montezuma's Revenge. Here, there are only two levels of hierarchies controlled by meta-controller for top level and controller for bottom level. The goal of the agent is to acquire a key which is situated at a location in the environment. The agent is required to learn multiple sequences

of actions to acquire the key first and then go to an exit door. The meta-controller takes in the state of the environment and its Deep Q Network (DQN) learns to choose the best meta-action, which here is a destination in the environment. The bottom level controller is responsible for primitive actions of motion and grasp and has its own separate DQN which chooses the best primitive action for a combined input of state and subgoal. In this case, as one can perceive, both levels have to work in tandem to reach the terminal state. There is temporal abstraction for the subgoals and since the reward from the environment is delayed, the sub actions performed by lower level controller is provided with intrinsic motivation. In spite of the simplification of state space for the complex environment, the learning of ideal sequence of actions from exploration and exploitation is a tedious task. Konidaris et al. [KKL18] argues that it is difficult for an agent to come up with high level plan from a low level domain of the world. Abstract representations of an environment is created for planning in high dimensional space.The agent learns these representations on its own using its sensory inputs and models a high level plan for a simple task in a particular environment.

3.3 Hierarchical Inverse Learning

Combining HRL with IRL is coming up as viable options in dealing with long horizon problems for practical applications. In [KGL+16], a long horizon task is decomposed into linear sub-goals and sequence of local rewards are learned with MaxEnt-IRL algorithm. The local rewards guide the agent to the given sub-goals which happen to be more efficient than the sparse true reward. The Q learning technique is used to learn the policies for the augmented state space for the problem. The paper proves that HIRL converges much better than other alternatives for the same experiment. This work is quite close to the work presented in this thesis as it deals with a long horizon task for setting up of a table. The task presented in this thesis is composed of multiple MDPs that transition among themselves and the switching is also modelled by another MDP at top level. Another alternative that can be modeled for the scenario of this thesis is the guided cost learning[FLA16]. Though the work explicitly does not mention the use of hierarchies, however it deals with high dimensional state space problem. The work presents a model that can learns a non-linear cost function using neural networks for a state space with unknown dynamics. This approach uses sample based approximation of MaxEnt-IRL and can be applied to a variety of real world challenges. As an experiment in the paper, the robot learns to carefully arrange plates in a dish rack from a human demonstration.

4 Methods

In this chapter, the focus is on the methods involved in setup and development of the system. The first section discusses the environment and process of data collection for the tasks involved. The second section gives an overview of the implementation for obtaining trajectories. The third section gives a brief picture of the system developed with hierarchies for planning and the final section provides more information into the different features used for the development.

4.1 Environment and Data Collection

The environment of the experiment includes 4m*4m area consisting of rigid objects such as table, a couple of identical chairs, a small shelf and a big shelf. An OptiTrack[1] motion capture system with 12 cameras is used to capture the demonstrations and a motion capture suit to be worn by the demonstrator. The cameras and the skeleton of the demonstrator in the suit are calibrated before experiment.

There were 16 different type of tasks performed in the dataset as defined in table 4.1. For these tasks, there were 4 coloured plates, 4 coloured cups, 1 bowl and 1 jug. The environment also consists of rigid objects as table, big shelf, small shelf and couple of chairs as depicted in figure 4.1. For instance, the task of setting table for three person, would mean the demonstrator should make sure he or she picks one object at a time from an initial spot (e.g. big Shelf) and place it to destination (e.g. table). In the end of this particular task, there must be 3 plates, 3 cups and either of jug or bowl must be on the table.

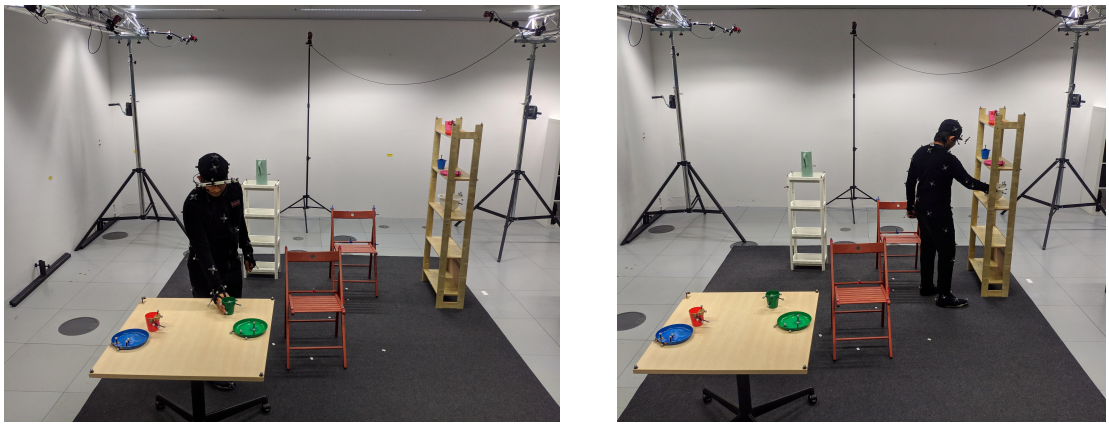

Figure 4.1: Demonstrator performing action

[1] https://optitrack.com/

S.No	Task description
1	setting table for one person
2	setting table for two persons
3	setting table for three persons
4	setting table for four persons
5	clear table
6	place jug and bowl on small shelf
7	place all cups on small shelf
8	place blue and pink objects on big shelf
9	place blue and red objects on big shelf
10	place blue and green objects on big shelf
11	place pink and red objects on big shelf
12	place pink and green objects on big shelf
13	place red and green objects on big shelf
14	place all cups on big shelf
15	place bowl and jug on big shelf
16	place all plates on big shelf

Table 4.1: Tasks performed in the dataset

For the data collection, the various tasks were performed in sequence without any particular order. Each of the objects in the scene including rigid objects have markers to track their position in space. Also the motion capture suit is accompanied with 50 markers to map the motion of the demonstrator in the space. For this work, the 2D positions and quaternion angles were mainly used.

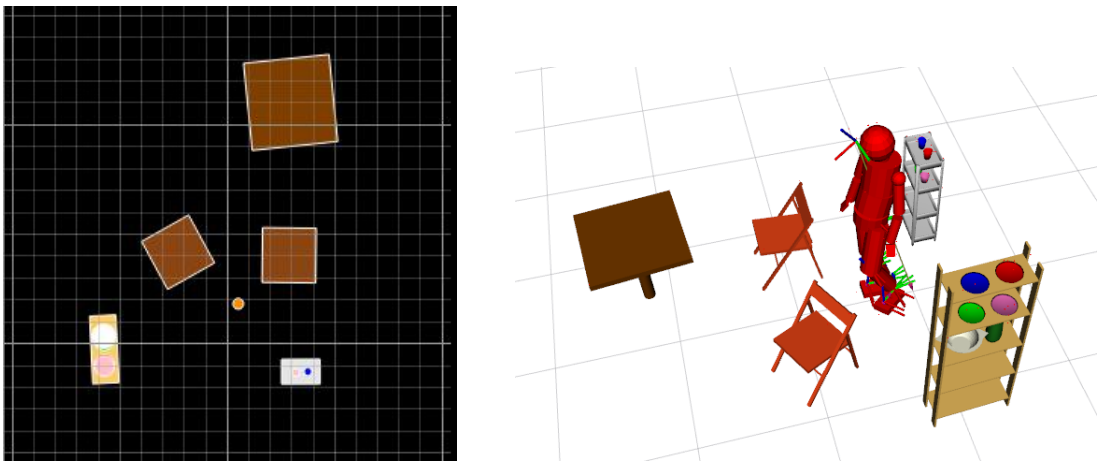

Figure 4.2: 2D representation of environment

Using this information, precise 2D interface of the environment was developed in PyQT where the activity performed can be viewed in 2D in parallel with 3D visualization in RViz [2] as depicted in figure 4.2.

4.2 Trajectory Formulation in 2D Map of Environment

A part of the thesis is to be able to navigate in the environment defined previously, by avoiding obstacles and making it to goal with efficiency. The Kitani et al. [KZBH12], uses Maxent IRL to learn the reward weights for the physical attributes in the environment. The linear sum of reward weights with feature responses of the physical attributes, provides the reward function for the states in the environment. With optimal control, a viable trajectory can be forecasted from an initial state to destination.

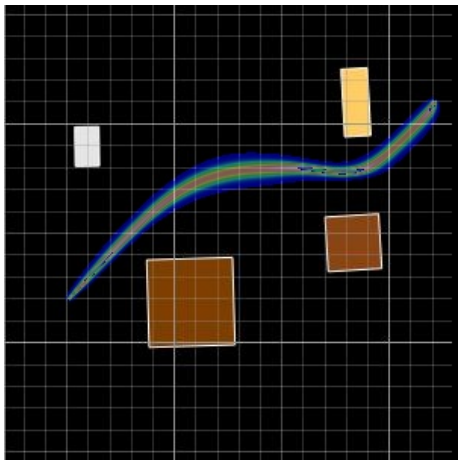

Figure 4.3: Trajectory prediction

The environment has obstacles in the form of chairs, tables and shelves. The way around these rigid objects would obviously be the user preferred path. In the Kitani et al. [KZBH12], top down visual images of a public location were used and features of physical attributes were generated after semantic labeling. A stride away from this approach, instead of the top down actual images, the images were extracted from the 2D representation of the environment, defined in section 4.1. This made semantic labeling easier as well and the trajectory was available from the motion capture setup. The figure 4.3 shows the prediction of trajectory between two points in the environment. The start being the lower left point to a point in the upper right, a path is predicted around the obstacles using corresponding reward map of the environment.

[2] https://github.com/ros-visualization/rviz

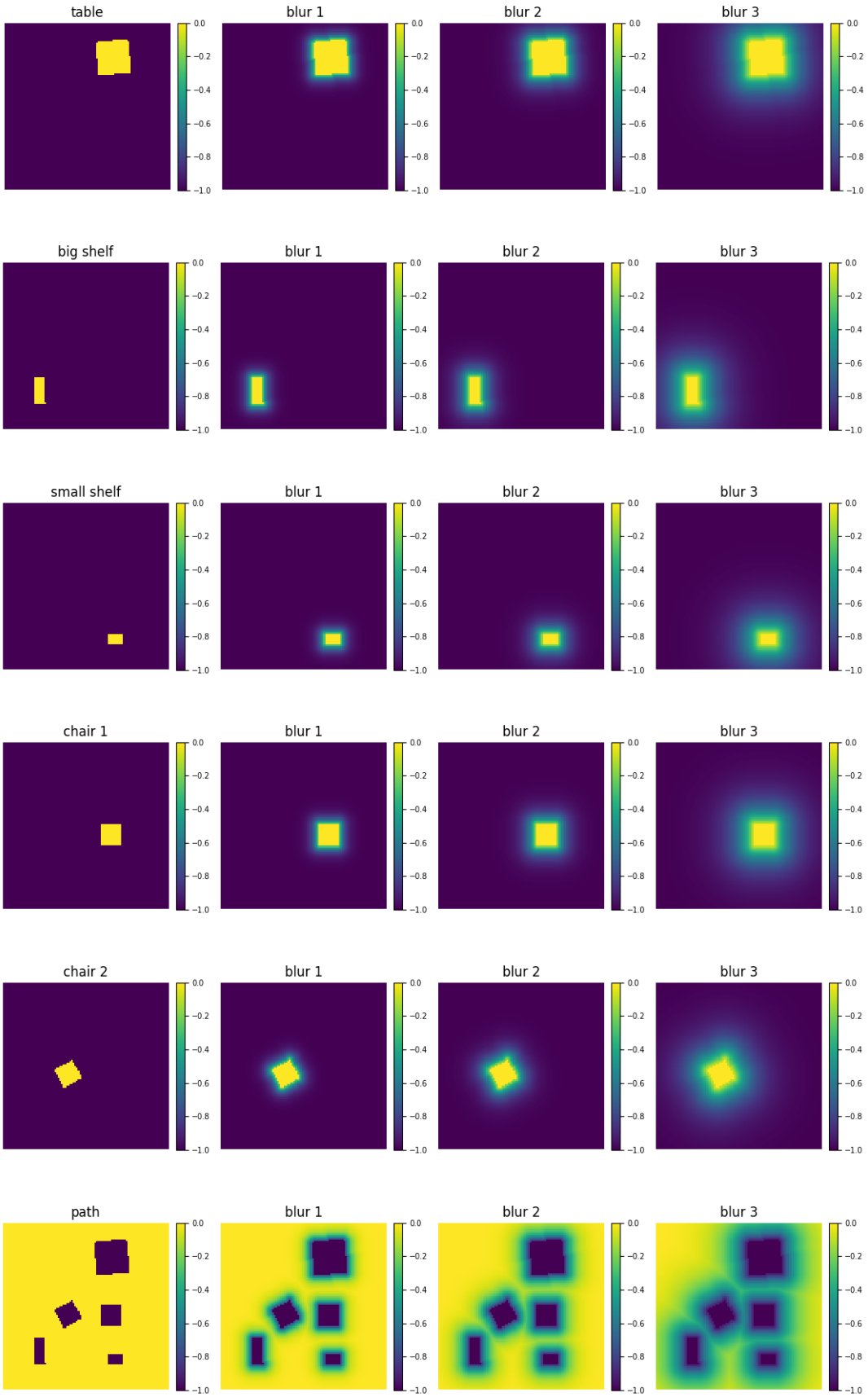

Figure 4.4: Feature maps of the environment

4.2.1 Feature Map

As defined in the Ziebart et al. [ZRG+09] and in Kitani et al. [KZBH12], four feature maps were generated for each physical attribute in the scene. The first feature is the probability map which indicates the presence of an object in the cell. The rest three are exponentiated distance function with variance, $\sigma^2 = 3, 5, 10$. Each feature map is of the size 100 pixel*100 pixel. As total, there are 25 feature maps for the 6 different physical attributes and an additional constant function. Each of the feature map is normalized in the range between [-1, 0]. The figure 4.4 depicts the feature maps used to generate the reward function. The 2D interface of the environment also generates feature tensor for a scene of the environment.

4.2.2 Learning Reward for the 2D Map

The trajectory logged from demonstrations, the image of the scenes and its corresponding feature maps, becomes the input for the model. The model uses the trajectory over the corresponding feature maps to calculate the empirical mean feature count of the demonstrations. The backward pass algorithm followed by forward pass algorithm learns state visitation frequency to calculate expected feature counts [ZMBD08][KZBH12]. This is repeated until the expected feature counts matches up to the empirical feature counts. The calculated reward weights is the information that would correspond to user preference in the environment.

The figure 4.5 depicts the final reward function obtained for a scene of the environment. The yellow surface has higher reward and the dark blue has lower reward. The reward weights for each of the feature maps were calculated using MaxEnt-IRL. The reward weights were highest for the feature maps of path and its three blurs. The rest of the feature maps had negligible reward weights in comparison to the path. Thus, the final reward function looks more like a combination of feature maps of the path.

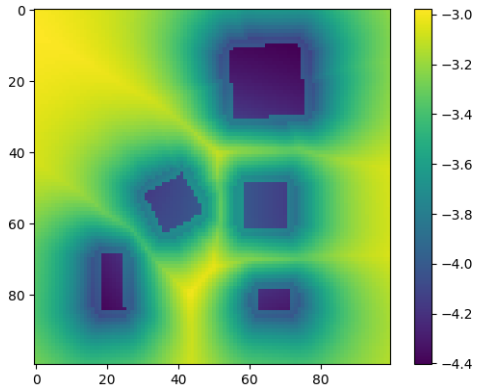

Figure 4.5: Reward function of a scene

4.3 Hierarchical Structure and State Representations

In this section, the hierarchies involved for the high level planning is explained for each level. The below section explains basic setup and prerequisites. The sections following correspond to each of the hierarchies in the system developed. Each of these subsection would describe deeper into the state and action spaces of the corresponding levels. For this thesis, the task of 'setting table for one person' is considered. Therefore, all the levels defined here are not applicable for other tasks in the dataset. The middle levels and the navigation level can be re-purposed for other tasks. However, top level cannot be used for other tasks.

4.3.1 Basic Setup and Prerequisites

The tasks for setting table for one person is quite complex compared to simple Atari games or taxi domain. There are three items- table, big shelf and small shelf as rigid objects which can accommodate other movable objects such as 4 plates, 4 cups and jug or bowl on them in different combinations at any point in time. There is also one additional possibility. The human demonstrator can also be handling one of the objects. Therefore, in any given instance, the movable objects can be in any of the four locations - table, big shelf, small shelf or human. In figure 4.6, couple of scenario from real demonstration is visualized in RViz. As it can be seen, table is set with red plate & cup with bowl in the first image and green plate & cup with jug in the second image.

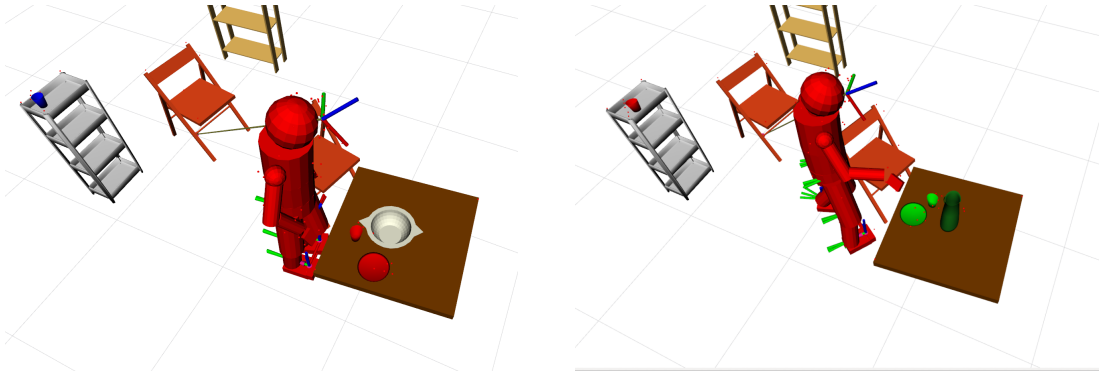

Figure 4.6: Different possibility of arrangement of table for one person

Thus to define this complex environment into a simpler format, the zones are defined. The environment is divided in to 4 zones namely:

- near the table
- near the big shelf
- near the small shelf
- away

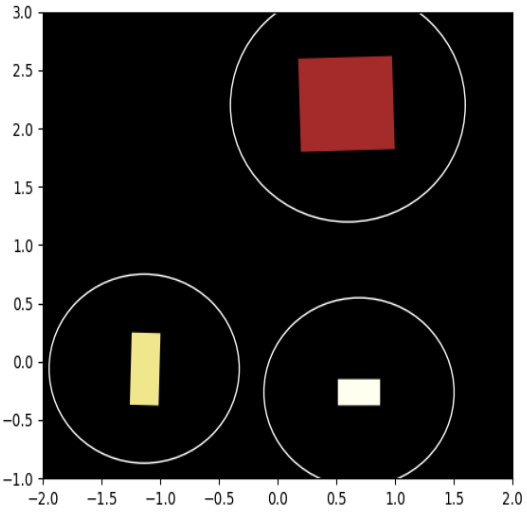

Figure 4.7: Zones of the environment

The movement of the demonstrator is tracked with respect to these zones. The zone values are enumerated starting from 0 to 3. The zones cover a circular patch of region around the rigid objects. The extent of circular region was decided based on the demonstrations and also making sure that zones do not overlap as depicted in figure 4.7. It is only possible to grasp or release within the first three zones. The fourth zone covers a larger region than others. The demonstrator continues to hold the object in this region. The location of movable objects are defined similarly, namely:

- on table
- on big shelf
- on small shelf
- with demonstrator

The location values are enumerated starting from 0 to 3. Any object placed on these rigid objects will have one of the location values.

Finally, putting the individual location of movable objects and location of demonstrator together, we can define the state of environment as vector of size 11(10 objects and demonstrator).

state - [cup1, cup2, cup3, cup4, plate1, plate2, plate3, plate4, jug, bowl, demonstrator]

As seen above, each value in the state correspond to the position of object in the scene based on locations defined and zone for the demonstrator. At any point in time, maximum of only one object is held by the human. For instance, state - [0,1,1,2,2,3,1,0,0,1,3], state[0] to state[3] gives location of all the cups; state[4] to state[7] gives location of all the plates ; state[8] to state[9] gives location of jug and bowl respectively; and finally state[11] gives location of human in terms of zone. As one can perceive, human is in zone 3 with a plate in his hand.

Action for this setup is very basic like move, grasp and release. From the actual motion capture dataset, the state of the environment is extracted in the format defined with respect to time. This sequence of state forms the basis for upcoming sections for defining their corresponding states and actions.

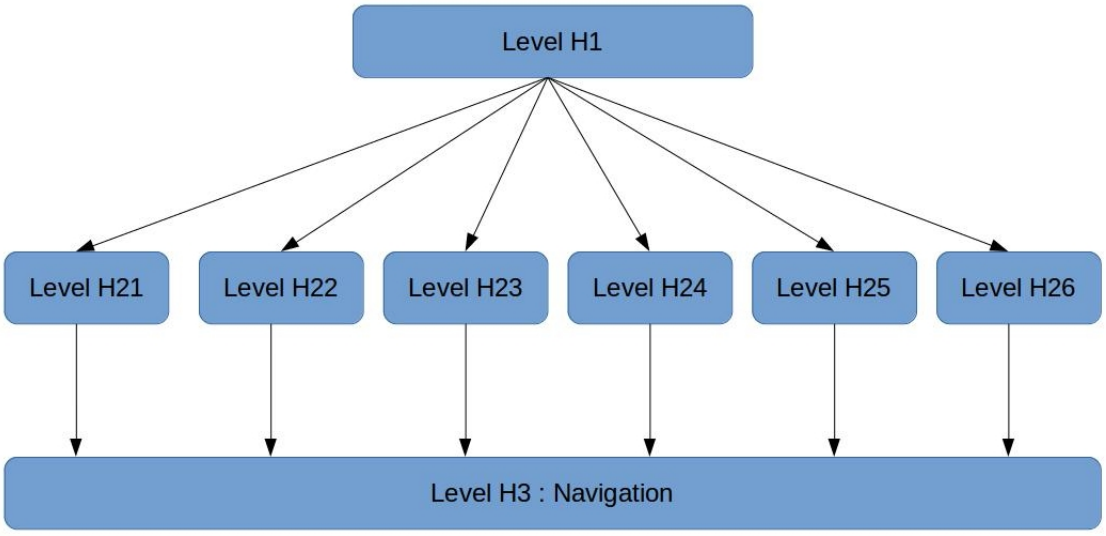

Figure 4.8: Hierarchical structure of the system

4.3.2 Level H1

This is the top level of the system. Here the state is defined with respect to number of cups, plates and jug or bowl on the table and shelves (both the shelves are considered as one entity). The state structure for this level is defined as below:

state = [cupTable, cupShelves, plateTable, plateShelves, jugbowlTable, jugbowlShelves]

Macro-actions for this level are presented in table 4.2

For instance, if the state = [1,3,1,3,0,2]. It means there is 1 cup on table, 1 plate on table. Rest of the cups, plates, jug and bowl are on either of the shelves.

There is only one terminal state for this level - [1,3,1,3,1,1] which corresponds to 1 cup, 1 plate and either jug or bowl on table. There is a point to note that jug and bowl are considered in same category like cups or plates since they are interchangeable in setting up of table. Total number of possible states are 75. These states are extracted from the state and actions defined in section 4.3.1. An example of a sequence for this level is shown in table 4.3.

Macro-actions		
Sub-level	Name	Description
H21	MoveCupTabletoShelves	move the cup on the table to either of the shelves
H22	MovePlateTabletoShelves	move the plate on the table to either of the shelves
H23	MoveJBTabletoShelves	move the jug or bowl on the table to either of the shelves
H24	MoveCupShelvestoTable	move the cup to the table from either of the shelves
H25	MovePlateShelvestoTable	move the plate to the table from either of the shelves
H26	MoveJBShelvestoTable	move the jug or bowl to the table from either of the shelves

Table 4.2: Macro-actions

Level H1		
S.No.	States	Meaning
1.	[4, 0, 1, 3, 0, 2]	initial state
2.	[4, 0, 1, 3, 1, 1]	jug moved to table from shelf
3.	[3, 1, 1, 3, 1, 1]	cup moved from table to shelf
4.	[2, 2, 1, 3, 1, 1]	cup moved from table to shelf
5.	[1, 3, 1, 3, 1, 1]	cup moved from table to shelf

Table 4.3: Example- sequence of states for Level H1

These possible states encapsulates all possibilities of arrangement of 10 movable objects in the environment with focus to table alone. The shelves are considered as single entity which separates only in the lower levels.

There were different types of tasks performed in sequence, for instance, setting table for three persons followed by setting table for one person. In such a scenario, the planning has to be done in a manner that objects have to be removed from the table to set for one person. Precisely in this case, two cups and two plates have to be removed. Such a structure of states also enables to learn to remove objects from the table to come to the defined terminal state rather than just add objects on to the table.

4.3.3 Level H2

This is the middle level of the system. In this level, the macro-actions from top level dictates the state structure and actions are more primitive. Since there are 6 macro-actions defined for top level, there are 6 branches in this level for each of them as depicted in the table 4.2

The state structure and actions for each of these branches are mostly identical. So therefore, for purpose of explanation, only the branch H21 would be the point of interest. However, there are some exceptions present amongst branches, which will be explicitly addressed.

Below is the state structure for the branches in this level.

state = [cupOntable, cupOnBigShelf, cupOnSmallShelf, skeletonLocation, objectCapturedIndication]

State defines the number of cups on each of the rigid objects, location of the person in the scene and an indication if cup is captured.

Actions of this branch are:

- MovetoBigShelf - move to a point near big shelf
- MovetoSmallShelf - move to a point near small shelf
- MovetoTable - move to a point near table
- GraspCupTable - grasp cup from the table
- ReleaseCupShelf - release cup on the shelf

It can be perceived that the states have direct link with the macro-action from top level. Similarly, each of the branches in this level are specific to type of movable objects like cups, plates or jug bowl. It is to be noted that the location of person with respect to zones is accounted here for the first time in this level.

There are a total of 100 states possible for this branch which are identical with branches H22, H24 and H25. Intuitively, it can be realised that, the branches H23 and H26 will have only 36 possible states (since only two objects - jug and bowl, comparing to four in other branches).

Regarding terminal states, there are multiple possibilities for each of the branches. For instance, for branch H21, the cups can be taken from the table to either of the shelves and there could multiple combination of other cups present on any of the rigid objects. This creates a number of possible terminal states.

This type of framework can be found analogous to Option Framework. The top level performs SMDP with options instead of actions and the middle levels are analogous to those options. Just like in case of options, the middle levels perform its policies until a termination condition. It has a sole purpose which it tries to fulfil and returns control to the top level. An example of a sequence for this branch H21 is shown in table 4.4.

Level H21		
S.No.	States	Meaning
1.	[3, 0, 1, 2, 0]	initial state of cups
2.	[3, 0, 1, 0, 0]	person moves near table from small shelf
3.	[2, 0, 1, 0, 1]	grasps cup
4.	[2, 0, 1, 2, 1]	person moves near small shelf from table
5.	[2, 0, 2, 2, 0]	release cup

Table 4.4: Example- sequence of states for Level H21

Basically, a cup from the table is moved to the big shelf. Similarly, other branches also perform their specific policies.

4.3.4 Level H3

This is the lowest level of the system which is responsible for navigation from one point to another in the environment. As defined in the section 4.2, this level uses the reward weights learned and feature map to formulate trajectory around the obstacles in the environment.

The actions from middle level dictates the motion in this level. There are 4 location points defined which are offset from each of the rigid objects. Thus any movement requested from above levels is planned between these points. For formulating paths, Dijkstra's algorithm [Dij59] is used over the cost map.

4.3.5 Features

There were two types of features used separately. The first one being euclidean distance to each of the rigid objects combined with indication of grasp or release. The second being the trajectory lengths to reach destination points near rigid objects combined with indication of grasp or release. For both the types of feature sets, the performance was compared for the policy generated for each of the branches in the middle level. From the position of demonstrator, euclidean distances or trajectory lengths would be calculated for the corresponding states in middle level. For grasp or release, the state in which it happens is set as 1. Trajectory lengths were calculated using Dijkstra's algorithm based on the reward map generated for the navigation level.

4.4 Integration and Visualization

In the previous section, the states and action for the levels were explained. However, the planning for a task is only successful when each of the levels work together in conjunction. The success of lower levels propagates to success of the planning by the top level. The policies learned for each of the level and branches are integrated such that there is transfer of control between the levels.

For instance, top level policy decides to move a plate from big shelf to table. The middle level branch for the corresponding macro-action, H24 gets the control. The policy of this branch runs until a terminal state is reached. For each of the steps involving motion, there is a transfer of control to the navigation level H3, which holds control until its task is completed.

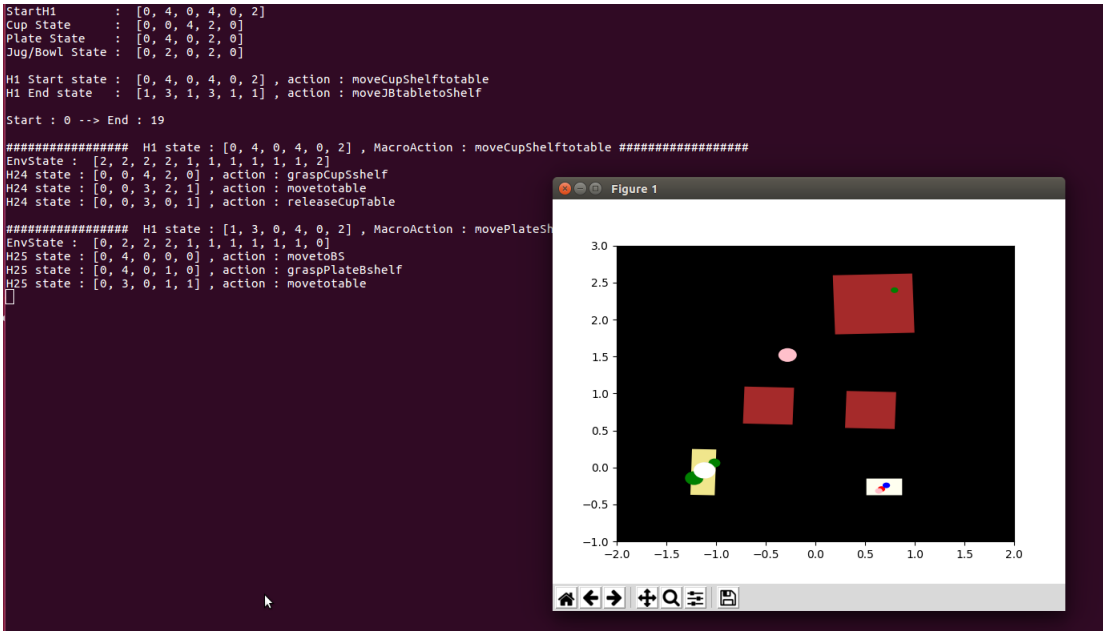

Figure 4.9: Simulation setup of the model

To realise these policies visually, the environment is simulated in 2D. As shown in figure 4.9,the planning can be viewed in parallel. The input for the simulation is simple initial state of the environment as defined in section 4.3.1. The simulation will plan as per the policies until the table is set for a person. The simulation was done for the actual cases from the dataset to see if planning done by the policies is similar to the demonstration.

5 Evaluation

In this chapter, evaluation of the policies learned for each level is discussed. This is done by using cross validation and ground truth comparison. The first section gives a brief idea about the cross validation test performed for the policies. It must be noted that the cross validation tests were performed for top and middle levels. For the navigation level, the comparison of the simulated navigation routes and ground truth is done in the second section. The second section also covers the comparison with ground truth and the whole simulation system for some of the demonstration scenarios for the task of setting up table for one person.

5.1 Cross Validation

To understand how good the policies are, leave-one-out cross validation was performed for each of the levels and branches defined in the system. Thus for each of the tests, one sample from the dataset was left out, IOC was performed to get reward values for the states and then using these new reward values, policy was generated. The left out sample was tested on this policy and its results were compared with actual trajectory. This was performed for all the trajectories from dataset for each of the policies used in the setup.

Also for understanding how good the policies work for the states in any level which were not covered in demonstrations, another test was performed where all the states were tested against the final obtained policy. This was to make sure if any given state could converge to one of the terminal state. As there were cases, the policy gets blocked and jumps between states.

For the validation of middle levels with respect to features used, a comparison is showed in the figure 5.1. In the figure, the blue bar is the result when no features were included, red bar for the features1 as euclidean distance and orange bar for the feature2 as trajectory lengths. As one can perceive, when features are used, the cross validation results are significantly better. Comparing euclidean distance and trajectory length as features, the latter performs in most cases, equal or better than the former. Also intuitively, we can understand that length of the path to a destination is a matter of concern for us humans as compared to direct euclidean distance, provided all other parameters are same for both cases. However, we cannot neglect the fact that there can be multiple parameters for making a decision between destinations. Euclidean distance and trajectory lengths are just two of many for the given scenario. For instance, given the task to pick a plate from table to one of the shelves, the human have to decide which shelf to go to. But the small shelf cannot accommodate a plate because of its small size. Therefore even if trajectory lengths to reach small shelf may be less, it is not a viable destination. Similarly there can be more parameters involved in decision making.

Figure 5.1: Cross validation result for different features

The top level of the system was not integrated with any features and it has 83% success rate with cross validation. Also the middle levels achieve a success rate of 84% on average.

5.2 Comparison with Ground Truth

It is imperative to compare the motion dynamics of the model with the actual demonstration path. Additionally, the decision making sequence of the model has to be compared to the ground truth. There were a total of 21 demonstrations for the task of setting table for one person. All these demonstrations were quite varied in their sequences and lengths. Some of the cases just had couple of steps while some were much longer. All the scenarios were tested in the model and their motion dynamics and decision making sequences were compared. For purpose of distinction of trajectories, colors were used. The description for each of the step is also provided for better comparison. In the simulation model, trajectories tend to overlap since the paths are formulated between set points only. Under this section, only three examples are discussed.

5.2.1 Example 1

The initial state of the environment is [2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2] as define in section 4.3.1. All the movable objects are on the shelves and there is no entity on the table. The person starts from zone 2 which includes the small shelf. The simulated sequence from the model is shown in the table 5.1 and ground truth in table 5.2.

S.No.	States	Meaning
1.	[2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2]	Initial state
2.	[0, 2, 2, 2, 1, 1, 1, 1, 1, 1, 0]	cup from small shelf to table
3.	[0, 2, 2, 2, 0, 1, 1, 1, 1, 1, 0]	plate from big shelf to table
4.	[0, 2, 2, 2, 1, 1, 0, 1, 1, 0, 0]	jug from big shelf to table

Table 5.1: Example1 - Simulated sequence from the model

S.No.	States	Meaning
1.	[2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2]	Initial state
2.	[2, 2, 2, 2, 1, 1, 0, 1, 1, 1, 0]	plate from big shelf to table
3.	[0, 2, 2, 2, 1, 1, 0, 1, 1, 1, 0]	cup from small shelf to table
4.	[0, 2, 2, 2, 1, 1, 0, 1, 1, 0, 0]	jug from big shelf to table

Table 5.2: Example1 - Ground truth sequence from the demonstration

Comparing the sequences, it can be seen that steps 2 and 3 differ in order of execution. Otherwise all the steps are identical in terms of location from which objects were grasped. Also it can be inferred that, though the human was near the small shelf, he decided to pick first plate from big shelf rather than cup from small shelf. Perhaps, orientation of the person could also be a reason for this disparity. Orientation as a feature could also be an option. Figure 5.2 depicts the ground truth path, the simulated path and heat map. The ground truth portrays the actual trajectory of the demonstrator between the table and shelves. Each coloured line shows start and end of a sub-task. In the simulated path, the paths of sub-tasks are superimposed and hence it is not visible distinctly. However, one distinction which is visible is that of the first sub-task which is of the line with color magenta. In the ground truth, the plate is first grasped from big shelf which is denoted by the magenta line moving towards left from a point near small shelf and acquiring it. While in the simulation, a cup is first grasped from small shelf, the magenta line extends from small shelf towards the table. The heat map shows the cost function of the environment with yellow having higher cost than dark blue. The path simulated follows the lower cost trajectory using Dijkstra's algorithm.

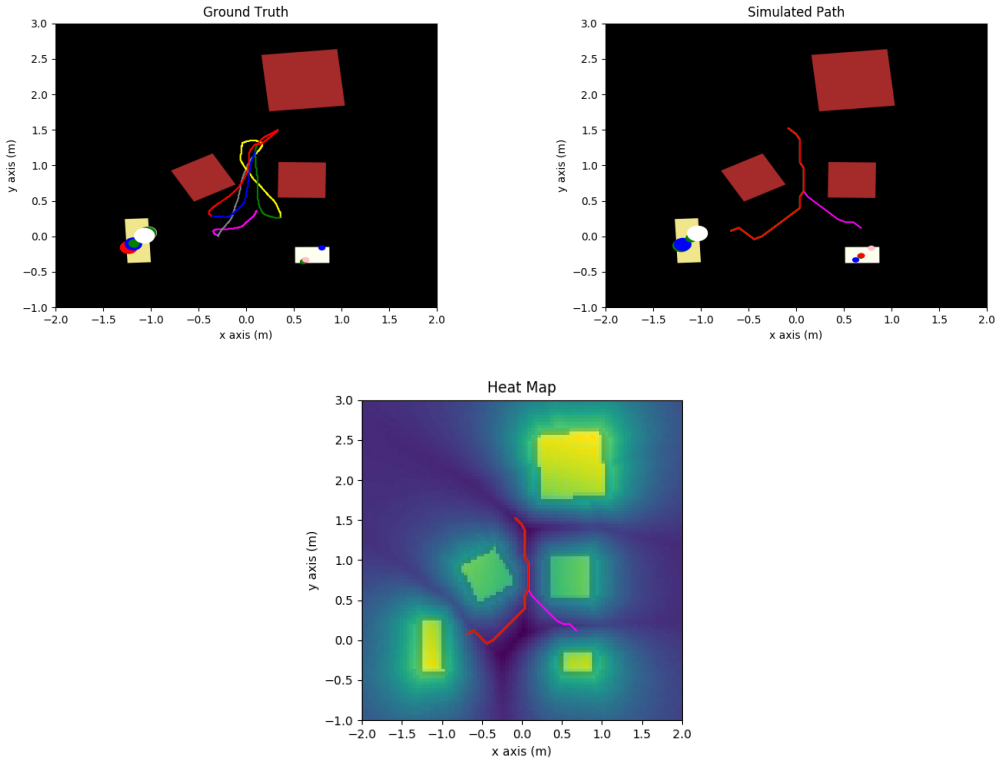

Figure 5.2: Example1 - Ground truth, Simulation path, Heat map

5.2.2 Example 2

The initial state of the environment is [2, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1] as defined in section 4.3.1. Most of the objects are on the shelves except for the jug on the table. This is easier scenario where only a cup and a plate has to be placed on the table. The simulated sequence from the model is shown in the table 5.3 and ground truth in table 5.4.

S.No.	States	Meaning
1.	[2, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1]	Initial state
2.	[2, 1, 1, 2, 0, 1, 1, 1, 0, 1, 0]	plate from big shelf to table
3.	[2, 0, 1, 2, 0, 1, 1, 1, 0, 1, 0]	cup from big shelf to table

Table 5.3: Example2 - Simulated sequence from the model

S.No.	States	Meaning
1.	[2, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1]	Initial state
2.	[2, 1, 0, 2, 1, 1, 1, 1, 0, 1, 0]	cup from big shelf to table
3.	[2, 1, 0, 2, 0, 1, 1, 1, 0, 1, 0]	plate from big shelf to table

Table 5.4: Example2 - Ground truth sequence from the demonstration

In this scenario, again the order of picking cup and plate is different compared to the ground truth. Otherwise all the steps are identical in terms of location from which objects were grasped. Since the paths are planned between already defined points, they overlap going back and forth in the model. Nevertheless, the number of up and down motions are same between the two. The figure 5.3 shows the ground truth, simulated path and heat map. The ground truth depicts the motion of the demonstrator between the table and big shelf back and forth. In comparison to ground truth, the simulated path takes a different route since the model is working with only single fixed points near a rigid object. The heat map depicts the cost function and lower cost path taken simulated using Dijkstra's algorithm.

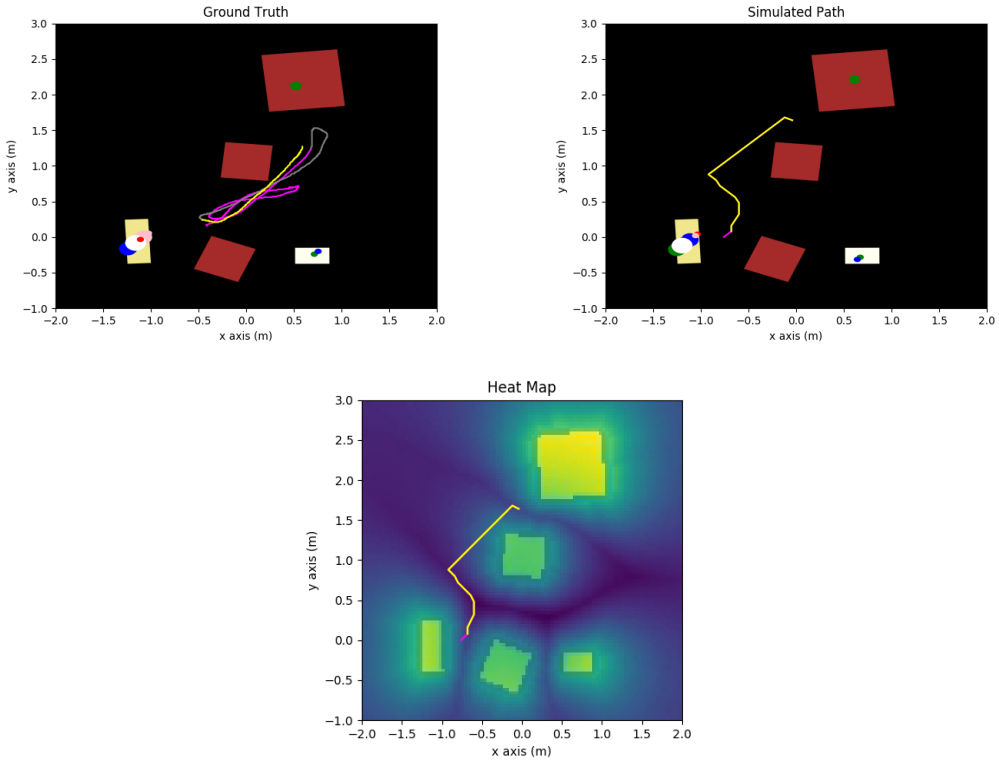

Figure 5.3: Example2 - Ground truth, Simulation path, Heat map

5.2.3 Example 3

The initial state of the environment is [0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1] as described in section 4.3.1. This scenario is quite different from previous cases since there are tasks to remove objects from the table. Initially, there are 4 cups on the table, plates on the big shelf, jug on the big shelf and bowl on the table. The simulated sequence from the model is shown in the table 5.5 and ground truth in table 5.6.

S.No.	States	Meaning
1.	[0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1]	Initial state
2.	[2, 0, 0, 0, 1, 1, 1, 1, 1, 0, 2]	cup from table to small shelf
3.	[2, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0]	plate from big shelf to table
4.	[2, 2, 0, 0, 0, 1, 1, 1, 1, 0, 2]	cup from table to small shelf
5.	[2, 2, 1, 0, 0, 1, 1, 1, 1, 0, 1]	cup from table to big shelf

Table 5.5: Example3 - Simulated sequence from the model

S.No.	States	Meaning
1.	[0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1]	Initial state
2.	[0, 2, 0, 0, 1, 1, 1, 1, 1, 0, 2]	cup from table to small shelf
3.	[0, 2, 2, 0, 1, 1, 1, 1, 1, 0, 2]	cup from table to small shelf
4.	[0, 2, 2, 2, 1, 1, 1, 1, 1, 0, 2]	cup from table to small shelf
5.	[0, 2, 2, 2, 1, 1, 1, 0, 1, 0, 0]	plate from big shelf to table

Table 5.6: Example3 - Ground truth sequence from the demonstration

There are a few differences in this case. In step 3, the model decides to pick plate from big shelf to table which happens only in step 5 for the ground truth. In ground truth, all cups were placed in the small shelf while in simulation, all except one were placed in small shelf. The figure 5.4 shows the ground truth, simulated path and the heat map. The ground truth depicts the actual trajectory of the demonstration between the table and the shelves. The simulated path again formulates trajectory similar to the first example in section 5.2.1 as the cost map is almost identical. Also the heat map is identical with respect to the cost function.

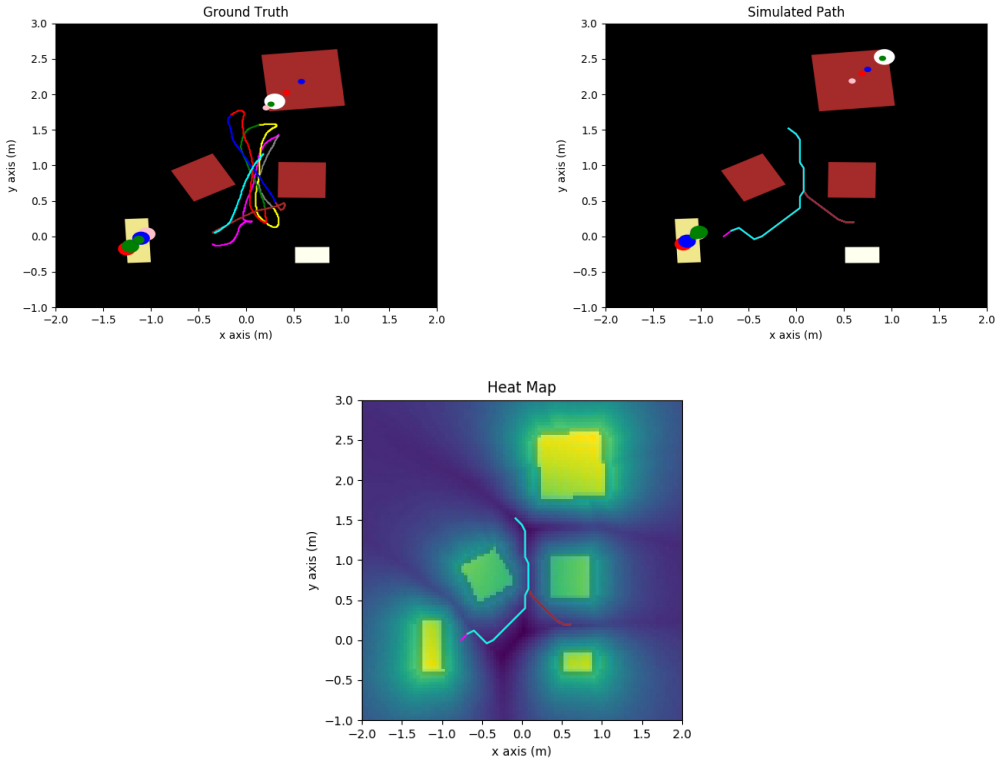

Figure 5.4: Example3 - Ground truth, Simulation path, Heat map

As a conclusion, the model is successful in predicting the sequence of tasks similar if not identical to a human in the environment. The simulated path is also successful in avoiding obstacles of the environment and reaching to the destination requested. The model can be improved further with more offset points around the objects so as to travel more in the environment. Additional features as described in section 5.1 can be incorporated to improve the results. With incorporation of affordance in the model, the prediction of where to place an objects on the surface of a rigid object can be extended.

6 Discussion and Future Work

This work opens up quite a few possibilities for expansion of the idea. The work can be extended for other similar tasks in the environment with few tweaks in the state structure and actions. The middle level can be shared among the tasks since most of the tasks performed can be encoded with such macro-actions. The navigation level too, can be re-purposed. The top level can be updated for other tasks, for example, the task for setting table for three person. The policy can be learned with a new terminal state that encodes the goal. It is possible to further increase the hierarchy levels for certain purposes. For instance, the different racks on the shelves can be encoded and a policy can be learned to decide which rack to choose to pick or place an object. Here, combining with affordances, the system can decide the viable regions in a plane to place an object. Currently, the system randomly places the movable objects inside the boundary confinements of the rigid objects. Affordances can be made use for grasp action as well. This would actually complete the task. Merely, it is not just to have set of objects on the location but also in an arrangement which human would do.

As mentioned in earlier chapters, further features can be included in the model to enhance the results. Orientation of the person, physical attributes of the objects, occupancy metric, etc. can be used as features. Orientation of person can play a role in planning. Humans perceive things that are in front of them and in their field of view. Though a favourable action may be a possibility in a state, it may not be pursued if it is out of the human's perception as explained in 5.2.1.

The physical build of the objects can be a feature as well. For instance, with the height of the jug, it could not be placed on the second rack of the big shelf. Also, plates could not be placed on the small shelf due to its small surface. Such factors can be encoded as features since they act as a hindrance to an action.

With the inclusion of these features, planning can be improved further. When a human makes a plan, a number of factors are considered before a decision is concluded. Quite analogous to this, any kind of planning in the environment will require features and its innate rewards.

7 Conclusion

This thesis presents a model of HIRL which works to imitate a high level task from demonstrations. The long horizon task of setting table for one person is decomposed into sub-tasks. The latent hierarchical state representation for the task is structured in three levels of hierarchy and local rewards for states of each levels are learned. The challenge is in defining the hierarchical state structures for the activity and adopting the right features that encodes the high level task. The proposed framework addresses the problem of learning local sub-tasks and working in tandem with other levels for the common final goal. This work also adopts activity forecasting framework [KZBH12] to learn the rewards for semantics in the environment which is part of the navigation level of the model. The comparison with ground truth in section 5.2 proves that the model is successful in imitating human for the particular task. It is also to be noted that there is still a room for improvement for the model in terms of using of different features which could enhance the results.

Bibliography

[AN04] P. Abbeel, A. Y. Ng. “Apprenticeship Learning via Inverse Reinforcement Learning”. In: Proceedings of the Twenty-first International Conference on Machine Learning. ICML ’04. Banff, Alberta, Canada: ACM, 2004, pp. 1–. ISBN: 1-58113-838-5. DOI: 10.1145/1015330.1015430. URL: http://doi.acm.org/10.1145/1015330.1015430 (cit. on p. 21).

[Bel57] R. Bellman. “A Markovian decision process”. In: Journal of Mathematics and Mechanics 6.5 (1957), pp. 679–684. URL: http://www.jstor.org/stable/24900506 (cit. on p. 17).

[Dee19] DeepMind. AlphaGo. 2019. URL: https://deepmind.com/research/case-studies/alphago-the-story-so-far (cit. on p. 15).

[Die00] T. G. Dietterich. “Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition”. In: Journal of Artificial Intelligence Research 13 (Jan. 2000), pp. 227–303. DOI: 10.1613/jair.639 (cit. on p. 20).

[Dij59] E. W. Dijkstra. “A Note on Two Problems in Connexion with Graphs”. In: Numer. Math. 1.1 (Dec. 1959), pp. 269–271. ISSN: 0029-599X. DOI: 10.1007/BF01386390. URL: http://dx.doi.org/10.1007/BF01386390 (cit. on p. 35).

[FLA16] C. Finn, S. Levine, P. Abbeel. “Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization”. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. ICML’16. New York, NY, USA: JMLR.org, 2016, pp. 49–58. URL: http://dl.acm.org/citation.cfm?id=3045390.3045397 (cit. on p. 24).

[HYY+17] T. Hirakawa, T. Yamashita, K. Yoda, T. Tamaki, H. Fujiyoshi. “Travel Time-dependent Maximum Entropy Inverse Reinforcement Learning for Seabird Trajectory Prediction”. In: 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR). IEEE. 2017, pp. 430–435 (cit. on p. 23).

[Jay57] E. T. Jaynes. “Information Theory and Statistical Mechanics”. In: Phys. Rev. 106 (4 May 1957), pp. 620–630. DOI: 10.1103/PhysRev.106.620. URL: https://link.aps.org/doi/10.1103/PhysRev.106.620 (cit. on p. 21).

[KGL+16] S. Krishnan, A. Garg, R. Liaw, L. Miller, F. T. Pokorny, K. Goldberg. “HIRL: Hierarchical Inverse Reinforcement Learning for Long-Horizon Tasks with Delayed Rewards”. In: CoRR abs/1604.06508 (2016). arXiv: 1604.06508. URL: http://arxiv.org/abs/1604.06508 (cit. on p. 24).

[KKL18] G. Konidaris, L. P. Kaelbling, T. Lozano-Perez. “From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning”. In: Journal of Artificial Intelligence Research 61 (2018), pp. 215–289. DOI: 10.1613/jair.5575 (cit. on p. 24).

[KNST16] T. D. Kulkarni, K. Narasimhan, A. Saeedi, J. B. Tenenbaum. “Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation”. In: CoRR abs/1604.06057 (2016). arXiv: 1604.06057. URL: http://arxiv.org/abs/1604.06057 (cit. on p. 23).

[KZBH12] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, M. Hebert. “Activity Forecasting”. In: Proceedings of the 12th European Conference on Computer Vision - Volume Part IV. ECCV’12. Florence, Italy: Springer-Verlag, 2012, pp. 201–214. ISBN: 978-3-642-33764-2. DOI: 10.1007/978-3-642-33765-9_15. URL: http://dx.doi.org/10.1007/978-3-642-33765-9_15 (cit. on pp. 23, 27, 29, 47).

[MPG+10] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, G. Theraulaz. “The Walking Behaviour of Pedestrian Social Groups and Its Impact on Crowd Dynamics”. In: PLOS ONE 5.4 (Apr. 2010), pp. 1–7. DOI: 10.1371/journal.pone.0010047. URL: https://doi.org/10.1371/journal.pone.0010047 (cit. on p. 23).

[NR00] A. Y. Ng, S. J. Russell. “Algorithms for Inverse Reinforcement Learning”. In: Proceedings of the Seventeenth International Conference on Machine Learning. ICML ’00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. ISBN: 1-55860-707-2. URL: http://dl.acm.org/citation.cfm?id=645529.657801 (cit. on p. 20).

[OPN18] T. Osa, J. Pajarinen, G. Neumann. An Algorithmic Perspective on Imitation Learning. Hanover, MA, USA: Now Publishers Inc., 2018. ISBN: 168083410X, 9781680834109 (cit. on p. 21).

[PR98] R. Parr, S. Russell. “Reinforcement Learning with Hierarchies of Machines”. In: Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10. NIPS ’97. Denver, Colorado, USA: MIT Press, 1998, pp. 1043–1049. ISBN: 0-262-10076-2. URL: http://dl.acm.org/citation.cfm?id=302528.302894 (cit. on p. 20).

[RA07] D. Ramachandran, E. Amir. “Bayesian Inverse Reinforcement Learning”. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. IJCAI’07. Hyderabad, India: Morgan Kaufmann Publishers Inc., 2007, pp. 2586–2591. URL: http://dl.acm.org/citation.cfm?id=1625275.1625692 (cit. on p. 20).

[RPLA18] A. Rudenko, L. Palmieri, A. J. Lilienthal, K. O. Arras. “Human Motion Prediction Under Social Grouping Constraints”. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS). 2018. URL: https://doi.org/10.1109/IROS.2018.8594258 (cit. on p. 23).

[SB98] R. S. Sutton, A. G. Barto. “Reinforcement learning: an introduction MIT Press”. In: Cambridge, MA (1998) (cit. on p. 17).

[SPS99] R. S. Sutton, D. Precup, S. Singh. “Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning”. In: Artif. Intell. 112.1-2 (Aug. 1999), pp. 181–211. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(99)00052-1. URL: http://dx.doi.org/10.1016/S0004-3702(99)00052-1 (cit. on p. 20).

[VOA14] D. Vasquez, B. Okal, K. O. Arras. “Inverse Reinforcement Learning algorithms and features for robot navigation in crowds: An experimental comparison”. In: IEEE-RSJ Int. Conf. on Intelligent Robots and Systems. Chicago, United States, 2014, pp. 1341–1346. DOI: 10.1109/IROS.2014.6942731. URL: https://hal.inria.fr/hal-01105265 (cit. on p. 23).

[Yan19] Yannis Flet-Berliac. The Promise of Hierarchical Reinforcement Learning. 2019. URL: https://thegradient.pub/the-promise-of-hierarchical-reinforcement-learning/ (cit. on p. 20).

[ZMBD08] B. D. Ziebart, A. Maas, J. A. Bagnell, A. K. Dey. “Maximum Entropy Inverse Reinforcement Learning”. In: Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3. AAAI’08. Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438. ISBN: 978-1-57735-368-3. URL: http://dl.acm.org/citation.cfm?id=1620270.1620297 (cit. on pp. 20, 21, 29).

[ZRG+09] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, S. Srinivasa. “Planning-based Prediction for Pedestrians”. In: Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS’09. St. Louis, MO, USA: IEEE Press, 2009, pp. 3931–3936. ISBN: 978-1-4244-3803-7. URL: http://dl.acm.org/citation.cfm?id=1732643.1732694 (cit. on pp. 21, 23, 29).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature