Learning Trajectory Planning for Follow-ahead Robots Using Deep *Q*-Learning

Sahar Leisiazar

Directed Studies Report

Supervisors:
Edward Park, Mo Chen

Mechatronic Systems Engineering
Simon Fraser University
Summer 2022

1 Introduction

Robotic assistance is becoming an increasingly important area of research; however, much progress is still needed before seamless human-robot interactions can become reality. In this course, we investigated the ability to follow ahead of a human, which is necessary for many types of assistant and companion robots. Such robots are in demand in various areas of human life, yet current robot capabilities to perform the task are limited to relatively simple and controlled environments. The ability to follow a person not just behind or side by side, but in front is important for applications such as search and rescue robots during emergencies, capturing video of physical activities, or monitoring elderly people.

Human-following robots face a lot of challenges in real-world environments. For example, the robot needs to distinguish the target person from other people or surrounding objects. The system should be capable of keeping a safe and comfortable angle and distance from the person. Also, it needs to predict the target person's trajectory and navigate to a point in front of the person. There are three different types of human following: following from behind, following from side, and following ahead of the person. Although following from behind or side-by-side is well studied, there are a few papers addressed the problem of following from ahead or leading. In this project, we trained an intelligent model for a mobile robot to follow a target person in front of her/his. We utilized a deep reinforcement learning (DRL) model to train the model and investigated the capabilities and limitations of using only one model in training the follow-ahead robot.

2 Background or Rationale

Human-robot interaction shows a relationship between humans and robots, where robots are required to perform stable, real-time, and safe interactions with a target person in the same natural space. As a typical human-robot interaction, human-following robots (HFR) maintain the relative distance and orientation between the target person and the robot, playing a significant role in service and auxiliary robots. In the common case of service robots, the HFRs should be able to predict the human's trajectory and move alongside in a suitable human-like manner.

Over the last decade, a significant progress in HFRs has been made. A cane-type walking-aid robot is developed by Yan et al. [1] to follow a human user for the safety and supervision of independent walking during rehabilitation training. The robot is designed to help people who can walk on their own but still need some assistance in different daily environments. The human orientation detection and walking intention estimation methods are proposed using a single Lidar sensor. Also, a finite time control method is proposed to follow the target person at a certain distance and orientation. Sekiguchi et al. [2] developed a human companion robot as an assistant. The robot estimates the person's movement and a nonlinear Model Predictive Controller is used to navigate the robot beside the target person. Also, the uncertainty of the person's motion estimation is considered in the controller.

A following mobile robot may encounter various fully or partially occlusions while following a person due to other objects in the environment. The problem of person following in factories is addressed in Zou et al. [3]. A network model (Convolutional attention-based differential LSTM) is trained with unoccluded images to predict the missing information of the occluded images. Aslan et al. [4] compared two different methods of solving complete occlusion handling in HFRs. A

hybrid-supervised approach is proposed by Pang et al. [5]. The approach enables the robot to learn features autonomously from monocular images and to enhance performance via robot-environment interaction. The main focus of the paper is the problems of object occlusion, pedestrian interference, target disappearance, and sensitivity to illumination variance. An RL (Reinforcement learning) agent is first provided with heuristic knowledge from a supervised learning (CNN) policy network and then strengthens its performance through interaction with the real-world by unsupervised actor-critic DRL models. On the other hand, Yao et al. [6] proposed a side-by-side following method in order to avoid occlusion. This method uses a proportional-derivative controller to predict the full occlusion when a target person turns into a corridor intersection. The robot uses 2D laser data to detect the corridor and legs of the target person in order to switch from back-side to side-by-side following.

All of the above papers studied the problem of either side-by-side or following from behind. Moustris et al. [7] proposed a front-following robot with a user intention recognition algorithm. Nikdel et al. [8] employed DRL to estimate the human trajectory and generate short-term navigational goals to guide the follow-ahead robot. However, the DRL model is combined with a trajectory planner to improve the safety and generalizability of the model. Therefore, in this project, we investigated the capabilities of using a DRL model in training a follow-ahead robot. The robot needs to satisfy specific criteria (reward functions) including front-following the person with a certain distance, angle and heading. The training methodology and results with different reward functions are presented in the following sections.

3 Research Methodology

3.1 System Modelling

In this project, we assumed an obstacle-free environment in which the robot is suppose to navigate in front of a walking person. Since both the robot and person move on a 2D plane, we present the global state of the robot and person with (x_r, y_r, θ_r) and (x_p, y_p, θ_p), respectively. (x, y) is the position and θ represents the orientation. The r and p subscripts represent the robot and person, respectively. Fig. 1 illustrates the global position and orientation of the human and the robot.

At each time step, the robot has two options for its linear velocity: 0.25 m/s and 0.5 m/s. Considering 0.4 sec as the time step duration, the robot's traveling distance at each time step is equal to either 0.1 or 0.2 meter. Therefore, the robot's next state can be computed with Eq. 1. ψ and dis are the turning angle and the traveling distance at each time step, respectively.

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} dis \times cos(\psi + \theta) \\ dis \times sin(\psi + \theta) \\ \theta + \psi \end{bmatrix} \tag{1}$$

In order to increase our DRL model performance, we used the relative pose of the person with respect to the robot (x, y, θ) as our DRL model's input.

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} x_p - x_r \\ y_p - y_r \\ \theta_p - \theta_r \end{bmatrix} \tag{2}$$

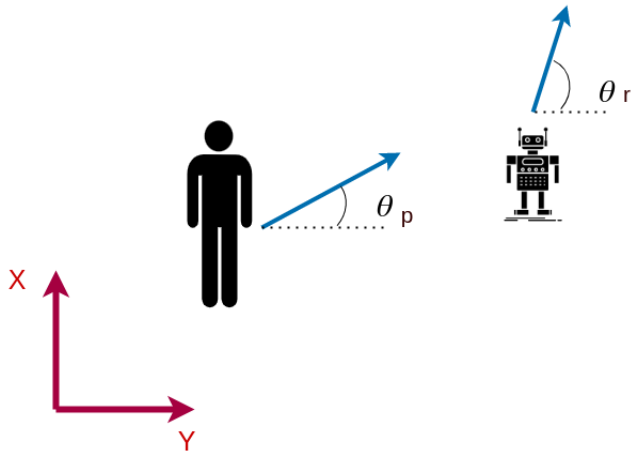

Figure 1: The global position and orientation of robot and person. The blue arrows show the person and robot's orientations.

3.2 Deep Reinforcement Learning

Reinforcement Learning involves an agent making a sequence of decisions. The agent tries to learn the optimal behavior in a potentially complex environment to obtain a maximum reward. This behavior is learned through interactions with the environment and observations of how it responds, similar to children exploring the world around them and learning the actions that help them achieve a goal. This discovery process is dependent on a trial-and-error search. It can learn the actions that result in eventual success in an unseen environment without the help of a supervisor. The main elements of an RL system are (i) an agent or a learner, (ii) an environment the agent interacts with, (iii) a policy that the agent follows to take action, and (iv) a reward signal that the agent observes upon taking actions.

There are several methods for implementing RL including:

- Imitation learning: aims to mimic human behavior in a given task.
- Policy gradient: relies upon optimizing parameterized policies with respect to the expected return (long-term cumulative reward) by gradient descent-based methods.
- Q-learning: presents an idea of assessing the quality of an action that is taken to move to a state rather than determining the possible value of the state is moved to.
- Model-based RL: attempts to overcome the issue of a lack of prior knowledge by enabling the agent to construct a functional representation of its environment.

As mentioned in Sec. 2, our mobile robot needs to make some decisions and predictions in order to follow a person in front of him/her. It needs to predict the target person's future trajectory and navigate to a point on it and be in front of the person. The DRL model inputs the human-robot relative pose, and generate a short-term goal for the robot's future pose. We trained a DRL model with Q-learning method.

The first step of training DRL models is defining a set of observations, actions, and rewards. In this project, the observation is human's state with respect to the robot which is the relative position

and angle. As an action, the model outputs robot's turning angle and linear velocity for the next time step. We defined three reward functions and at each time step, the agent receives sum of them as a total reward. The reward functions are:

- distance reward (R_d): The agent receives higher value if it stays in a distance of 1.5 meters. The reward reduces if the robot gets too close or far from the person. Fig. 2 and Eq. 3 represent the value of distance reward for different person-robot distances (D).

$$R_d = \begin{cases} -(1-D) & D <= 1 \ \ and \ \ D > 0.5 \\ 0.5 \times (0.5 - abs(D-1.5)) & D <= 2 \ \ and \ \ D > 1 \\ -0.25 \times (D-1) & D <= 5 \ \ and \ \ D > 2 \\ -1 & otherwise \end{cases} \tag{3}$$

- person-robot angle reward (R_α): The angle between the person-robot vector and the person-heading vector is defined as α. The less the value of the angle is, the more reward the agent receives. The value of person-robot angle reward for different α are shown in Fig. 3 and Eq. 4.

$$R_\alpha = \begin{cases} 0.5 \times ((10-\alpha)/10) & \alpha < 10 \\ -0.25 \times \alpha/180 & otherwise \end{cases} \tag{4}$$

- heading reward (R_h): The agent receives a higher reward if the robot's heading (θ) is same as the person's heading. The difference of the human and robot's heading angles is defined with β. Fig. 4 and Eq. 5 show the value of heading reward for different β.

$$R_h = \begin{cases} (-\beta + 10) * 0.25 & \beta < 10 \\ -1 & otherwise \end{cases} \tag{5}$$

The total reward (R_t) is defined as:

$$min(max(R_t = R_d + R_\alpha + R_h, -1), 1) \tag{6}$$

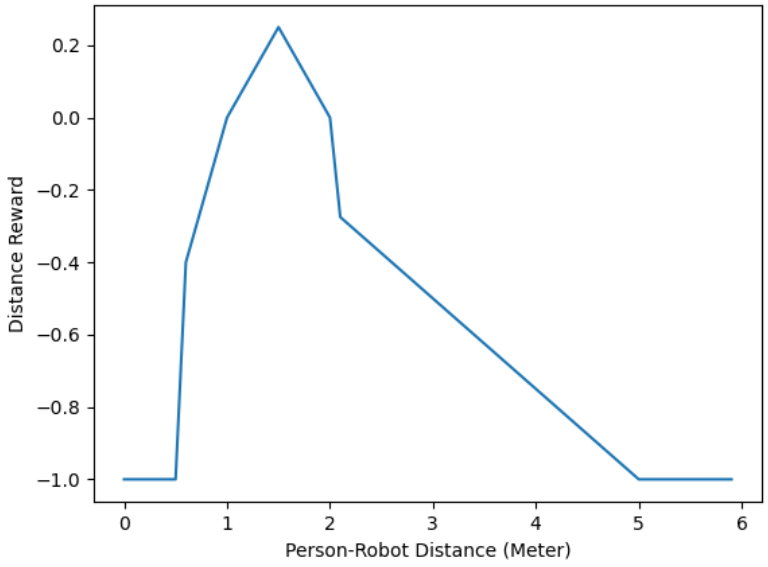

Figure 2: The robot receives a max distance reward (0.25) if it keeps a 1.5-meter distance from the target person. The minimum reward is -1 if the robot gets too close or far from the person.

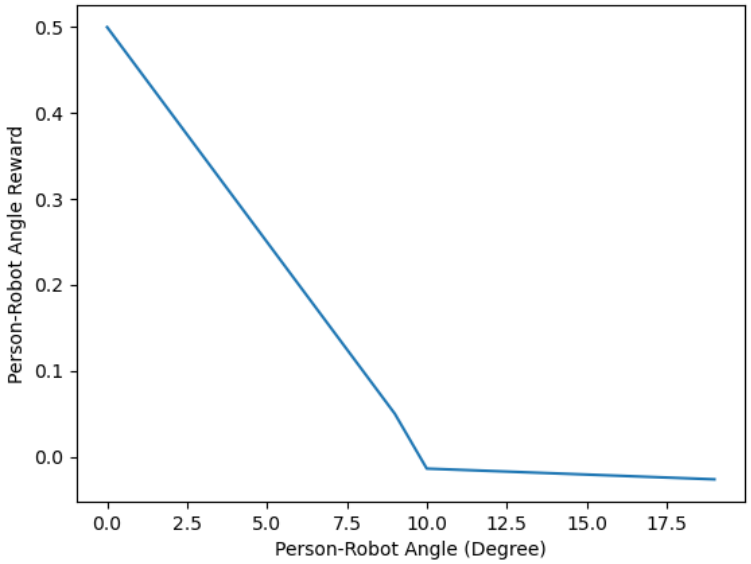

Figure 3: The robot receives a max person-robot angle reward (0.5) if it stays in front of the target person and a negative reward if the person-robot angle is greater than 10 degrees.

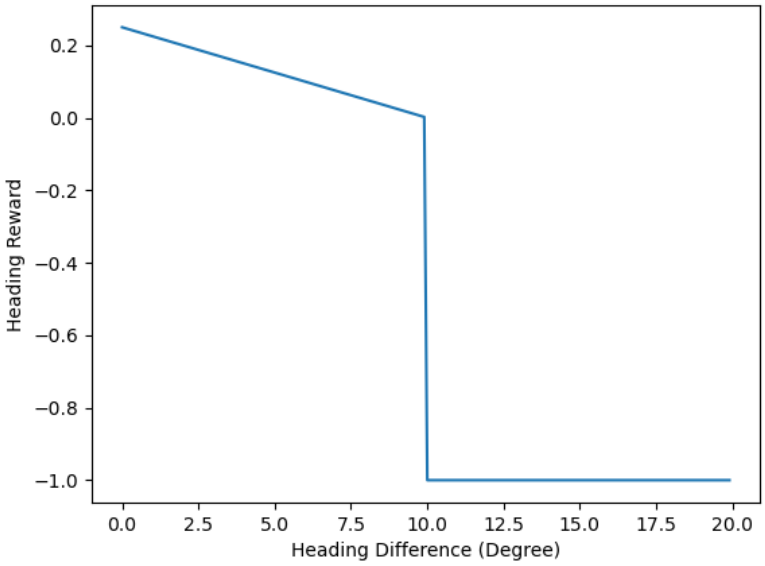

Figure 4: The robot receives a max heading reward (0.25) if the robot's heading is same as the person's heading and a negative reward (-1) if the difference between the person's heading and the robot's heading is more than 10 degrees.

3.3 Deep Q-Learning

Each state of a system is a consequence of its previous state which in turn is a result of its previous state. However, storing all this information, even for environments with short episodes, will become readily infeasible. To resolve this, we assume that each state follows a Markov property, i.e., each state depends solely on the previous state and the transition from that state to the next state. If the agent knew the expected reward of each action at every state, it would perform the sequence of actions that will eventually generate the maximum total reward. This total reward is called Q-value and formalized as:

$$Q(s,a) = r(s,a) + \gamma \, max_a Q(s',a) \tag{7}$$

Equation 7 states that the Q-value yielded from being at state s and performing action a is equal to the immediate reward $r(s,a)$ plus the highest Q-value possible from the next state s'. The γ is the discount factor that controls the contribution of rewards further in the future. In any state of the system, Q-learning helps the agent figure out which action to perform. In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output. Fig. 5 illustrates a neural network model to estimate the expected value of all actions for a given state.

In this project, we trained a Q-model to estimate the expected reward for all the possible actions that the robot can take. The action space includes two linear velocities and three turning angles. Each action defines a turning angle and linear velocity for the robot. At each time step, given a human and robot state, the robot chooses a velocity and angle with the maximum Q-value. The

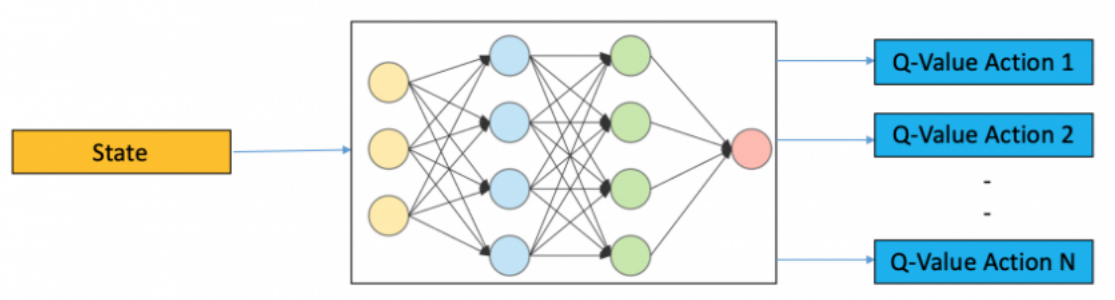

Figure 5: Schematic model of using a neural network that inputs a state and outputs the expected value of each action.

relative distance of the robot from the human is considered as an observation and the reward is defined as Eq. 6.

In order to implement the Q-Learning method, we utilized the Alg. 1. First, we generated a dataset of (s,a,s',r) which is the tuple of current state, action, next state and reward. considering the time step duration of 0.4 second, the duration of each episode is the number of time steps multiplied by 0.4 second. At each episode, the robot and human start from their initial point and move with a sequence of actions. The human's actions are defined in a way to generate different shapes of trajectories such as straight line, circle and waveform. However, the robot's actions are generated uniformly random. At each time step, the robot chooses a random velocity and angle, the reward (Eq. 6) and the next state (Eq. 1) are calculated and the tuple of (s,a,s',r) is stored in a buffer. Moreover, we generated a dataset of size 1000000 episodes with different human trajectories.

To start training the Q-network, first, the algorithm defines a target network which has the same architecture as the Q-network but with frozen parameters. For every N iteration, the parameters from the Q-network are copied to the target network. Next, the algorithm samples some random batches of transitions (s,a,s',r) from the buffer, calculates the target values using the target network and performs gradient descent with respect to the Q-network parameters to minimize the loss. The loss is just the squared difference between the target network and the Q-network.

Algorithm 1 Q-Learning

Take some random action, observe (s,a,s',r) to the buffer
for Number of Iterations **do**
  Sample a mini-batch (s_j,a_j,s'_j,r_j) from the buffer
  Compute $y_j = r_j + \gamma\, max_{a'_j} Q_{\phi'}(s'_j,a'_j)$ using the target network $Q_{\phi'}$
  $\phi' \leftarrow \phi' - \alpha\, \Sigma_j \frac{dQ_\phi}{d\phi}(s_j,a_j)(Q_\phi(s_j,a_j) - y_j)$
  Update ϕ': copy ϕ every N iterations
end for

4 Experiments and Results

In this section, we present the results of training the Q-network with different reward functions (R_d, R_α, R_h) and compare the results with the model trained with the total reward (R_t). We evaluate the capabilities of the DRL method to achieve the maximum reward and front-following the target person. In the following experiments, the person walks on a defined trajectory such as a straight line, circle and waveform trajectories, and the robot needs to take one of the 6 actions in the form of $(angle\ deg, velocity\ m/s)$:

$$(-20, 0.25),\ (0, 0.25),\ (+20, 0.25),\ (-20, 0.5),\ (0, 0.5),\ (+20, 0.5)$$

At each experiment, the training loss, the test reward and the test trajectory generation plots are shown for the agent trained with each reward functions, separately and the total reward function. In the test trajectory plots, the human and robot's trajectories are shown with light and dark blue lines, respectively. Also, the markers on the trajectories represent the human and robot's state at each time step. Moreover, the initial points of human and robot (x_0, y_0, θ_0) are $(0,0,0)$ and $(1.5,0,0)$, respectively.

In all the experiments, the network's hyper-parameters which are listed in Table 1, are same and achieved by try and error. The *batch size* indicates the number of samples selected for training at each iteration, *target update freq* is the frequency of updating the target network, *turning angle* represents the angle if the robot needs to turn to the right or left.

Table 1: The hyper-parameters for training the Q-model

hyper parameter	value
batch size	128
target update freq	30
turning angle	20 deg
number of hidden layers	1
hidden layer size	16
learning rate	0.01
discount factor (gamma)	1
number of iterations	150

4.1 Straight line

In this experiment, the persons walks in a straight line and the robot needs to select an action to move in front of the person. Fig. 6 illustrates the human trajectory and robot's action space. The human's trajectory is shown with blue dashed lines and the robot's action space is shown with 6 red arrows indicate three angles and 2 linear velocities.

The Fig. 7 shows the training loss, test reward and test trajectory generation considering the total reward function. The agent/robot is well trained and selects the best action in order to achieve the maximum total reward. Therefore, the results of training the model with each of the reward functions (R_d, R_α, R_h) for this experiment is not included.

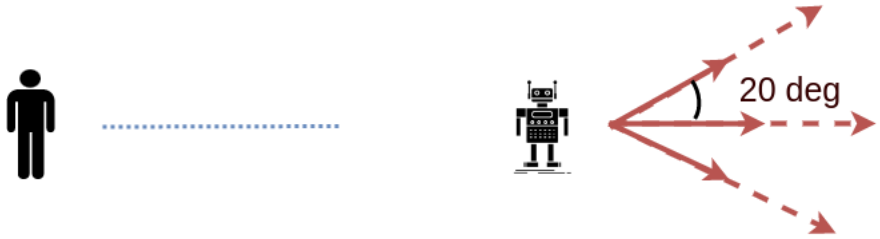

Figure 6: The action space for the robot and human in the straight line trajectory

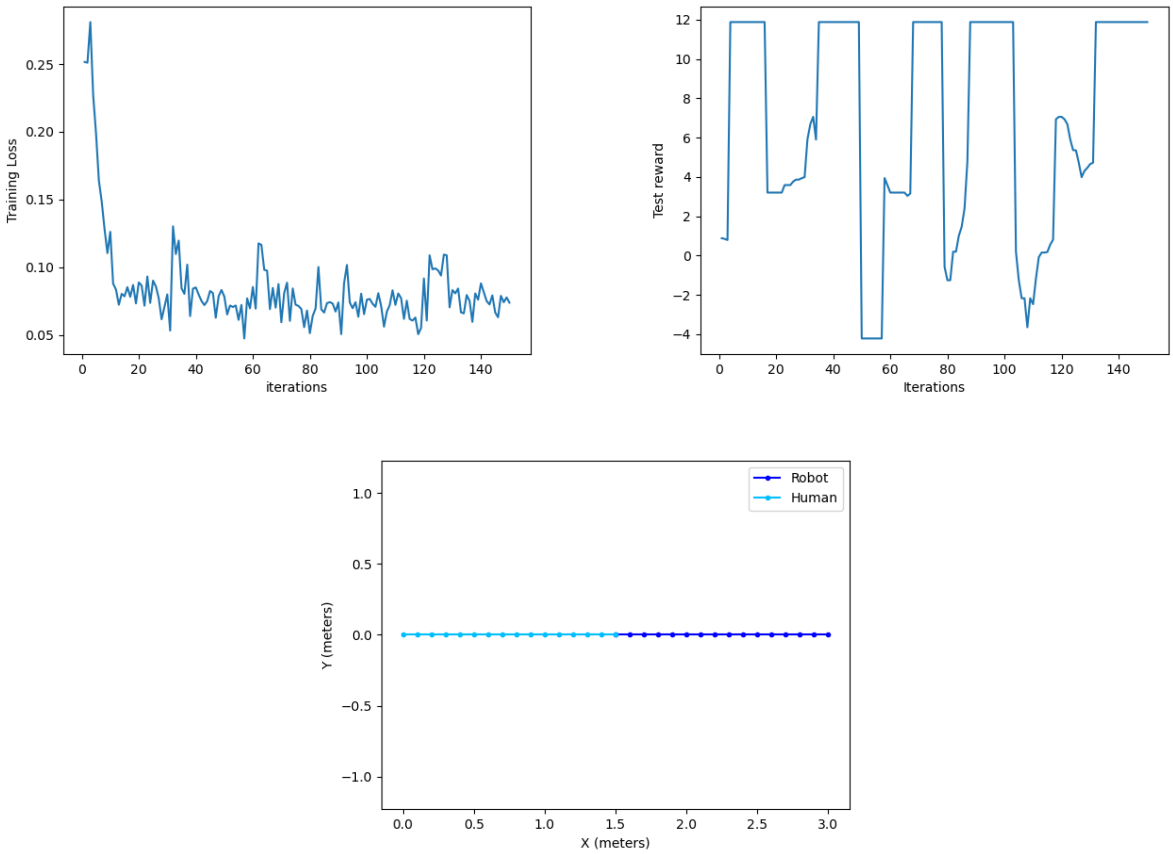

Figure 7: The training loss, test reward and test trajectory estimation plots for straight line trajectory considering the total reward function.

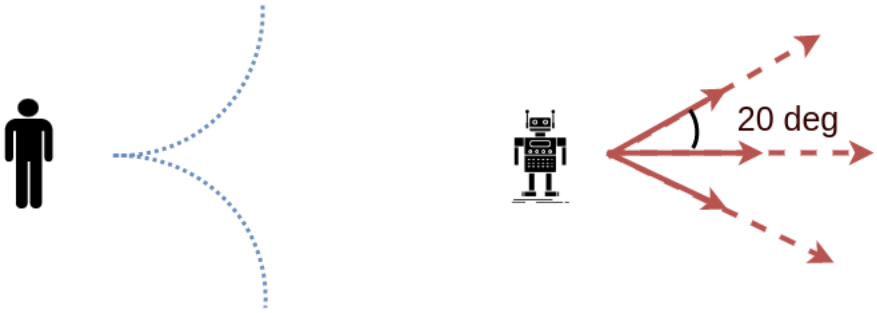

Figure 8: The action space for the robot and human in the circle trajectory. The human's trajectories are shown is dashed blue line.

4.2 Circle

As illustrated in Fig. 8, at each time step, the human turns 10 degrees to the right/left which generates a circle trajectory. The results of experiments with different reward functions are presented in the following sections.

4.2.1 Total reward function

Fig. 9 and 10 show the training loss, the test reward and the test trajectory generation considering the total reward. There are two plots for the trajectories in which the human turns to the right or left. Theoretically, The agent needs to turn to its right/left with maximum velocity to navigate in front of the human. However, the results show that the agent could not choose the best action at every time step.

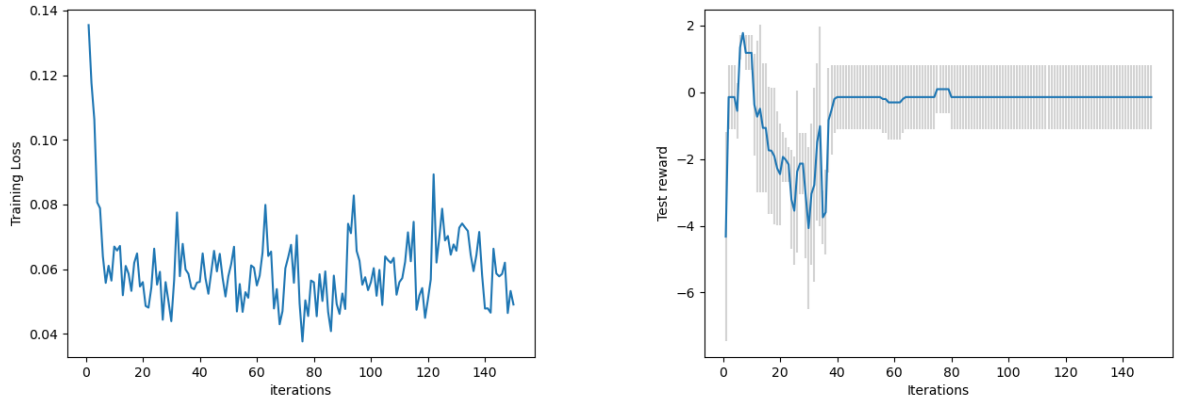

Figure 9: The training loss and test reward plots for circle trajectory experiment considering the total reward function. In the test reward plot, the mean return (sum of reward in an episode) of 100 tests and the std value are shown with a blue line and grey bars, respectively.

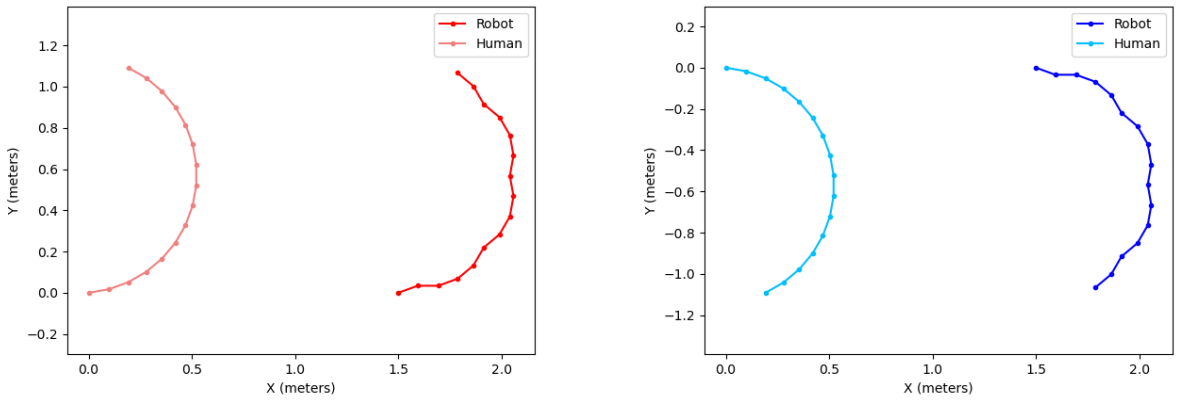

Figure 10: The test trajectory estimation plots for the circle trajectory experiment considering the total reward function. The human and robot's trajectories are shown with light and dark blue/red lines, respectively.

4.2.2 Distance reward

Figures 11 and 12 represent the training loss, test reward and test trajectory generation for circle trajectory considering just the distance reward. The results show that the trained agent chooses a sequence of actions to keep the human-robot distance close to 1.5 meter.

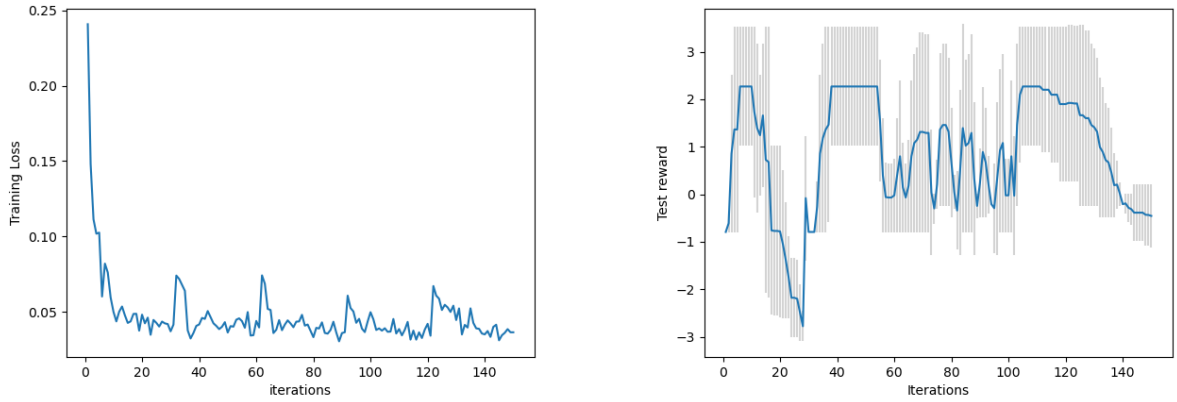

Figure 11: The training loss and test reward plots for circle trajectory experiment considering the distance reward function. In the test reward plot, the mean return (sum of reward in an episode) of 100 tests and the std value are shown with a blue line and grey bars, respectively.

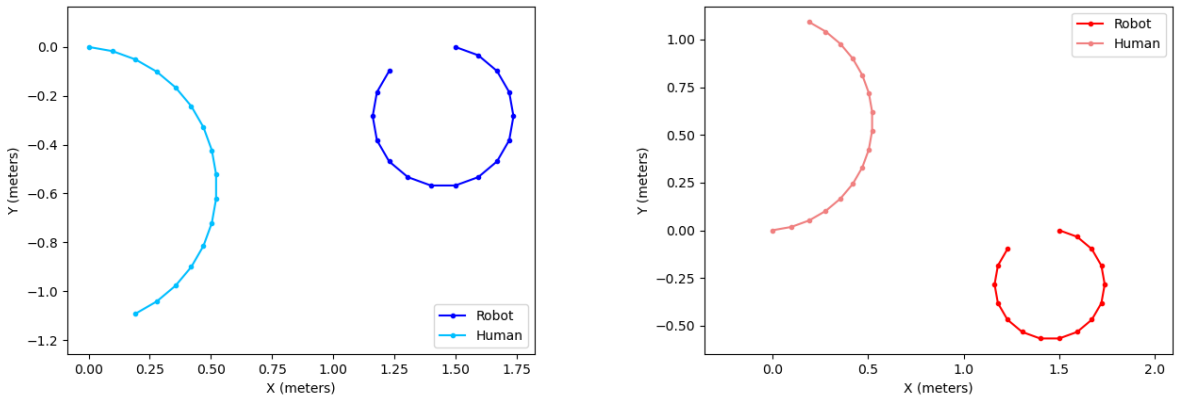

Figure 12: The test trajectory estimation plots for the circle trajectory experiment considering the distance reward function. The human and robot's trajectories are shown with light and dark blue/red lines, respectively.

4.2.3 Person-robot angle reward

The results of training the agent considering just the person-robot angle is shown in fig. 13 and 14. The agent chooses the maximum velocity in order to reduce the α angle and navigate in front of the person.

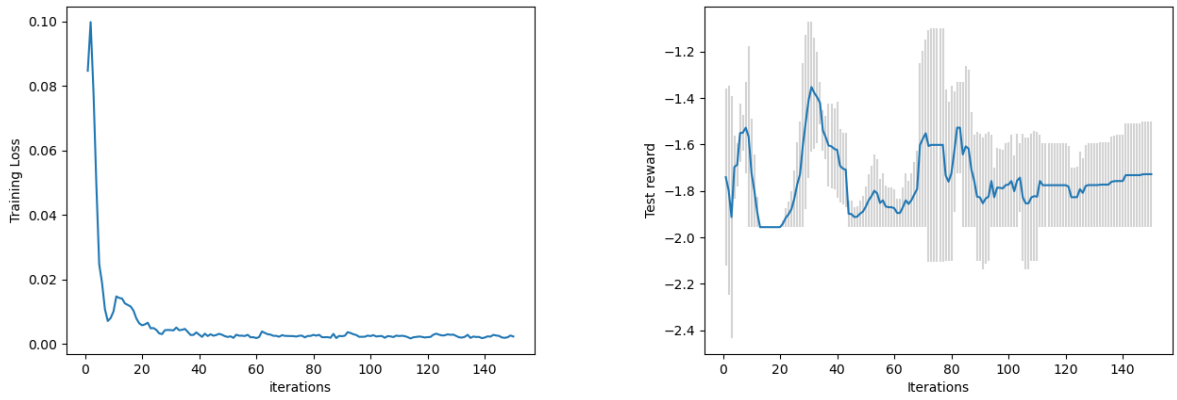

Figure 13: The training loss and test reward plots for circle trajectory experiment considering the person-robot reward function. In the test reward plot, the mean return (sum of reward in an episode) of 100 tests and the std value are shown with a blue line and grey bars, respectively.

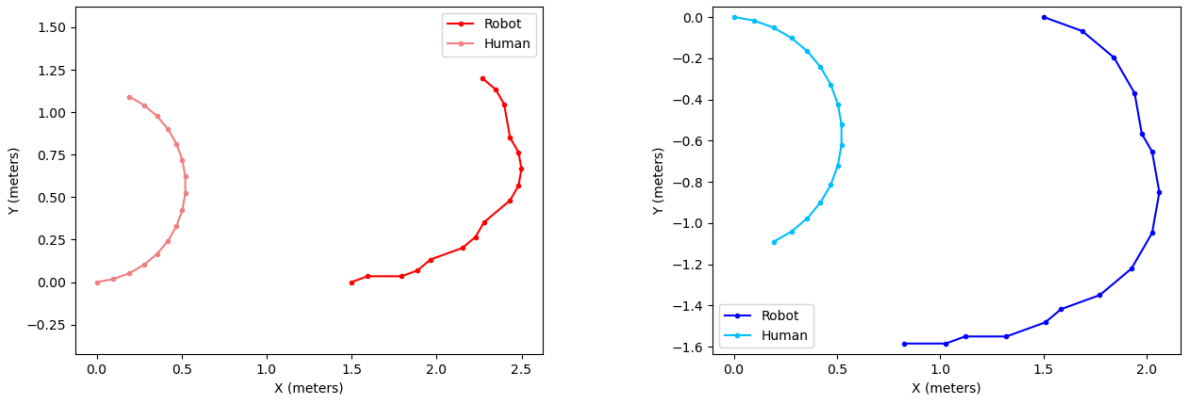

Figure 14: The test trajectory estimation plots for the circle trajectory experiment considering the person-robot reward function. The human and robot's trajectories are shown with light and dark blue/red lines, respectively.

4.2.4 Heading reward

Considering just the heading reward, the training loss, test reward and test trajectory generation are presented in Fig. 15 and 16. The results show that the trained agent attempts to navigate with a same heading as the human.

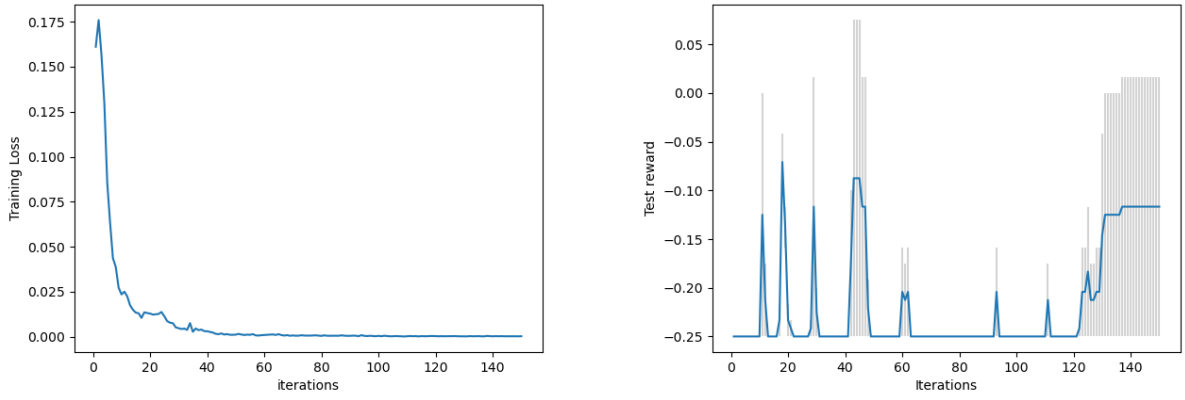

Figure 15: The training loss and test reward plots for circle trajectory experiment considering the heading reward function. In the test reward plot, the mean return (sum of reward in an episode) of 100 tests and the std value are shown with a blue line and grey bars, respectively.

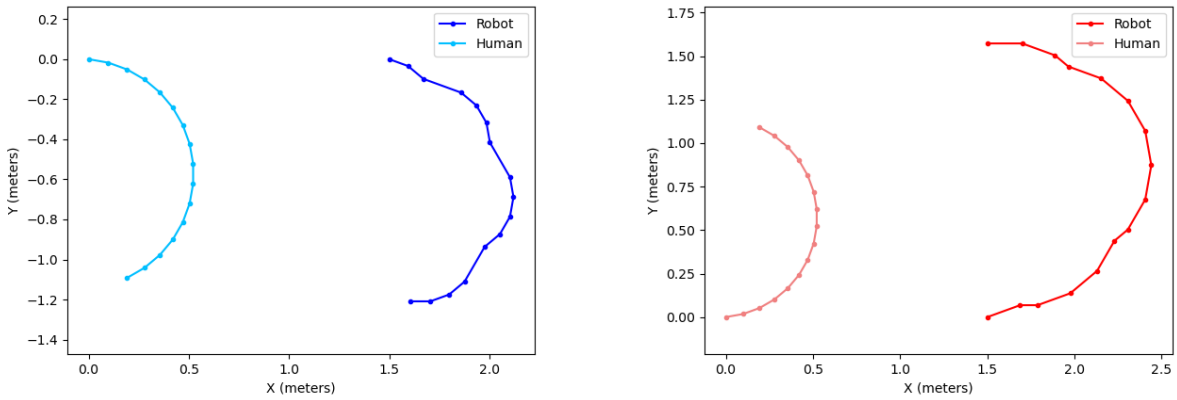

Figure 16: The test trajectory estimation plots for the circle trajectory experiment considering the heading reward function. The human and robot's trajectories are shown with light and dark blue/red lines, respectively.

4.3 Waveform

As illustrated in Fig. 17, the human walks on a waveform trajectory and same as the previous experiments, the robot has 6 actions to take. In the following experiments, we present the results of training the agent with different reward functions.

4.3.1 Total reward function

Fig. 18 shows the training loss, the test reward and the test trajectory generation considering the total reward. However, the results show that the agent generated a trajectory similar to the human's trajectory instead of front-following the person.

Figure 17: The action space for the robot and human in the wave form trajectory

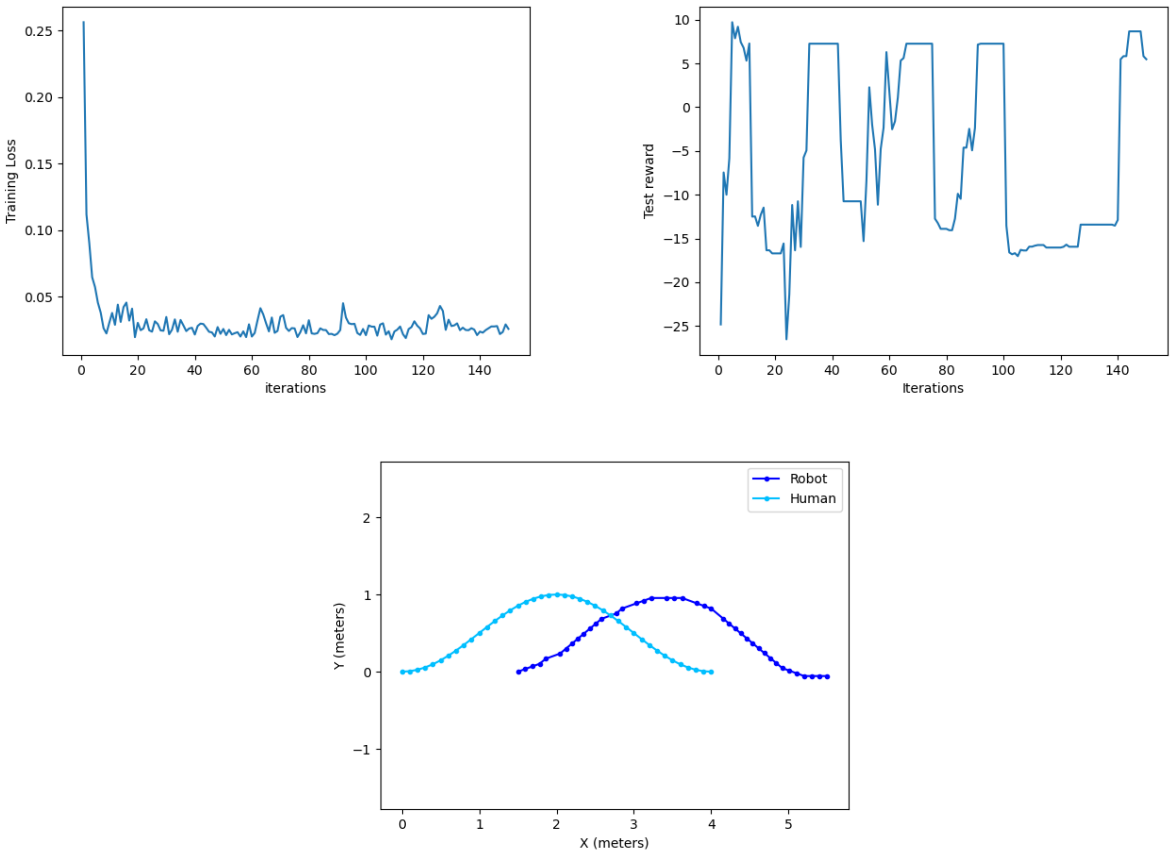

Figure 18: The training loss, test reward and test trajectory estimation plots for waveform trajectory considering the total reward.

4.3.2 Distance reward

The training loss, test reward and test trajectory generation for wavefrom trajectory considering just the distance reward are presented in Fig. 19. The results show that the trained agent chooses a sequence of actions to keep the human-robot distance close to 1.5 meter.

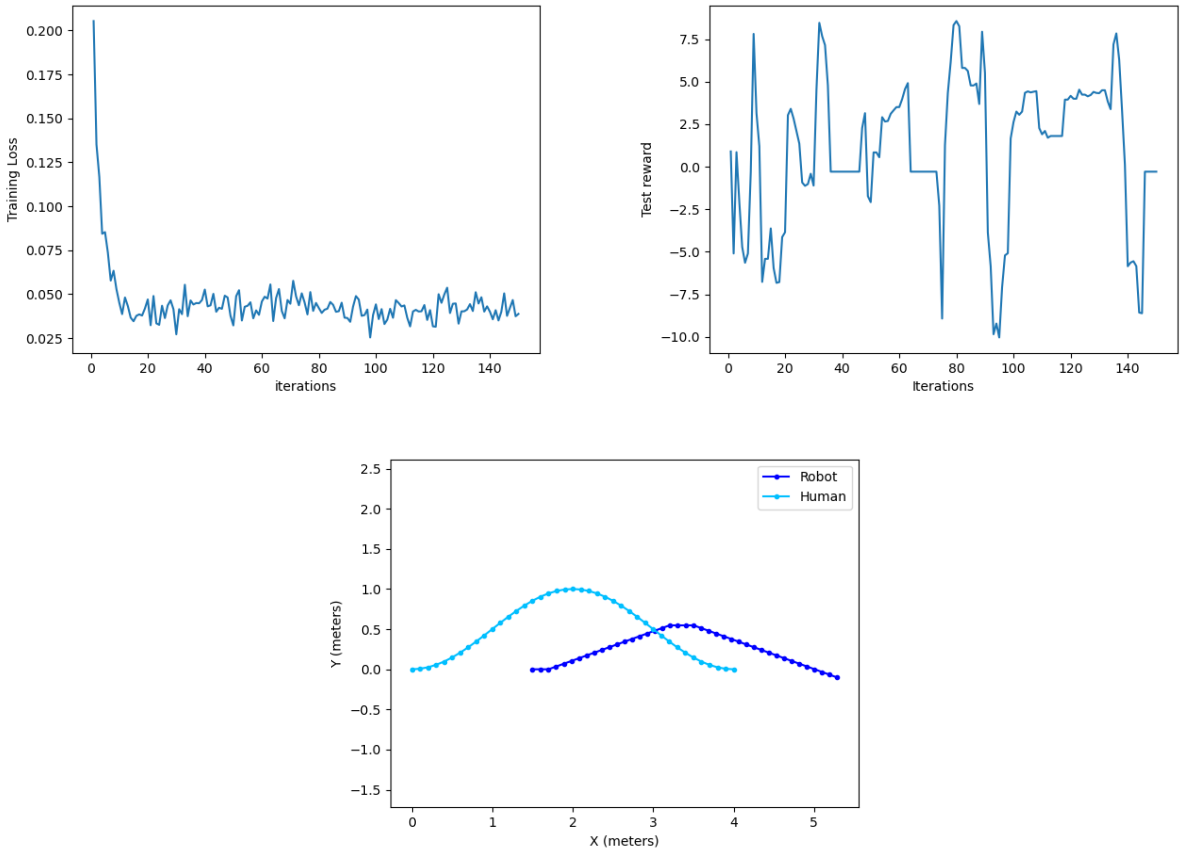

Figure 19: The training loss, test reward and test trajectory estimation plots for waveform trajectory considering just the distance reward.

4.3.3 Person-robot angle reward

The results of training the agent considering just the person-robot angle is shown in fig. 20. The agent chooses the best angle and velocity in order to reduce the α angle and navigate in front of the person.

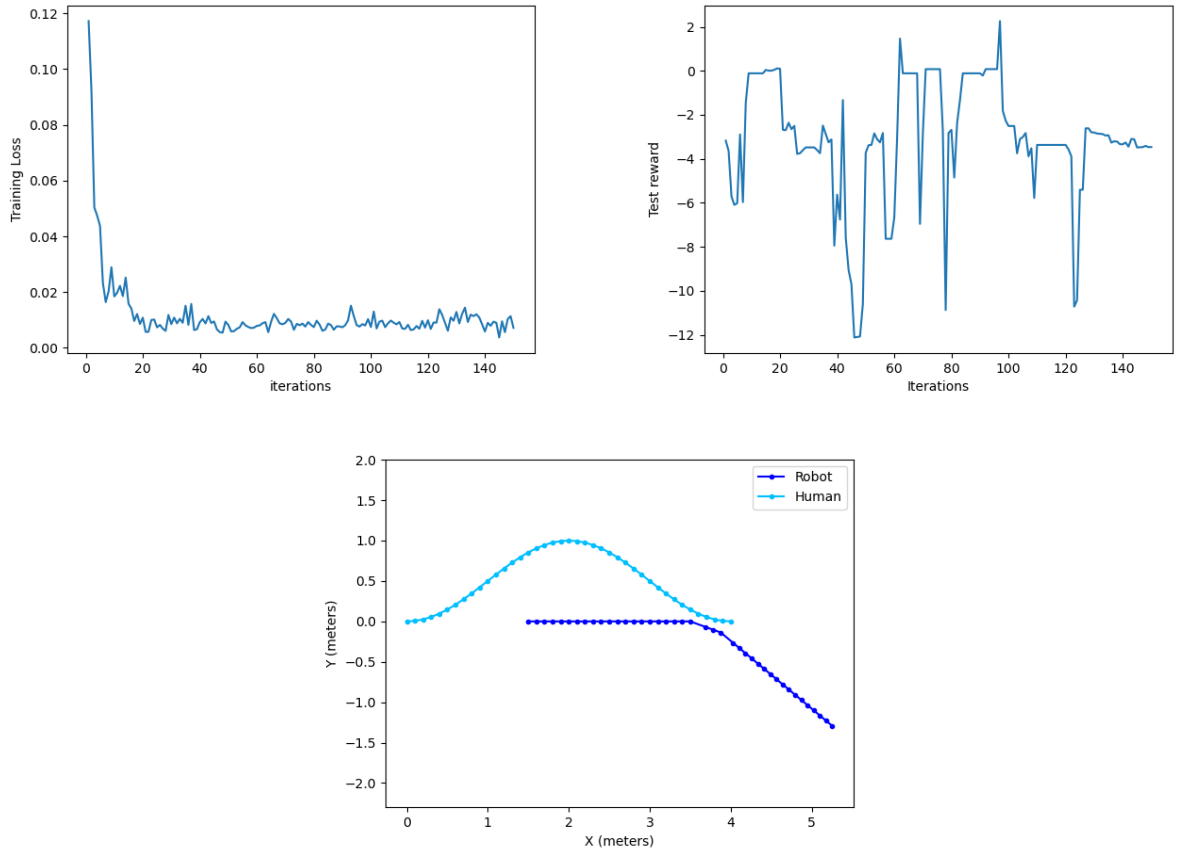

Figure 20: The training loss, test reward and test trajectory estimation plots for waveform trajectory considering just the person-robot angle reward.

4.3.4 Heading reward

Considering just the heading reward, the training loss, test reward and test trajectory generation are presented in Fig. 21. The results show that at each time step, the trained agent selects a sequence of actions to have the same heading as the human.

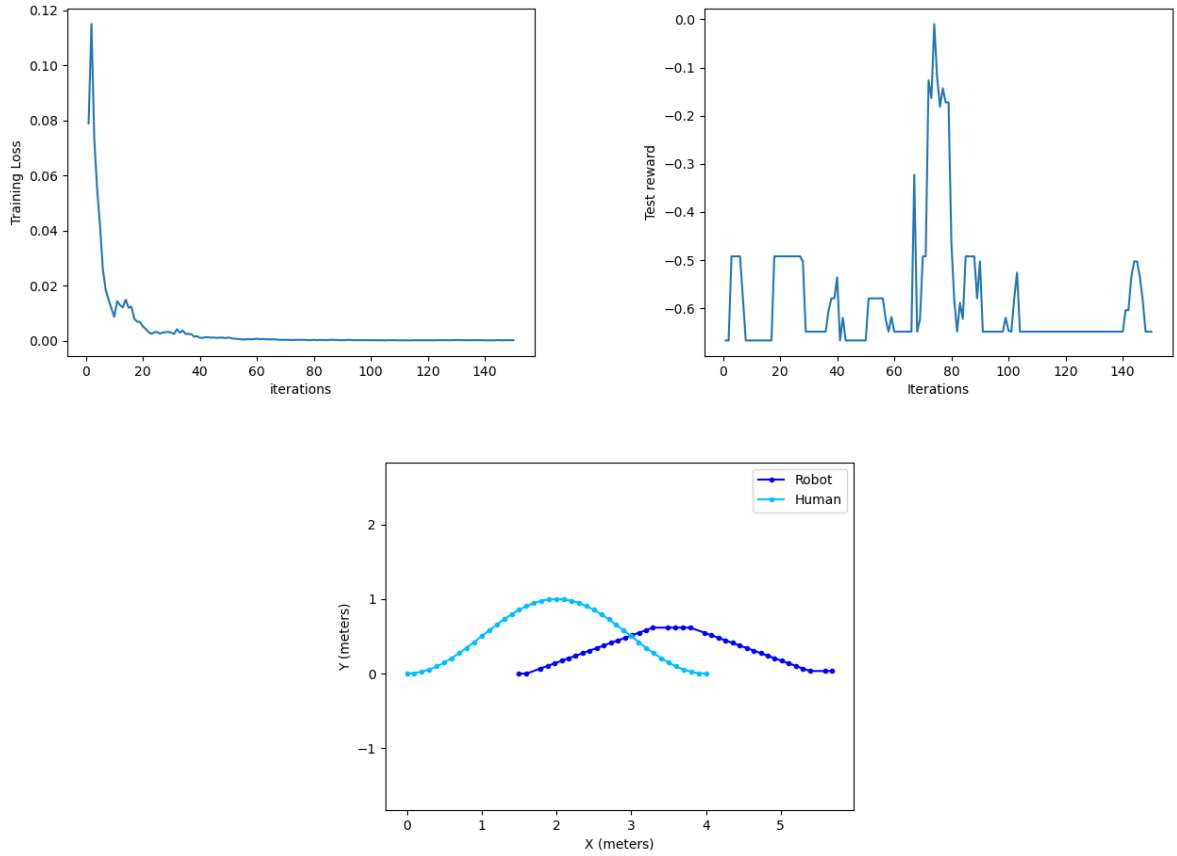

Figure 21: The training loss, test reward and test trajectory estimation plots for waveform trajectory considering the heading reward.

5 Discussion

In this project, we employed the deep Q-learning method of DRL to train an agent for front-following robotic application. We defined three different reward functions and trained the agent with each of them, separately and all together. The performance of the trained agents are tested in three different human trajectories. The results show that the agent can front-follow the target person when the person walks on a straight line. The robot achieves the maximum total reward by choosing the angle equal to zero and the velocity equal to $0.25\ m/s$ to stay in front of the person.

The results for circle and waveform trajectories show that the agent is not able to satisfy all the three criteria (distance, person-robot angle and heading) when the agent is trained with the total reward function. However, when considering just one of the rewards, the agent attempts to choose the best action to achieve the maximum reward. Therefore, we conclude that one of the main disadvantages of using only one DRL model for the front-following robots is the lack of ability to satisfy multiple criteria. It is extremely difficult or data inefficient to apply standard DRL, which makes a sequence of low-level decisions at every time step, to train an agent to account for all of the criteria.

Therefore, we are suggesting to combine Monte Carlo Tree Search (MCTS) with DRL to make high-level decisions such as where the robot should go in the short term, leaving low-level execution to well-studied controllers in both the control theory and learning literatures. Such a hierarchical approach has been shown to be effective in improving data efficiency in recent work [9, 8].

MCTS is a heuristic search algorithm, recently most notably employed in software that plays board games. MCTS has been used in board games like Go, chess, shogi, bridge, and poker, as well as in turn-based-strategy video games. MCTS has also been used in self-driving cars, for example in Tesla's Autopilot software [10]. The focus of MCTS is on the analysis of the most promising moves – or in our case, high-level decisions made by the robot – expanding the search tree based on a random sampling of the search space. The application of MCTS in games is based on many rollouts or simulated trajectories. In each rollout, the game is rolled out to the very end by selecting moves at random. The final game result of each rollout is then used to weigh the nodes in the game tree so that better nodes are more likely to be chosen in future rollouts. The primary objective of the algorithms is to find the best path to follow in order to win the game. In other words, look/search for a way of traversing the tree that finds the best nodes to achieve victory.

The majority of AI problems can be cast as search problems, which can be solved by finding the best plan, path, model, or function. In this project, MCTS can be employed and combined with DRL to observe the human-robot states and make high-level decisions in order to front-follow the target person. The combined method is able to consider and satisfy all the reward functions and improve the performance of the human following robot.

References

[1] Q. Yan, J. Huang, Z. Yang, Y. Hasegawa, and T. Fukuda, "Human-following control of cane-type walking-aid robot within fixed relative posture," *IEEE/ASME Transactions on Mechatronics*, 2021.

[2] S. Sekiguchi, A. Yorozu, K. Kuno, M. Okada, Y. Watanabe, and M. Takahashi, "Uncertainty-aware non-linear model predictive control for human-following companion robot," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8316–8322.

[3] J. Zou, "Predictive visual control network for occlusion solution in human-following robot," *Assembly Automation*, 2021.

[4] M. F. Aslan, A. Durdu, K. Sabanci, and M. A. Mutluer, "Cnn and hog based comparison study for complete occlusion handling in human tracking," *Measurement*, vol. 158, p. 107704, 2020.

[5] L. Pang, Y. Zhang, S. Coleman, and H. Cao, "Efficient hybrid-supervised deep reinforcement learning for person following robot," *Journal of Intelligent & Robotic Systems*, vol. 97, no. 2, pp. 299–312, 2020.

[6] H. Yao, H. Dai, E. Zhao, P. Liu, and R. Zhao, "Laser-based side-by-side following for human-following robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2651–2656.

[7] G. P. Moustris and C. S. Tzafestas, "Intention-based front-following control for an intelligent robotic rollator in indoor environments," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–7.

[8] P. Nikdel, R. Vaughan, and M. Chen, "Lbgp: Learning based goal planning for autonomous following in front," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3140–3146.

[9] A. Li, S. Bansal, G. Giovanis, V. Tolani, C. Tomlin, and M. Chen, "Generating robust supervision for learning-based visual navigation using hamilton-jacobi reachability," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 500–510.

[10] https://www.youtube.com/watch?v=j0z4FweCy4Mab$_c hannel = Tesla$.