Human-Friendly Autonomous Robot Navigation by Deep Reinforcement Learned Collision Avoidance

Student: Yu Han Daisy Wang, Independent Schools Foundation Academy, Hong Kong

Supervising Teacher: Dr. Yee Pan Angelo Leung, Independent Schools Foundation Academy, Hong Kong

Academic Advisor: Jia Pan, University of Hong Kong, Hong Kong

2019

Abstract

With the development of the robotics techniques in recent years, service robots have taken more and more forefront roles, going from behind the scenes work to direct interactions with humans in households, offices, and public areas [1], [2]. However, existing works [3], [4] often overlook the interaction with humans. For instance, walls and humans are uniformly modelled as obstacles in the navigation process. In this case, human social space is intruded and navigation rules are violated. In this project, we propose a human-aware motion planning approach based on Deep Reinforcement Learning [5] to enhance human-robot interactions (HRI). In particular, we first differentiate humans from other objects through human detection techniques [6]. Second, the human information and the associated behavioral preference are integrated into the local map of the robot's motion planning framework. Third, the motion planning approach leverages the local map information to generate a human-friendly trajectory that can optimally balance the robot's safety, efficiency, and social-consistency during the navigation. The proposed approach will be implemented on a mobile robot to demonstrate dexterous and human-friendly autonomous navigation in complex scenarios with rich human activities, such as office and school. In addition, based on our developed autonomous navigation robot, we will further investigate the importance of human-awareness for service robots, by studying the different responses of human crowds with various densities when interacting with a robot taking or not taking into account the social rules. In this way, we hope to contribute to a deep understanding of human-robot social interactions within crowds.

Keywords: Service Robots, Human-Robot Interaction, Autonomous Navigation, Deep Reinforcement Learning.

Table of Contents

Chapter 1 Introduction

1.1 Background

With the development of the robotics techniques in recent years, service robots have taken more and more forefront roles, going from behind the scenes work to direct interactions with humans. These service robots have taken both virtual and physical forms. In daily life, many virtual service robots exist in the from of ”intelligent assistants”, such as Siri [7], which has been widely applied to streamline various mundane tasks on people’s personal electronic devices via voice control. In terms of robots with physical representations, although they are commonly used in many industry environments to streamline production, most people seldom interact with them in daily life. One of the reasons is that so many robots cannot enter people’s daily lives is due to the fact that they lack the ability of autonomous navigation, making them difficult to integrate into everyday life.

For the robots that interact with humans, it is imperative that they account for human-centered designs instead of just efficiency in manufacturing, in order to ensure optimal integration into daily life. Currently, many techniques fail to take into to account the contexts in which they will be deployed in, leading to many solutions that look good on paper, but fail to deliver in real life. This leads to scenarios from hard to use user interfaces, to innocent people being misidentified and facing arrest due to lack of representation in facial recognition training data [8]. Scenarios like these reveal a need for more human-centered designs, designs which allow for more harmonious human-AI interactions. Only through creating more harmonious human-robot relationships can we start successfully integrating AI into people’s everyday lives. As autonomous navigation becomes a more and more important part of our daily lives, it is imperative that the solutions being proposed are human-centered, in order to ensure that solutions are able to integrate well into daily life.

1.2 Challenges

However, there are two main challenges for a robot to conduct human-friendly navigation. Firstly, traditional approaches often fail to navigate a robot in crowds as they are highly complex and dynamic. Specifically, traditional navigation algorithms tend to treat navigation as an energy function, and approach solving this function as an optimization problem, with a collision free path being seen as the most optimized solution [9]. Secondly, when observing other approaches, we can see that many solutions chose to treat all obstacles alike without human-centered awareness. This is due to lack of visual perceptibility, which allows the robot to differentiate between human and non-human obstacles. Even in robots

with visual perceptibility, human-centered design is not always implemented. Both of these reasons lead to navigation algorithms that lack human-awareness, fail to respect basic social norms regarding navigation, and thus have a hard time integrating into people's daily lives.

To address these challenges, visual perceptibility should be achieved to allow the robot to be able to distinguish between different objects. However, the robot should not only be able to distinguish humans from other obstacles, it should also be able to act accordingly, and act in a human-friendly way of navigation. Hence, further processing of the information received from the sensors needs to be conducted in order to ensure that the robot will act differently towards humans and other non-human obstacles. Lastly, this system should be able to run in real time in order to ensure that the robot is able to respond to changes in its environment in order to cope with the demands of crowd navigation.

Currently, most existing approaches for motion planning, learning based or not, tend to treat all obstacles alike. These approaches may succeed at conducting collision-free navigation, but they fail to create human-friendly navigation as they do not respect basic human social norms. Earlier work done with the approach of velocity obstacles by Fiorini et al. [10] simply accounted for the velocities of other agents in the system. Similarly, later approaches such as Optimal Reciprocal Collision Avoidance (ORCA) also calculated the optimal approach solely based on the measured velocities of other agents in the system as well, failing to account for human-robot interactions [11]. Thus, we can see that there is a dire need for motion-planning methods that are human-aware, and are able to respect basic human social norms.

1.3 Human-Friendly Navigation

In this paper, we would like to define human-friendly as being able to observe some human social norms with regards to movement and navigation, such as respecting personal social space when possible. By respecting basic human social norms, the robot is able to be better ensure comfort of humans, thus helping to create a more positive relationship between the service robot and humans, and creating better Human Robot Interactions (HRI).

Our contributions can be summarized as follows:

- We use of visual perception to capture social information of the surrounding environment
- We present a sensor-fusion module to generate a laser scan with social awareness
- We propose a deep reinforcement learning based human-aware motion planning approach

1.4 Organization of Report

This report is organized into six chapters. Chapter two will present a literature review on traditional and learning based approaches for robot navigation, as well as human friendly navigation. Chapter three covers the initial problem formulation, as well as the deep reinforcement learning of this project. Chapter four will discuss the approaches taken for this project, regarding visual perception, the sensor-fusion module, and human-friendly navigation. Chapter five covers the experiment setup, including hardware setup, and software setup, as well as the experiment results. Finally, chapter six will conclude the report.

Chapter 2 Related Work

2.1 Deep Learning-based Visual Perception

To achieve human-friendly navigation, visual perception methods need to be introduced. Recently, many advanced deep learning approaches have been widely applied in object recognition, object detection, and semantics segmentation.

The field of deep learning, as we know it today, was due to a breakthrough in 2006 [12]. Hinton el al. [12] demonstrated that by initializing each layer in their neural network via unsupervised learning, then finishing with supervised learning, they could significantly outperform traditional neutral networks, prompting renewed interest in neural networks. This work was furthered in 2012 with AlexNet [13], a convolutional neural network for image detection. AlexNet was groundbreaking for its accuracy in image detection, which was achieved through use of rectified linear units (ReLU) to reduce training time, using dropout to counter overfiting, and overlap pooling to reduce the overall size of the network [13]. However, AlexNet is still very computationally demanding to run. In order to counter this problem, Ross Girshick developed R-CNN [14]. R-CNN differs from AlexNet through its use of selective search to select 2000 regions, called region proposals, from the input photo [14]. This greatly decreases the number of regions that has to be processed by the convolutional neural network, making it less computationally demanding than AlexNet. This work was furthered with Faster R-CNN [6], which eliminated the use of selective search by allowing the network to learn how to generate region proposals. Despite Faster R-CNN's increase in speed over R-CNN, it is still not fast enough for running in real time.

To counter the problems posed by R-CNN and Faster R-CNN, Joseph Redmon developed the You Only Look Once (YOLO) framework [15]. YOLO differs from the previous approaches in how the image is inputted into the network. Instead of analysing at the entire image, or a set number of region proposals, YOLO does two things: First, bounding boxes are drawn on the image to indicate regions which could potentially contain objects. Then, YOLO splits the image into an S by S grid, and employs a single convolutional network to detect the class probability of each bounding box. Bounding boxes with class probabilities above the threshold are then used to predict the location of objects in the image. Through this method, YOLO was able to massively outperform the other competitors, achieving true realtime detection [15].

2.2 Traditional Approaches for Robot Navigation

Regarding the problem of robot navigation and obstacle avoidance, much work has already been undertaken. Current works can be divided into three categories: map-based navigation, map-building navigation and map-less navigation [16].

Map-based navigation methods leverage a model of the environment provided by humans for purposes of localization and path planning. For instance, Chatila et al. [17] proposed a sensor-fusion approach to process the model of the environment and provide a sequence of landmarks to be used navigation. Map-based navigation rely on map information provided by the user, limiting these types of methods to static scenes.

To overcome this limitation, some researchers have proposed map-building navigation, which use sensor input to map a model of the robot's environment for navigation. Most of these methods are based off of the Simultaneous Localization and Mapping (SLAM) approach. Some of the earlier works include systems such as RHINO [18], a mobile robot which used learning techniques to map and plan its path and react in real time to its environment. Another early example would be RHINO's successor, MINERVA [19], a robot designed to act as a tour guide in the Smithsonian. This robot employed use of similar learning techniques, mapping and planning its path in real time. Both of these approaches rely on first creating a map of their surroundings, then utilizing this map for path planning and navigation, and adjusting to obstacles via local obstacle avoidance algorithms. However, these approaches require some degree of prior knowledge to describe and rebuild the map, thus still limiting the usages.

To address this issue, other researchers have proposed map-less navigation, which does not require prior description of the environment. Van Den Berg et al. [4] proposed a method of reciprocal velocity obstacles (RVO) for multi-agent navigation. This method is based on the assumption that all other agents in the system operate on similar collision avoidance reasoning, and plans a collision free path based on this approach. However, this assumption does not allow the method to function well in crowds, as human crowd behaviour violates the fundamental assumption made in the method. This work was furthered by Optimal Reciprocal Collision Avoidance (ORCA) [11]. In this method, each robot was treated as independent of each other, and the optimal approach was calculated based on the measured velocities of other robots. Here, the calculation of the velocities of other robots allowed for much more accurate calculations of collision free paths, especially with respect to navigation in crowds. However, NH-ORCA suffers from the freezing pedestrian problem, in which two agents that face head on collision will simply freeze instead of taking slight detours [20], thus failing in crowd navigation as well.

To summarize, traditional approaches, although successful at robot navigation, do not fair well tasked at navigating crowds due to their highly dynamic and complex nature. Traditional approaches' reliance on hand-

crafted parameters means they are not robust, and struggle adapting to different scenarios. Thus, some machine learning based approaches are proposed.

2.3 Learning Based Approaches for Robot Navigation

After Lillicrap et al.[21] proposed the Deep Deterministic Policy Gradient (DDPG) algorithm and applied it to continuous control, much work has been undertaken in reinforcement learning (RL). Deep RL algorithms have shown great successes in many areas, such as navigation [22], [23], exploration [24], [25] and motion planning [26]. Bojarski et al. [27] solely used raw camera data for simple autonomous driving in the real world, demonstrating the possibility of deep learning in navigation. Chen et al. [28] introduced a value network of deep reinforcement learning to model human-robot cooperative behaviors in dynamic environments. Ziebart et al [29] proposed an inverse reinforcement learning (IRL) based navigation algorithm, reducing the problem of crowd navigation to a utility function, and using the principle of maximum entropy, specify an IRL structure.

To summarize, existing learning-based methods tend to focus more on predicting the trajectory of different agents in multi-agent environments during the path planning process for the purposes of obstacle avoidance. However, they tend to treat all agents in the system alike, thus failing to respect basic social norms, resulting in great difficulty for these projects to be successfully integrated into daily life.

2.4 Human-Friendly Navigation

Regarding the problem of human-friendly navigation, work has been done as early as 2000. In 2000, Alami [30] presented his robot “Diligent”, which contained a navigation planner that could account for tasks set by a human supervisor, and act accordingly. Early work regarding human-friendly path planning was proposed by Peter et al. [31], where inverse reinforcement learning was used to generate human-like behaviour while navigating through crowds. Guzzi et al. [32] presented a local sensing algorithm which proactively predicted the paths of humans, then used that information, as well as a heuristic model, to behave in a way which was “predictable and legible” and “acceptable” towards humans. This would then be furthered by Pfeiffer et al. [33] in 2016, who proposed a data-driven model that aimed to predict the actions of human agents, as well as produce movement that would be ”predictable” for humans. Chen et al. [26] proposed an approach towards socially-aware navigation using deep learning. They introduced socially-aware collision avoidance in an reinforcement learning network, where pairs of simulated agents navigate around each other to learn a human-friendly policy, which is then generalised to a multi-agent systems.

This allows their robot to demonstrate autonomous navigation at human walking speed in large crowds.

Kim et al. [34] proposed an IRL-based method for the purposes of learning human movement based on trajectories generated by an expert to learn socially acceptable behavior. This learned policy is then used to achieve human-centered navigation. Similar work was done in Pedestrian ORCA (PORCA) [20], where reinforcement learning was again used to model pedestrian behaviour. PORCA combined a pedestrian motion model and a partially observable Markov decision process algorithm in order to create a planning system that considered both intentions and interactions of pedestrians, despite the uncertainty in these aspects.

Through the above, we can see that in order to engage in human-friendly navigation, learning techniques can be used in order to ensure that the robot is able to detect humans so that path planning can fully account for humans.

Chapter 3 Preliminary

3.1 Problem Definition

The robot navigation problem can be typically formulated as a Partially Observable Markov Decision Process (POMDP). A POMDP differs from a Markov Decision Process in the fact that its states are not fully observable, thus the decisions-making process is based on incomplete information, formulated from a set of observations. Generally, the POMDP framework can be described as a 5-tuple containing $(\mathbb{S}, \mathbb{A}, T, R, \mathbb{O})$, where $\mathbb{S}$ is the state space; $\mathbb{A}$ is the action space; T is the state transition model; R is the reward function, and $\mathbb{O}$ is the observation space. Their specific definitions in the robot navigation problem are as follow:

3.1.1 State Space

The state space $\mathbb{S}$ consists of a set of state vectors $\mathbf{s}$ which represent the decision-making environment. In the robot navigation problem, the state vectors include the position of objects $\mathbf{s}_p$, the velocity of objects $\mathbf{s}_v$, the acceleration of objects $\mathbf{s}_a$, the intention of objects $\mathbf{s}_i$ and the shape of objects $\mathbf{s}_s$. Since the sensors of robots only perceived limited information with noise, these state vectors usually cannot be fully measured, and must be deduced from observations in the observation space. Hence, the state space can be written as a 5-tuple consisting of all of the existing state vectors, as shown in Eqn. 3.1

$$(\mathbf{s}_p^t, \mathbf{s}_v^t, \mathbf{s}_a^t, \mathbf{s}_i^t, \mathbf{s}_s^t) \in \mathbf{s}^t \tag{3.1}$$

3.1.2 Action Space

A set of actions of the agent represent the action space $\mathbb{A}$. In terms of the robot navigation problem, the feasible action space is related to the locomotion of the mobile platform. In this project, differential two wheeled drive is implemented. Thus, we can only control the linear velocity of the robot $\mathbf{a}_{linear}$ and the angular velocity $\mathbf{a}_{angular}$. Hence, the action space can be written as a 2-tuple consisting of the linear velocity and the angular velocity, as shown in Eqn. 3.2

$$(\mathbf{a}_{linear}^t, \mathbf{a}_{angular}^t) \in \mathbf{a}^t \tag{3.2}$$

3.1.3 State Transition Model

The state transition model T represents the probabilities of transitioning between different states. The agent takes an action $\mathbf{a}^t$ in the state $\mathbf{s}^t$ at time step t, which

causes the environment to transition to the new state $\mathbf{s}^{t+1}$ with probabilities $T(\mathbf{s}^{t+1}|\mathbf{s}^t,\mathbf{a}^t)$. For example, the movement of the robot at every step may randomly affect the pedestrian behavior in the environment.

3.1.4 Reward Function

To measure the agent's performance, the reward function R is introduced. After the transition to a new state is finished, the agent will receive a reward. The reward is generated by the reward function R which depends on the new state, the last action taken and the last state the robot was in (i.e. $R(\mathbf{s}_{t+1},\mathbf{a}_t,\mathbf{s}_t)$). The objective of the agent is to maximize the reward. Therefore, the design of the reward function significantly impacts the behavior that the robot finally learns. The goal of this project is to make it so that the robot can reach the final destination as quickly as possible while still avoiding collisions. The robot is punished for collisions with other obstacles, and rewarded for reaching its goal. Referring to [35], the reward function is designed as follow:

$$r^t = R(\mathbf{s}_{t+1},\mathbf{a}_t,\mathbf{s}_t) = \begin{cases} 20 & \text{if } \|\mathbf{s}_p^t - \mathbf{s}_i^t\| < 0.1 \\ -20 & \text{else if } \textbf{collision} \\ 2.5\cdot(\|\mathbf{s}_p^{t-1} - \mathbf{s}_i\| - \|\mathbf{s}_p^t - \mathbf{s}_i\|) & \text{otherwise.} \end{cases} \tag{3.3}$$

3.1.5 Observation Space

The observation space $\mathbb{O}$ is composed of a set of observation vectors $\mathbf{o}$. The scope of this observation space depends on the sensors of the agent. In this project, the robot is equipped with the following sensors: a 2D LiDAR, a color camera, a depth camera, and an inertial measurement unit (IMU). Hence, the observation vectors can be written as as $\mathbf{o}_{lidar}$, $\mathbf{o}_{color}$, $\mathbf{o}_{depth}$ and $\mathbf{o}_{IMU}$ respectively. Based of the these observations, the robot can infer the goal position $\mathbf{o}_{goal}$. Since these sensors cannot perceive perfectly, the observation model Ω is introduced to account for the observation uncertainty $\Omega(\mathbf{o}^{t+1}|\mathbf{s}^{t+1},\mathbf{a}^t)$. Hence, the mathematical respresentation of the observation space can be seen in Eqn. 3.4.

$$(\mathbf{o}_{lidar}^t, \mathbf{o}_{color}^t, \mathbf{o}_{depth}^t, \mathbf{o}_{imu}^t, \mathbf{o}_{goal}^t) \in \mathbf{o}^t \tag{3.4}$$

3.2 Deep Reinforcement Learning

With the rapid development of the Artificial Intelligence (AI), many POMDP problems have been solved by Deep Reinforcement Learning approaches [5], [36]. The optimal policy π^* can be defined according to the Bellman equation:

$$\pi^* = \arg\max_{\mathbf{a}^*} R(\mathbf{s}^(t+1),\mathbf{a}^t,\mathbf{s}^t) + \gamma \int_{\mathbf{s}^{t+1}} T(\mathbf{s}^{t+1}|\mathbf{s}^t,\mathbf{a}^t) V^*(\mathbf{s}^{t+1}) \mathrm{d}\mathbf{s}^{t+1} \tag{3.5}$$

where $\gamma \in [0,1]$ is the discount factor; $V(\mathbf{s})$ is the State-Value function, defined as the expected reward.

The optimal policy is a sum of the maximum of the reward function, and the discount factor multiplied by the the sum of, at time point $t+1$, the state transition model, multiplied by the State-Value function.

$$V(\mathbf{s}^t) = \mathbf{E}[\sum_{t'=t}^{\infty} \gamma^{t'-t} r^t] \tag{3.6}$$

The State-Value model at time step t can be rewritten as the sum to infinity of the discount factor at time step $t'-t$ minus the reward factor at time step t.

Based on these definitions, many RL algorithms can find the optimal policy via interaction with the environment. In next chapter, we present the training details for the robot navigation problem.

Chapter 4 Approach

In this chapter, we first introduce the visual perception method we implemented on our robot, which is used to capture the social information in the surrounding environment. Then, to integrate all of the relevant sensor information into the navigation module, we propose a sensor fusion module. Finally, we present the decision-making network used to achieve human-friendly navigation.

4.1 Visual Perception

To realize real-time object detection with limited computational resources, YOLO [15] is employed as our visual perception module. Compared to Faster R-CNN [6], YOLO has faster inference time, but comparable detection accuracy [15]. Due to our limited computational resources, faster inference time is crucial towards being able to detect humans in real-time.

The YOLO family of object detectors are able to achieve faster inference time due to their architecture. Unlike other methods, which conduct object recognition in multiple steps, YOLO treats object recognition as a single regression problem. YOLO splits the inputted image into an $S \times S$ grid, where each grid cell predicts only one object [15]. The cell responsible for predicting an object is the cell which contains the center of the object. Each grid cell predicts B bounding boxes, each which contain a box confidence score, as well as C class probabilities, one per class. The box confidence score reflects how likely the bounding box actually contains an object, while the class probability represents how likely an object belongs to a certain class. YOLOv3, the version used in this paper, uses Darknet-53, a 53 layer convolutional network, as its backbone feature extractor [37]. Additional convolutional layers are added to the network in order to make predictions at three scales, increasing YOLOv3's ability to predict objects at images of different scale. The last layer predicts the bounding boxes, box confidence score, as well as the class probabilities [37]. An example of the output from YOLOv3 can be seen in Fig. 4-1.

To further improve the inference time, the NVIDIA TensorRT platform is introduced [38]. The TensorRT first optimizes the trained network model by merging the repeated operations in it. Then, TensorRT can reduce the inference precision of the network (e.g. half precision floating format FP16) to achieve lower latency without losing too much accuracy. Finally, the optimized inference module is built to capture the social information in the environment.

Figure 4-1: A demonstration of the output from the YOLOv3 object detector.

4.2 Sensor Fusion

To generate a laser scan with social-awareness, a sensor fusion module is proposed to integrate visual information into the raw sensor data of the 2D LiDAR. This is because there are two limitations for the raw data. Firstly, the 2D point cloud information on simplifies the description of the obstacles in the environment, making it that the robot cannot fully realize the collision detection, as it exists in the three-dimensional space. Secondly, this point cloud data only contains the geometric information of the environment and ignores the semantic information, which we have already established plays a crucial role in human-friendly navigation. Therefore, we introduce a sensor-fusion module to solve these problems. The structure of the sensor fusion module can be divided into two parts:

First, the scan returned by the 2D laser scanner is fused together with the point cloud information returned by the depth camera, in order to detect obstacles that the 2D LiDAR may not have detected. Points from the y axis of the point cloud are filtered in order to detect obstacles that are lower than the robot, but may not have been detected by the LiDAR. We then label the corresponding location of these obstacles on the 2D LiDAR scan. Hence, the collision scan can be represented as the output of the depth fusion function, with inputs as the 2D LiDAR and the depth information, as seen in Eqn. 4.1.

$$\mathbf{o}^t_{collison} = \mathscr{F}_{depth}(\mathbf{o}^t_{lidar}, \mathbf{o}^t_{depth}) \tag{4.1}$$

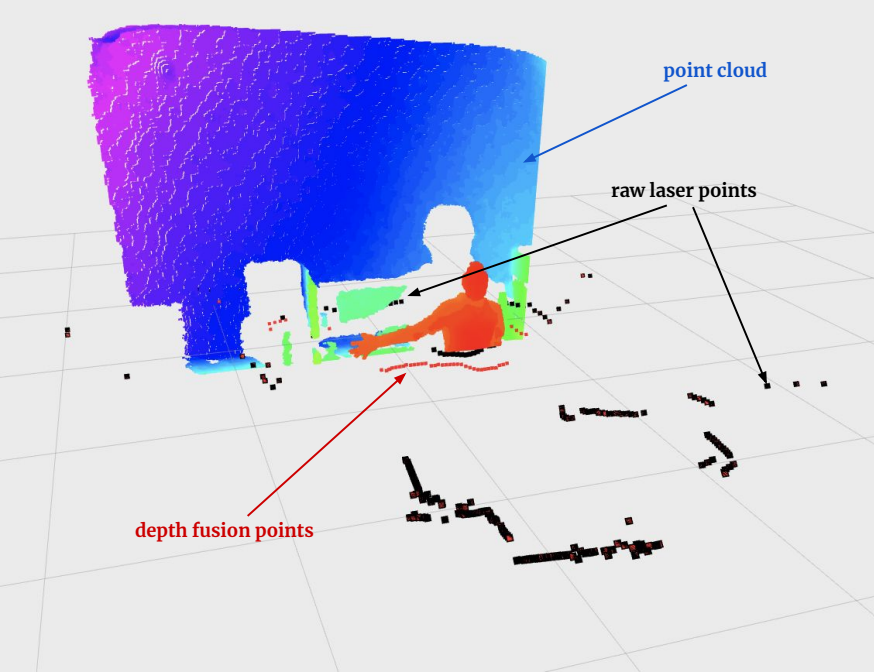
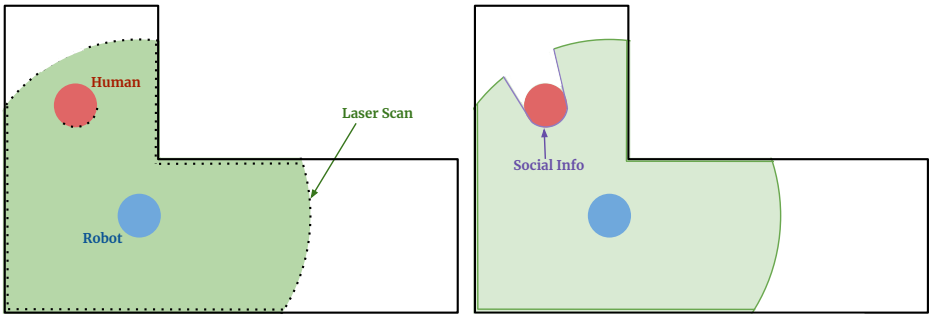


Figure 4-2: A demonstration of the output from the depth fusion. Points in black represent the raw 2D LiDAR data, while the points in red represent the poitns added to the LiDAR data after the depth fusion.

Second, as the bounding boxes of objects are drawn by the visual perception module, the coordinates of the humans in the image can be found. By calibration of the image coordinates and LiDAR coordinates, we can label the points in the laser scan which belonging to the humans. We propose that robots should not only avoid collision with humans, but also should not intrude the social space of humans. Therefore, to keep a certain social distance from human, we need to integrate the social information into the raw sensor data to generate a human-aware laser scan which represents the social feasible space. The social feasible space can be represented as the output of the social function, with inputs of the collisions scan and the RGB information.

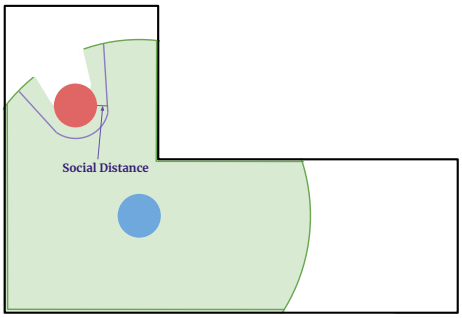
$$\mathbf{o}^t_{social} = \mathscr{F}_{social}(\mathbf{o}^t_{collision}, \mathbf{o}^t_{rgb}) \tag{4.2}$$

The social feasible space computation can be divided into four steps. Because the point cloud coordinates on the 2D plane are based on a robot-centered polar coordinate system, we first transform the points on the polar coordinate system to the Cartesian coordinate system. Then, we connect these points into a polygon and offset the line segments which are labeled as human by the predetermined social distance. Thus, we can obtain the polygon of the social feasible space. Lastly, we construct rays from the robot with different angles, and compute the intersection of the smallest distance for the polygon. Through this, we obtain the human-aware laser scan, as seen in Fig. 4-4.

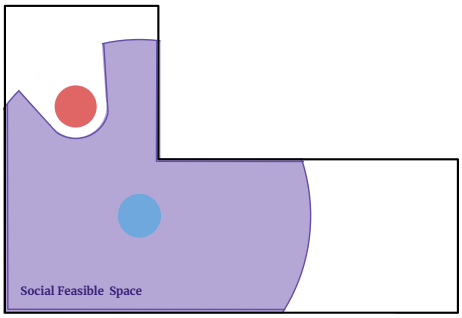


(a) Step 1: Setup of the environment

(b) Step 2: Preliminary mapping



(c) Step 3: Social distance drawn



(d) Map modified

Figure 4-3: A diagram explaining the procedures taken in the sensor-fusion module. In step 1, a setup of the environment is depicted. In step 2, prelimary mapping is undertaken using the 2D LiDAR sensor. In step 3, the social distance is calculated by offsetting the line segments representing the edges of the human. In step 4, the sensor-fusion module modifies the map generated in step 2 according to the boundaries draw in step 3.

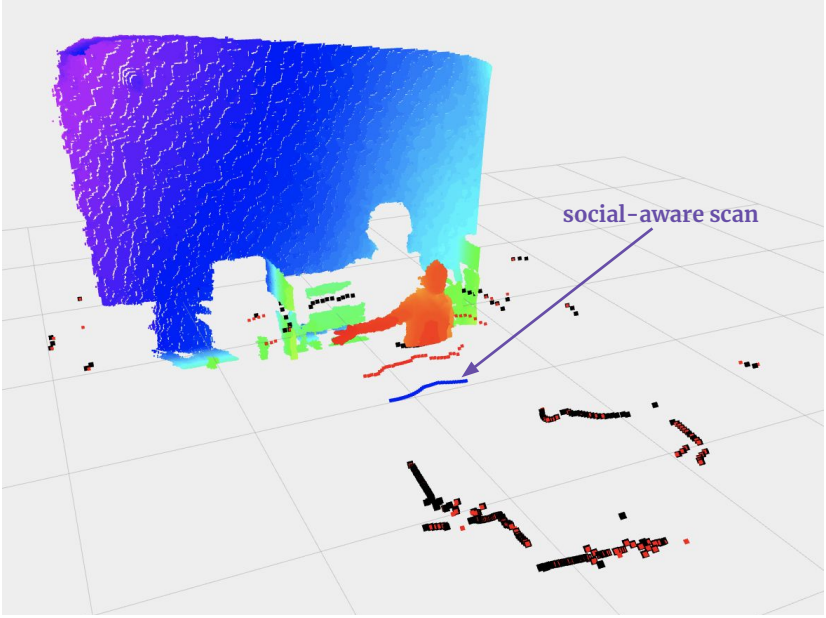


Figure 4-4: An example of the output of the sensor-fusion module. Note the blue line, demonstrating the offset edges belonging to the human in frame.

Finally, the human-aware laser scan is then fed into the navigation algorithm, thus allowing the navigation algorithm to achieve human-friendly navigation.

4.3 Human-Friendly Navigation

To achieve harmonious HRI, the robot should have the ability to adapt to the highly dynamic movement of humans. Thus, we introduce the navigational network trained by reinforcement learning.

Referring to previous related work [35], we build a end-to-end reinforcement learning network with the 2D LiDAR scan as input. The network deploys a 1D convolutional neural network to extract the information of laser scan, and the output of the network is the linear velocity and angular velocity, as defined in Chapter 3. The training environment is based on the multi-agent simulator Stage [39].

Differing from [35], which used multiple training scenarios to learn a collision avoidance policy for multi-robot systems, we introduce a crowd simulator to simulate the pedestrian behavior in our simulation [11], [40], in order to ensure the robot can learn how to interact with humans instead of other robots. The training scenario includes 50 robots and 50 people, and randomly generating their start points and goals in a 25m $\times$ 25m free space. Since it is computationally expensive to generate a human-aware scan in simulations, the raw laser scan is used to train the network how to navigate in the local feasible space. In the real world testing, we feed the social feasible space into the network to conduct human-aware navigation.

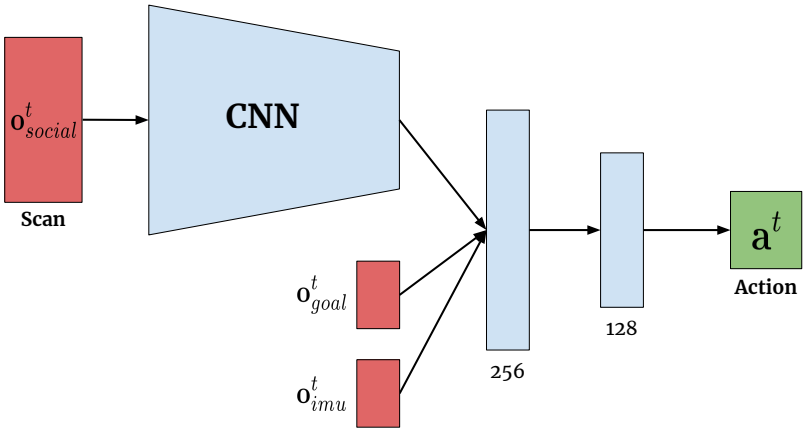


Figure 4-5: The navigation network architecture. A five hidden layers CNN module is proposed to extract the feature in the laser scan $\mathbf{o}^t_{social}$ effectively. Then, the other relative information $\mathbf{o}^t_{imu}$ and $\mathbf{o}^t_{goal}$ is combined with the feature of the scan. Finally, the action is computed.

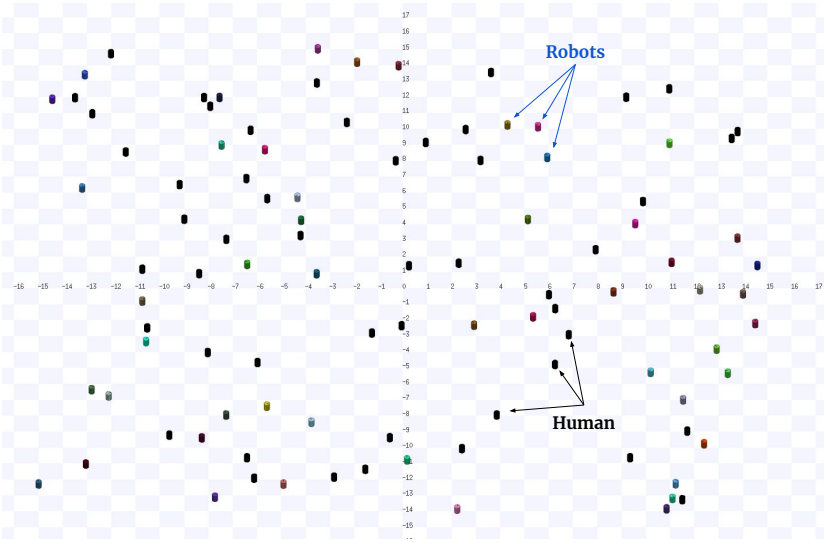


Figure 4-6: A demonstration of the simulation environment.

For the training algorithm, we implement the state-of-the-art RL algorithm Soft Actor-Critic (SAC) [41] in order to accelerate the training process. The training pseudo code is shown in Algorithm 1. The whole training process costs about 1.5 hours for 300 episodes, and the cumulative reward curve is shown in Fig. 4-7.

Algorithm 1 Soft Actor-Critic for navigation in crowds

Initialize policy network π_{θ_t}, Q-value function $Q_{\phi_t}(\mathbf{o}_t, \mathbf{a}_t)$ and replay buffer $\mathbb{B}$; set up a multi-robot simulator $\mathbb{E}$.
for iteration $= 1, 2, ...,$ **do**
  // Collect data
  for timestep $t = 1$ to T **do**
    for robot $i = 1$ to N **do**
      // Get an action by π_θ
      $\mathbf{a}_i^t = \pi_{\theta_t}(\mathbf{o}_i^t)$
      // Execute action in the environment
      $(r_i^t, \mathbf{o}_i^{t+1}) \leftarrow \mathbb{E}.step(\mathbf{a}_i^t)$
      // Add data to the replay buffer
      $\mathbb{B}$.add$(\mathbf{o}_i^t, \mathbf{a}_i^t, r_i^t, \mathbf{o}_i^{t+1})$
    end for
  end for
  // Update network
  if $\mathbb{B}$.size$() > B$ **then**
    // Sample a branch of the training data item
    $\tau \leftarrow \mathbb{B}$.sample$()$
    // Apply SAC learning rule to compute the loss [41]
    $l_t =$ ComputeLoss$(\tau; \theta_t, \phi_t)$
    // Update network parameters by the Adams optimizer [42]
    $\theta_{t+1}, \phi_{t+1} =$ UpdateParameters$(l_t; \theta_t, \phi_t)$
  end if
end for

Eventually, based off the simulation, the navigation network learns the optimal policy. In order to enable the robot to navigate with human-awareness, the social scan, extracted by the visual perception module and the sensor-fusion module is applied in the real world experiments.

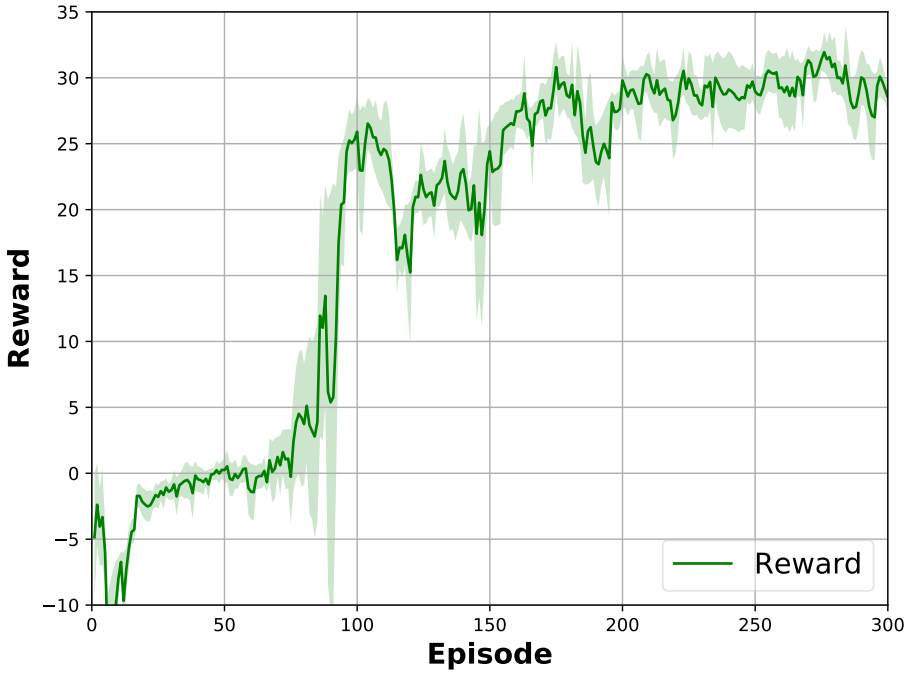


Figure 4-7: Cumulative reward curve of the training process.

Chapter 5 Experiments

In this chapter, we first demonstrate the experiment setup, including the robot hardware, software and the experiment scenarios. Finally, we present the experiment results.

5.1 Experiment Setup

5.1.1 Hardware

For this project, two separate mini computers (Intel Celeron Processor J1900 [43] and Jetson TX1 Module [44]) are used to implement our algorithm. In terms of the sensors, as we stated in Chapter 3.1.5, a 2D LiDAR, a color camera with a wide angle lens, a depth camera [45] and the IMU module is used. All of these are mounted on the mobile platform Spark-T [46]. A diagram of the hardware used can be seen in Fig. 5-1.

To achieve faster runtime of the program, two different computers are used to account for different computation. The main computer has a Intel Celeron Processor J1900 CPU with four cores x86-64 architecture[43], and accounts for the algorithms that can not be parallel computed (e.g. the sensor-fusion algorithm). The Jetson TX1 Module has a 256 CUDA cores based on ARM architecture [44], and accounts for the programs that support CUDA parallel acceleration (e.g. the computer vision module). Images are first processed with YOLO in the GPU board, allowing for much faster processing of images. Once human labelling is completed in the Jetson TX1, the processed information is then sent to the main computer via Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The main computer uses the sensor-fusion module as described in Section 4.2 to create a human-aware laser scan. This is used by the navigation network as described in Section 4.3 to conduct navigation, and instructs the wheels to move accordingly. The usage of two different computers, one CPU heavy and another GPU heavy, allows for faster runtime of the programs, thus enabling the robot react to its surroundings in real time.

As for sensors, a 2D LiDAR, a color camera with a wide angle lens, and a depth camera have been chosen. The laser scanner is mounted at the bottom of the robot, and returns a basic map of all sides of the robot. The depth camera is mounted at the middle of the robot, and returns a depth information of the region in front of the robot. This depth information is processed through the sensor-fusion module, and is used to identity obstacles that the 2D LiDAR may have missed. Object detection for obstacles in front of the robot is conducted through inclusion of the RGB information from the wide angle camera. A wide angle lens has been used in order to allow the robot to continue to detect people while it is

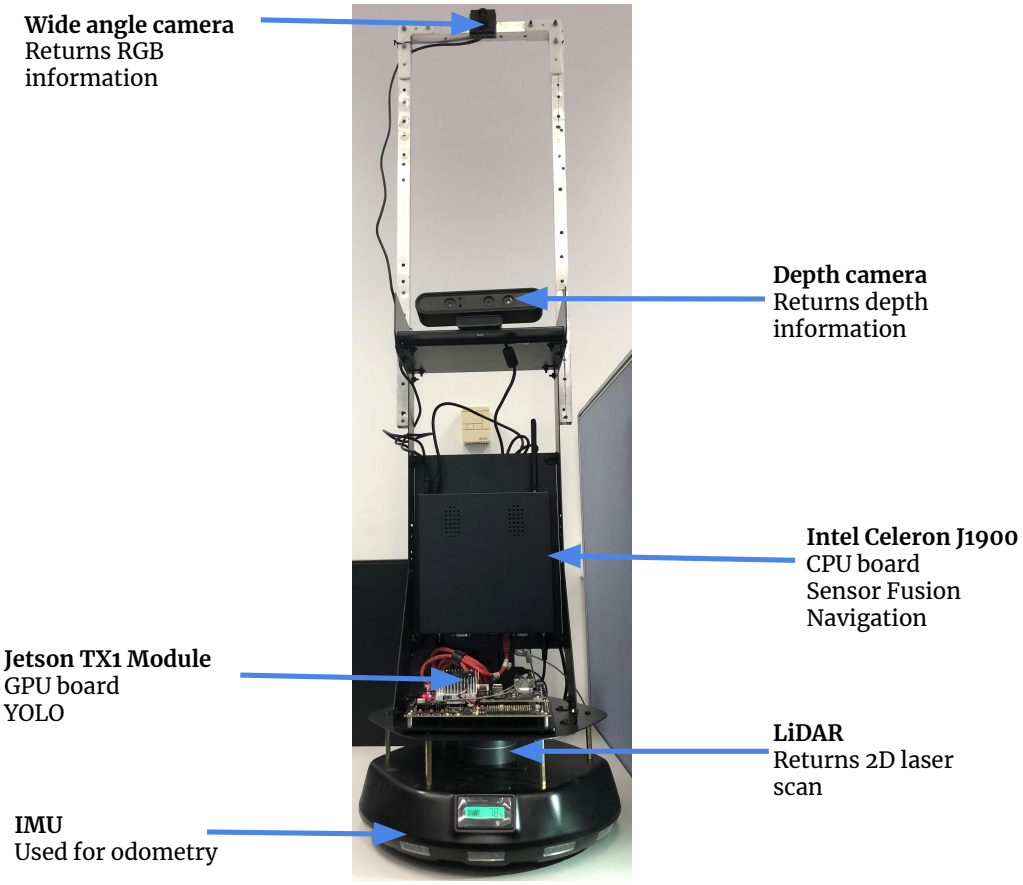


Figure 5-1: A diagram of the hardware used in this project. The 2D LiDAR is situated on the the mobile platform at the bottom of the robot. While mounted at the middle is the depth camera, and mounted at the top is the color camera with a wide angle lens. The mobile platform has two wheels located at its base. This allows for the robot to move forwards and backwards, as well as turn.

passing them. This allows the robot to perform the correct action based on the obstacles ahead of it, ensuring harmonious HRI.

5.1.2 Software

The project was created on Robotic Operating System (ROS). ROS acts as a middleware between robotic software and robotic hardware [47]. It provides services very similar to operating systems, such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [47]. ROS lets different hardware components communicate with each other, allowing one to create multi-machine programs. Sensor driven programs benefit massively from ROS, as its function as a middleware allows for ease of communication between different components. As The inclusion of the node structure allows ROS programs to be extremely modular and scalable. This project is sensor driven due to the need of the robot to be able respond to its environment in realtime, hence the usage of ROS in order to easily create a multi-machine program.

ROS was used to achieve a multi-machine system as it greatly increases the runtime of the program and makes the program more flexible. Usage of ROS image processing tasks to be allocated to the GPU, while delegating other tasks to the CPU by establishing a connection between the two computers. Based on ROS, the software framework consists of three main levels: the sensor level, computational level, and actuator level. A diagram of the overall framework can be seen in Fig. 5-2.

The sensor level is composed of four different components: a 2D LiDAR, a color camera with a wide angle lens, a depth camera, and an IMU. The LiDAR returns a laser scan, the color camera returns RGB, the depth camera returns a point cloud with depth information, and the IMU returns the velocity and angles of the mobile platform. All of these sensor messages are subscribed by the CPU, which is located in the computation level.

The computational level is composed of two computers: the Intel Celeron J1900 CPU board and the Jetson TX1 GPU board. Both of these computers are installed with the Ubuntu system. The CPU subscribes to all of the sensor messages, then publishes the RGB image, which is subscribed by the GPU. The GPU conducts object detection on the image through the YOLO object recognition system. After that, the x-coordinates of the bounding boxes containing humans are extracted and published. This is subscribed by the CPU, which contains the sensor-fusion module and the navigation module. The sensor-fusion module fuses together data from the depth camera and the color camera with a wide angle lens to label obstacles missed by the 2D LiDAR, as well the location of humans in frame. The points containing humans are offset, making the robot perceive the humans as larger obstacles. This modified socially-aware scan is then published and subscribed by the navigation module, and is then inputted into the navigation network as described in section 4.3. Through usage of the

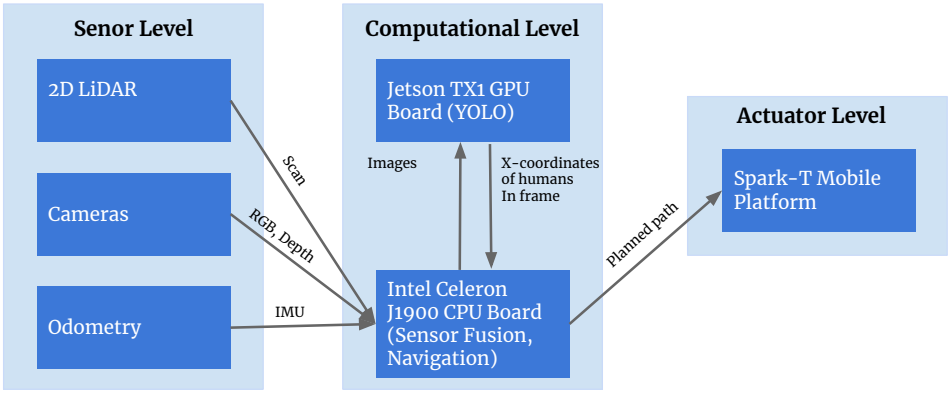


Figure 5-2: A diagram depicting the framework of the software for this project. The software contains three layers: the sensor layer, the computational layer, and the actuator layer.

Question	Answer
Do you think this robot disturbed you? (1 - 5, 1 = not at all, 5 = a lot)	
Do you think the distance between the robot and you is appropriate? (1 - 5, 1 = too close, 5 = acceptable)	
Please estimate the closest distance between you and the robot. (10 CM, 20 CM, 30 CM, 40 CM, 50 CM)	
Please give a final score to the navigation algorithm (considering time and speed in which the robot reached its goal, and whether or not it disturbed you in this process) (1 - 5, 1 = bad, 5 = good)	

Table 5-1: User Survey for the Multi-Human Testing

pre-trained navigation network and the human-aware laser scan, the robot is able to conduct human-friendly navigation. The results of the navigation module are then sent to the mobile platform, which is located in the actuator level.

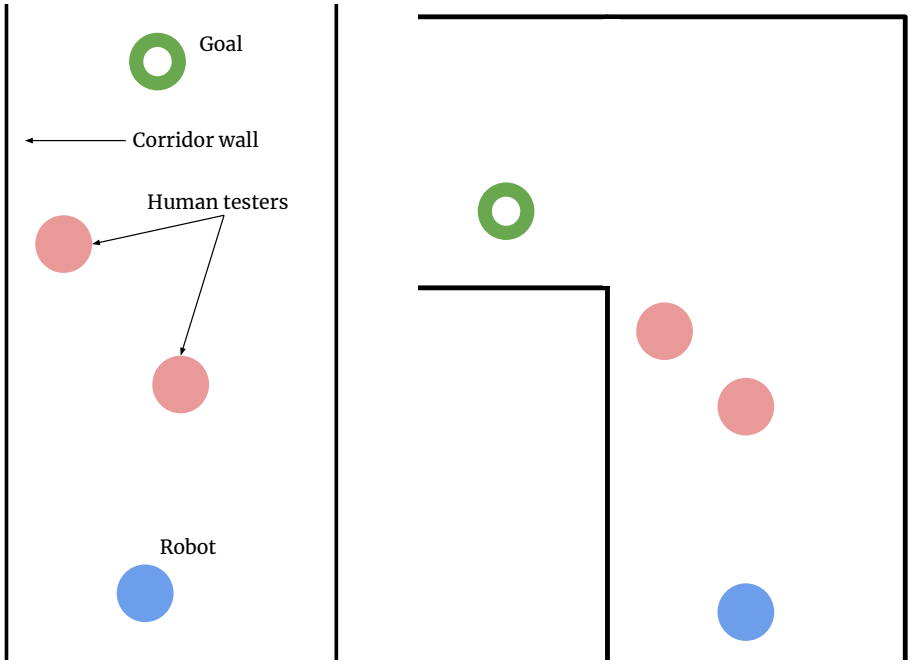
In the actuator level, the mobile platform receives commands from the navigation module and executes the commands, navigating in a human-friendly way.

5.2 Experiment Results

5.2.1 Single Human Interaction Experiment

5.2.1.1 Setup

For the single human tests, the experiment setup had either a human obstacle or a non-human obstacle stand in the path of the robot, while the robot tried to navigate to a goal behind them.



(a) Scenario 1 (b) Scenario 2

Figure 5-3: Diagrams depicting the setup of the two testing scenarios.

(a) Demonstration of the social distance of human obstacles and the robot.

(b) Demonstration of the social distance of non-human obstacles.

Figure 5-4: Images from the single human testing. The robot displays a much greater social distance when faced with a human obstacle, compared to a non-human obstacle.

5.2.1.2 Results

In this scenario, the robot demonstrated an marked increase in social distance when avoiding humans, versus avoiding non-human obstacles, as shown in Fig. 5-4. As conditions were kept the same in both experiments, this demonstrates that our robot is reacting to its surroundings in real-time, and is able to respect the social distance of humans while navigating in single human scenarios. The baseline algorithm, not being able to discern whether not an obstacle was human or not, simply concerns itself with finding the shortest path, which is to stick as close as possible to the human. However, our method, being able to recognise humans and account for their social distance, and bypasses the human. The marked increase in the avoidance distance indicates the success of the human-friendly navigation method in conducting human-friendly navigation with regards to single human scenarios.

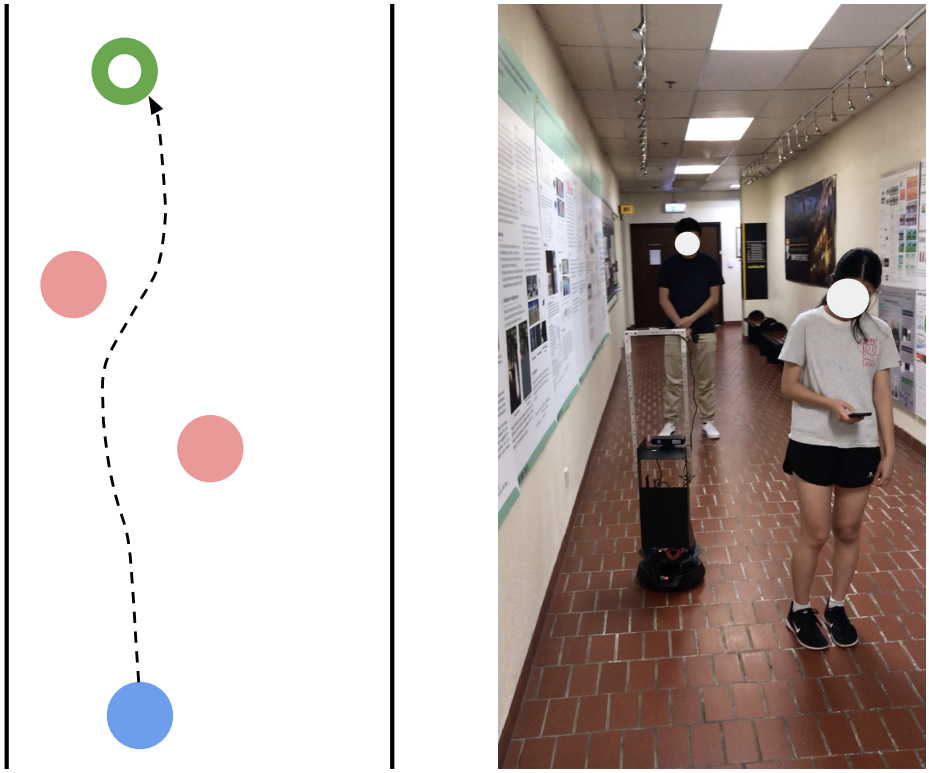
5.2.2 Mutli-Human Experiments

5.2.2.1 Setup

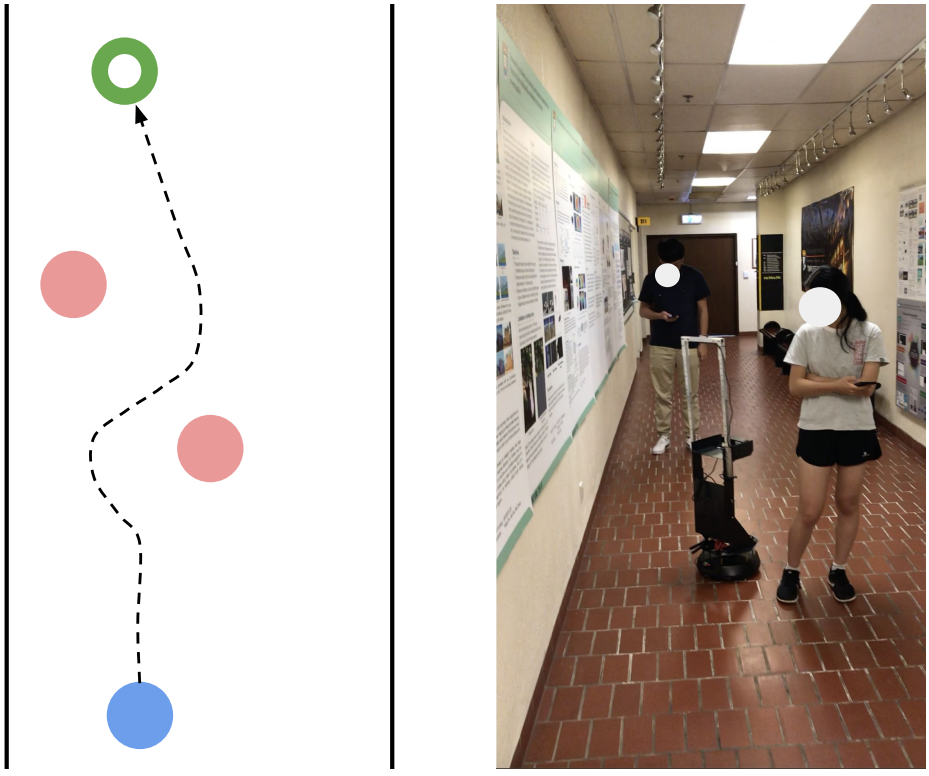
For multi-human tests, all testing took place in a corridor. All human volunteers were divided into group of two persons, each group was asked to stand in the robot's path in two different scenarios, while the robot would navigate to its goal and avoid collision with human and other obstacles. A diagram of the two scenarios used for multi-human testing can be seen in Fig. 5-3. Each volunteer tested the same scenario twice, one with our human-friendly navigation algorithm, another with a baseline, non human-friendly navigation algorithm. The volunteers were not told which algorithm was being tested. After testing, the volunteers were asked to compare the two by filling out a survey. The survey questions given to the multi-human testing participants can be seen in Tab 5-1.

5.2.2.2 Results

In scenario 1, the baseline navigation algorithm had a tendency to try and squeeze between the participants, while our algorithm was able to respect the social distance, and bypass the participants, as shown in Fig. 5-5. In all trials, the environment was kept the same, and only the algorithm used changed, demonstrating that the baseline algorithm was not human-friendly, while our approach was. This is because the baseline method does not account for humans and instead treats all obstacles as equal, thus it only accounted for the shortest path, leading it to only determine whether not the distance between the two people would be great enough for it to pass, which it determined it was. In contrast, our method account for the social-distance of the humans in the tests, and thus determined that there was not enough space to pass through the two test subjects, and bypassed them. The affects of this can be seen in the testing results (as shown in Tab.5-2). There was a statistically significant ($p < 0.05$)

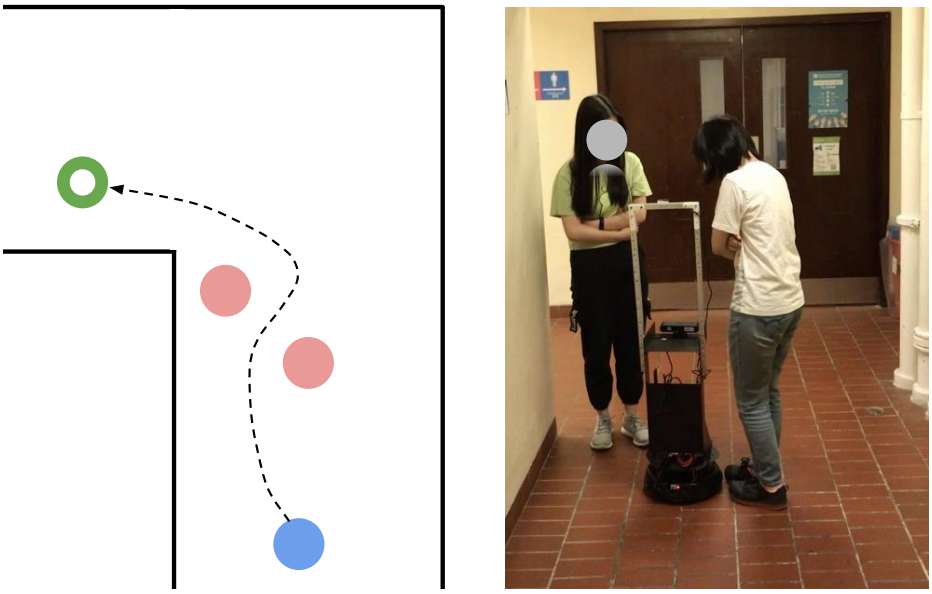


(a) An example of the path taken by the baseline method in scenario 1.

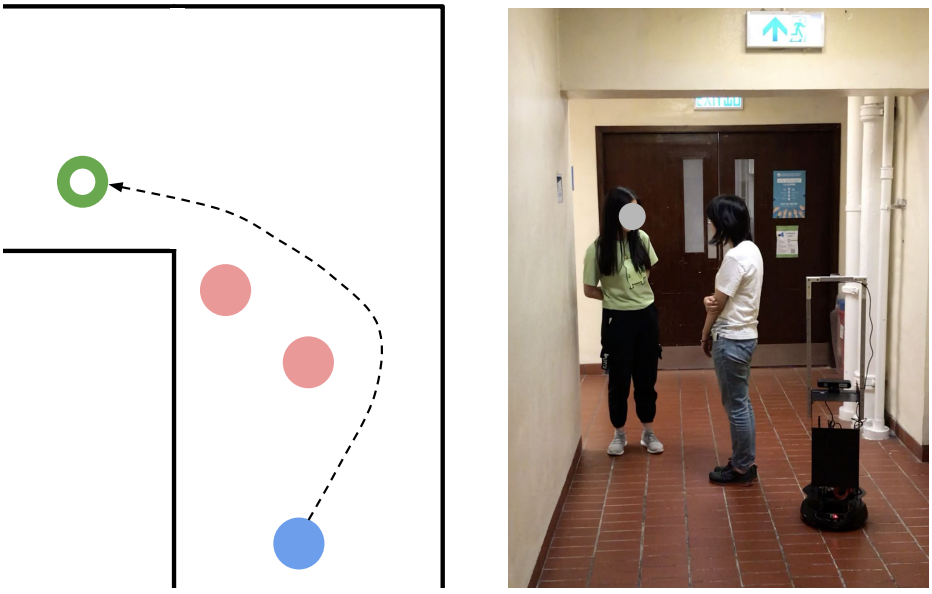


(b) An example of the path taken by our method in scenario 1.

Figure 5-5: A comparison of paths taken by the baseline method and our method in scenario 1.



(a) An example of the path taken by the baseline method in scenario 2.



(b) An example of the path taken by our method in scenario 2.

Figure 5-6: A comparison of paths taken by the baseline method and our method in scenario 2.

decrease in the disturbance level with the baseline navigation algorithm and our human-friendly navigation algorithm, indicating that our method is indeed a more human-friendly approach. This is again reflected in the higher mean and median performance score for our approach, which helps to further reconfirm this point. Perceived distance had a greater mean and median for our approach, indicating that users were able to discern a visible different between the two approaches.

In scenario 2, the baseline method caused the robot to come very close to the participants, but our method was able to respect and maintain the social distance. This is again because the baseline method does not account for a human's social-distance, hence the baseline method simply takes the shortest route possible, which requires the robot to come as close as possible to the participants. With our human-friendly approach, the robot is able to differentiate the humans in frame, and thus take a much more human-friendly approach by avoiding the humans. However, due to the setup of the testing, this effect was less prominent than scenario 1. As shown in Fig. 5-6, the robot with our method rapidly swerves right after passing the front participant, as it fails to sense the front participant anymore. Instead, it swerves to the right in order to keep social distance with the back participant. Hence, the front participant gave more negative feedback, while the back participant gave more positive feedback. This is reflected in the statistical difference (as shown in Tab.5-2), which is less significant ($p > 0.05$) but still visible. This is shown in the decreased mean and median disturbance level, coupled with the increased mean and median performance score. This further supports that our method is more human-friendly that standard navigation methods. Perceived distance again had a greater mean and median for our approach, indicating that users were still able to discern a visible different between the two approaches.

In summary, our results demonstrate that our method is able to successfully navigate autonomously in both single human scenarios and multi-human scenarios, and is rated more human-friendly in both scenarios. The success of our method in multiple tests and multiple scenarios suggests that our method is able to cope with a wide variety of real world scenarios, and is not just limited to the ones presented here.

Table 5-2: Descriptive Statistics of Multi-Human Testing Results

Scenario 1							
	Baseline			Human-friendly navigation			p-value
	M	SD	Mdn	M	SD	Mdn	
Disturbance level (H1)	6.08	2.25	7	2.42	1.85	2	0.0020
Performance score (H2)	3.08	1.38	3.5	4	1.08	4	0.1059
Perceived distance (cm)	20.83	13.20	15	32.5	11.64	30	0.0306
Scenario 2							
	Baseline			Human-friendly navigation			p-value
Disturbance level (H1)	5.09	2.64	5	3.82	2.21	3	0.2975
Performance score (H2)	3.36	1.37	4	3.54	1.07	3	0.8123
Perceived distance (cm)	19.10	10.83	10	23.55	9.9	20	0.2307

Chapter 6 Conclusion

To conclude, this paper presents a human-friendly autonomous robot that conducts navigation based on deep reinforcement learned collision avoidance. In order to achieve this task, object detection was performed on camera imagery to detect humans, which was then used to label the corresponding coordinates of the humans in the 2D LiDAR scan. Depth information was also used to further account for any obstacles that the 2D LiDAR may have missed. This processed scan was then fed into the navigation algorithm, allowing for the algorithm to distinguish between human and non-human obstacles, respecting the human's social distance. The method was tested on a variety of scenarios multiple times, including single human and multi-human scenarios. In single human tests, the robot demonstrated a marked difference in its approach towards avoiding humans and non-human obstacles. In multi-human scenarios, robot again demonstrated a discernible difference in its approach towards humans and non-human obstacles. The user surveys further reconfirm that our approach is more human-friendly than traditional navigation methods, as demonstrated by lower reported disturbance levels and higher performance scores for our method.

References

[1] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, “Intention-aware online pomdp planning for autonomous driving in a crowd,” in *2015 ieee international conference on robotics and automation (icra)*, IEEE, 2015, pp. 454–460.

[2] J. Wirtz, P. G. Patterson, W. H. Kunz, T. Gruber, V. N. Lu, S. Paluch, and A. Martins, “Brave new world: Service robots in the frontline,” *Journal of Service Management*, vol. 29, no. 5, pp. 907–931, 2018.

[3] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[4] J. Van den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 1928–1935.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.

[6] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[7] A. Inc., *Apple launches iphone 4s, ios 5 & icloud*. [Online]. Available: `https://www.apple.com/ie/newsroom/2011/10/04Apple-Launches-iPhone-4S-iOS-5-iCloud/`.

[8] N. Chokshi, “Facial recognition’s many controversies, from stadium surveillance to racist software,” *The New York Times*, May 19, 2019. [Online]. Available: `https://www.nytimes.com/2019/05/15/business/facial-recognition-software-controversy.html` (visited on 09/01/2019).

[9] O. K. Bruno Siciliano, *Springer Handbook of Robotics*. Springer, 2008.

[10] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[11] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, “Reciprocal n-body collision avoidance,” in *Robotics research*, Springer, 2011, pp. 3–19.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[14] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *CoRR*, vol. abs/1311.2524, 2013. arXiv: 1311.2524. [Online]. Available: http://arxiv.org/abs/1311.2524.

[15] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *CoRR*, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[16] G. N. DeSouza and A. C. Kak, “Vision for mobile robot navigation: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 2, pp. 237–267, 2002.

[17] R. Chatila and J.-P. Laumond, “Position referencing and consistent world modeling for mobile robots,” in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, IEEE, vol. 2, 1985, pp. 138–145.

[18] S. T. A. W. DieterFox and T. D. T. M. TimoSchmidt, “Map learning and high-speed navigation in rhino,” 1998.

[19] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, “Minerva: A second-generation museum tour-guide robot,” vol. 3, Feb. 1999, 1999–2005 vol.3, ISBN: 0-7803-5180-0. DOI: 10.1109/ROBOT.1999.770401.

[20] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee, and D. Manocha, “Porca: Modeling and planning for autonomous driving among many pedestrians,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3418–3425, 2018.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[22] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, “Target-driven visual navigation in indoor scenes using deep reinforcement learning,” in *ICRA*, 2017, pp. 3357–3364.

[23] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, “Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–8.

[24] S. Thrun and Y. Liu, “Multi-robot slam with sparse extended information filers,” in *Robotics Research. The Eleventh International Symposium*, Springer, 2005, pp. 254–266.

[25] S. Thrun, W. Burgard, and D. Fox, “A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping,” in *ICRA*, vol. 1, 2000, pp. 321–328.

[26] Y. F. Chen, M. Everett, M. Liu, and J. P. How, “Socially aware motion planning with deep reinforcement learning,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 1343–1350.

[27] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.

[28] Y. F. Chen, M. Liu, M. Everett, and J. P. How, “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning,” in *ICRA*, 2017, pp. 285–292.

[29] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in *Proc. AAAI*, 2008, pp. 1433–1438.

[30] R. Alami, I. Belousov, S. Fleury, M. Herrb, F. Ingrand, J. Minguez, and B. Morisset, “Diligent: Towards a human-friendly navigation system,” in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, vol. 1, Oct. 2000, 21–26 vol.1. DOI: 10.1109/IROS.2000.894576.

[31] P. Henry, C. Vollmer, B. Ferris, and D. Fox, *Learning to navigate through crowded environments*, May 2010. DOI: 10.1109/ROBOT.2010.5509772.

[32] J. Guzzi, A. Giusti, L. M. Gambardella, G. Theraulaz, and G. A. Di Caro, “Human-friendly robot navigation in dynamic environments,” in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 423–430.

[33] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R. Siegwart, “Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 2096–2101. DOI: 10.1109/IROS.2016.7759329.

[34] B. Kim and J. Pineau, “Socially adaptive path planning in human environments using inverse reinforcement learning,” *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, 2016.

[35] T. Fan, X. Cheng, J. Pan, D. Monacha, and R. Yang, “Crowdmove: Autonomous mapless navigation in crowded scenarios,” *arXiv preprint arXiv:1807.07870*, 2018.

[36] OpenAI, *Openai five*, https://blog.openai.com/openai-five/, 2018.

[37] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[38] NVIDIA, *Nvidia tensorrt: A programmable inference accelerator*. [Online]. Available: `https://developer.nvidia.com/tensorrt`.

[39] R. Vaughan, “Massively multi-robot simulation in stage,” *Swarm intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[40] S. Curtis, A. Best, and D. Manocha, “Menge: A modular framework for simulating crowd movement,” *Collective Dynamics*, vol. 1, pp. 1–40, 2016.

[41] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” *arXiv preprint arXiv:1801.01290*, 2018.

[42] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[43] Intel, *Intel® celeron® processor j1900*. [Online]. Available: `https://ark.intel.com/content/www/us/en/ark/products/78867/intel-celeron-processor-j1900-2m-cache-up-to-2-42-ghz.html`.

[44] NVIDIA, *Jetson tx1*. [Online]. Available: `https://developer.nvidia.com/embedded/jetson-tx1`.

[45] O. Astra, *Astra pro*. [Online]. Available: `https://orbbec3d.com/product-astra-pro/`.

[46] NXROBO. [Online]. Available: `https://www.nxrobo.com/product_product/cate/1`.

[47] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: An open-source robot operating system,” in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.