DESIGN OF A SOCIAL ROBOT AND SAFE SOCIAL NAVIGATION WITH DEEP REINFORCEMENT LEARNING

by

Kemal Bektaş

B.S., Mechanical Engineering, Boğaziçi University, 2018

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my supervisor, Prof. H.Işıl Bozma for all of her support, assistance, and encouragement during the thesis.

I would like to thank Assoc. Prof. Emre Uğur for his support during the thesis and accepting to be in my jury. I also would like to thank Prof. Erhan Öztop for accepting to be in my thesis jury. I also would like to express my gratitude to academicians of Boğaziçi University that make a stand for academic freedom for more than a year.

Being a part of the Intelligent System Laboratory did not only make me a robotics engineer, but also broadened my perspective about life. So I would like to express my thanks and appreciation to my colleagues Meriç Durukan, Serhat İşcan, Doğan Patar and Kadir Türksoy. They have always supported me and made my time in ISL enjoyable.

Finally, I would like to use this opportunity to express my profound and sincere gratitude to my family and all of my friends for their encouragement and endless support.

This study has been supported in part by TUBITAK EEEAG-118E857 and BAP 19A02M5 projects.

ABSTRACT

DESIGN OF A SOCIAL ROBOT AND SAFE SOCIAL NAVIGATION WITH DEEP REINFORCEMENT LEARNING

This thesis is concerned with the design and development of a social robot that can navigate around in a socially compliant manner. The importance of this problem is due to the growing demand of using robots in human-populated environments. In this thesis, this problem is addressed in two concurrent parts. The first part has focused on the physical design and development of a social robot - named as SempRob. SempRob is aimed to have a sympathetic appearance while also having a design in which its visual sensors are located appropriately for environmental sensing. In the second part, the social navigation capability of the social robot is developed. First, a novel navigation method referred to as artificial potential function with reinforcement learning (APF-RL) method. In addition, an ellipse-based representation of obstacles is developed for efficient obstacle representation. Furthermore, environmental complexity measures are defined in order to ensure that learning scenarios incorporate a range of maneuvering difficulties. Both simulation and experimental results with SempRob demonstrate that APF-RL method enables the robot to move safely and efficiently in complex environments. Following, APF-RL method is extended to Social APF-RL method so that the robot additionally respects the comfort zones of the humans while navigating. This requires the robot to detect the humans in its surroundings and to track them spatially. A deep learning based human detection algorithm is combined with a Kalman filter for this purpose. Finally, Social APF-RL method is modified to be applicable in human following as well. All the proposed methods are tested on the developed robot successfully.

ÖZET

SOSYAL ROBOT TASARIMI VE DERİN PEKİŞTİRMELİ ÖĞRENME İLE GÜVENLİ VE SOSYAL HAREKET PLANLAMASI

Bu tezde, sosyal olarak uyumlu bir şekilde hareket edebilen bir sosyal robotun tasarımı ve geliştirilmesi amaçlanmıştır. Günlük hayatta ve insanların bulunduğu ortamlarda robotların gittikçe daha fazla kullanılmaya başlanması bu problemi daha önemli hale getirmektedir. Bu tezde, bu problem iki ayrı bölümde ele alınmıştır. İlk bölümde SempRob adı verilen robotun tasarımına ve gerçeklenmesine odaklanılmıştır. Robot tasarlanırken, görünüşünün insanlara sempatik gelecek şekilde olmasına ve sensör lerin en uygun şekilde konumlandırılmasına dikkat edilmiştir. İkinci bölümde, robotun hareket planlama algoritmalarının geliştirilmesine odaklanılmıştır. İlk olarak, robotun karmaşık ortamlarda güvenli ve etkili hareketini sağlayan pekiştirmeli öğrenmeli yapay potansiyel fonksiyonlar (APF-RL) yöntemi önerilmiştir. Bunu sağlamak için elips tabanlı yeni bir engel modelleme yöntemi de geliştirilmiştir. Ayrıca, öğrenme senaryolarının tüm karmaşık ortamları kapsaması için yeni karma şıklık seviyesi metrikleri tanımlanmıştır. Hareketin modelinin eğitimi önce simülasyonda yapılmış sonrasında fiziksel robota aktarılmıştır. Sonrasında, APF-RL yöntemi modifiye edilerek sosyal navigasyon yöntemi olan sosyal APF-RL geliştirilmiştir. Bu yöntem APF-RL'den farklı olarak ortamdaki insanların konfor alanlarına girmemeye özen gösterir. Sosyal navigasyon ortamdaki insanların algılanmasını ve uzamsal olarak takip edilmesini gerektirir. Bunun için bir derin öğrenme temelli insan algılama yöntemi Kalman filtresiyle birleştirilerek kullanılmıştır. Son olarak geliştirilen hareket yöntemleri insan takibi uygulamasında kullanılmak için de uygun haline getirilmiştir. Önerilen yöntemler geliştirilen robot üzerinde gerçek hayatta başarıyla test edilmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT iv

ÖZET v

LIST OF FIGURES viii

LIST OF TABLES x

LIST OF SYMBOLS xi

LIST OF ACRONYMS/ABBREVIATIONS xiii

1. INTRODUCTION 1
   1.1. Contribution 2
   1.2. Organization of Thesis 3

2. Social Robot Design 4
   2.1. Related Literature 4
   2.2. Our Design 5
   2.3. SempRob Software Design 8

3. Mapless Robot Navigation 9
   3.1. Related Literature 10
   3.2. Artificial Potential Functions 12
   3.3. Elliptic Obstacle Modeling 14
   3.4. Reinforcement Learning 17
   3.5. APF-RL 19
      3.5.1. RL Method 19
      3.5.2. Environmental Complexity 22
      3.5.3. Training 24

3.6. Experimental Results 25
3.6.1. Simulation Results 26
3.6.2. Real Robot Experiments 28

4. Socially Compliant Robot Navigation 31
4.1. Related Literature 32
4.2. Human Detection 33
4.3. Human Tracking 34
4.4. Socially Compliant Robot Navigation 35
4.5. Experimental Results 40
4.5.1. Simulation Results 40
4.5.2. Real Robot Results 43

5. Socially Compliant Human Following 45
5.1. Related Literature 45
5.2. Human Following 46
5.3. Experimental Results 47
5.3.1. Simulation Results 47
5.3.2. Real Robot Results 48

6. CONCLUSION AND FUTURE WORK 51

REFERENCES 53

7. APPENDIX A: ROBOT MANUAL 61

8. APPENDIX B: SOFTWARE USAGE 63

9. APPENDIX C: IMAGE USAGE 64

LIST OF FIGURES

Figure 2.1. SempRob: Complete design. 6

Figure 2.2. SempRob: Head and face design. 6

Figure 2.3. Body Shell Parts. 7

Figure 2.4. SempRob robot. 7

Figure 2.5. SempRob head and face. 8

Figure 3.1. Potential Field Visualization. 12

Figure 3.2. Disk-Based Approximation. 15

Figure 3.3. Ellipse approximation. 15

Figure 3.4. Representation of obstacles as ellipses for APF. 16

Figure 3.5. Distance computation to an ellipse obstacle model. 16

Figure 3.6. Basic Scheme of Reinforcement Learning. 18

Figure 3.7. APF-RL method. 18

Figure 3.8. Intermediate goal region $\mathcal{W}$. 19

Figure 3.9. Training Environment. 25

Figure 3.10. Learning Curve of APF-RL. 26

Figure 3.11. Simulation test environments. 27

Figure 3.12. Comparative paths: APF-RL vs DWA+D. 30

Figure 4.1. Proxemics zones. 31

Figure 4.2. YOLO Human Detection Example. 34

Figure 4.3. Examples of enlargement of human ellipses. 37

Figure 4.4. Learning Curve of Social APF-RL. 39

Figure 4.5. Training Environment of Social APF-RL. 39

Figure 4.6. Test Environments for Social Navigation. 41

Figure 4.7. Sample test environment for real robot experiments. 43

Figure 4.8. Social APF-RL Example 44

Figure 5.1. Flow of processing for human tracking. 45

Figure 5.2. Human Following Location Selection. 47

Figure 5.3. Human following simulation: Human path vs robot path. 48

Figure 5.4. Example Following Environment. 49

Figure 5.5. Following Paths Comparison. 49

LIST OF TABLES

Table 3.1. Related Works for Mapless Robot Navigation. 11

Table 3.2. Static environment simulation results. 28

Table 3.3. Dynamic environment simulation results. 28

Table 3.4. Real robot results. 29

Table 4.1. Socially Compliant Navigation: Simulation Results. 42

Table 4.2. Socially Compliant Navigation: Real Robot Results. 44

Table 5.1. Human Following: Simulation Results. 48

Table 5.2. Human Following: Real Robot Experiments. 50

Table 7.1. Characteristics of the Base Platform. 61

LIST OF SYMBOLS

a_t	Action at time t
c	Robot pose
d_f	Human following distance
EC	Environment complexity
EC_s	Static environment complexity
EC_d	Dynamic environment complexity
g	Goal position
$g*$	Final goal location
h_c	Human coefficient for ellipse
h_{room}	Length of the room
k	Attraction parameter of APF
$\mathcal{O}$	Obstacles
o	Obstacle
$\mathcal{S}$	State space
q	Laser data
q_h	Label of the laser data
r_g	Goal reward function
r_h	Social penalty function
r_o	Obstacle reward function
r_t	Reward at time t
s_t	State at time t
T_M	Episode time limit
V_h	Velocity of the human
v_i	Velocity of ith obstacle
v_{max}	Maximum linear Speed
v_r	Velocity of the robot
w_{max}	Maximum angular Speed
w_{room}	Width of the room

$\beta_{\mathcal{O}}$	Repulsion function of APF
γ_g	Attraction function of APF
θ_f	Human following angle
θ_i	Yaw of ith obstacle
π_t	Policy at time t
ρ	Radius of the robot
ρ_i	Radius of ith obstacle
ρ_o	Radius of the obstacle
τ_o	Collision distance threshold
τ_p	Goal proximity threshold
τ_x	X size of waypoint region
τ_y	Y size of waypoint region
$\hat{\varphi}$	Potential function of APF
Ω	Observation space
ω	Observation

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
APF	Artificial Potential Function
APF-RL	Artificial Potential Functions with Reinforcement Learning
CNN	Convolutional Neural Network
DRL	Deep Reinforcement Learning
DWA	Dynamic Window Approach
DWA+D	Dynamic Window Approach with Dijkstra
HOG	Histogram of Gradients
IRL	Inverse Reinforcement Learning
LIDAR	Light Detection and Ranging
SAC	Soft Actor Critic
SFM	Social Force Model
USB	Universal Serial Bus
YOLO	You Only Look Once

1. INTRODUCTION

While mobile robots have been initially used in the automation of industrial work, more and more, they are aimed to be used in different sectors such as service, health or education. This requires their operating in human-populated environments such as homes, cafeterias, hospitals or airports. Sharing common workspaces leads to new possibilities for human-robot interaction - such as robot guides [1]. As such, the concept of 'social robots' has been introduced [2]. One of the primary features of these robots is to consider explicitly human presence. This is observed to manifest itself in two related aspects: physical realization and social navigation.

The former is important since the design of a robot has a significant effect on what people think about them. Therefore, a robot with all necessary social features but having an unappealing design cannot be successful at social interaction. The latter is important since most service tasks require the robot to move around people. There are two primary considerations in regards to social robots and navigation.

- First, social mobile robots are expected to be capable of navigating reliably even in dynamically changing environments. Thus, they should be able to navigate without using maps or external plans. This is because map building and updating may not be practical due to time-sensory data constraints and/or the environment might be dynamic. As such, they differ from most of their industrial counterparts that typically rely on such maps or plans. Rather, their navigation needs to be reactive - namely they should rely solely on the incoming sensory data. The robots need to be capable of reaching to their goal locations without any collisions along the way.
- Second, again differing from their industrial counterparts, social mobile robots are additionally expected to navigate considering the social norms and human comfort - even if map knowledge is not available [3]. There are two primary scenarios that differ in the robot's goal.

  - Socially compliant navigation in human-populated areas: In these scenarios, socially compliant navigation can be explained as navigating like a human without disturbing the people around. Social robots must have this feature so they can work in settings like airports and shopping malls.
  - Socially compliant human following: In these scenarios, socially compliant navigation can be explained as navigating behind the human while respecting his/her personal zone.

This thesis has focused on both the physical realization of a social robot and social navigation methods - considering mapless navigation, socially compliant navigation in human-populated areas and socially compliant human following.

1.1. Contribution

The contributions of this thesis can be summarized as follows:

- Physical realization: A social robot named as SempRob has been designed and developed. Likability has been the primary consideration in the design. In addition, it is equipped with the necessary visual sensors in order to perceive the environment and act accordingly. SempRob is used for experiments of this thesis and it can be used for a variety of daily-life tasks.
- Mapless robot navigation: A novel reactive navigation method referred to as Artificial Potential Function with Reinforcement Learning (APF-RL) method is proposed. In addition, an ellipse-based representation of obstacles is developed. Furthermore, environmental complexity measures are defined as to ensure a wide range of scenarios are used in learning. Differing from previous work, while APF-RL method does not require map of the environment, it can also be used even in complex environmental settings.
- Socially compliant robot navigation: APF-RL method is extended to Social APF-RL so that the robot's navigation becomes socially compliant. As such, the robot takes social conventions into account while navigating and aims to stay away from

the personal zones of pedestrians around as much as possible.

- Socially compliant human following: The proposed Social APF-RL is modified for human following. The method is evaluated both in simulation and on the real robot.

1.2. Organization of Thesis

The organization of the thesis is as follows:

Chapter 2: The social robot designed and manufactured is presented in in this chapter. First, related work is discussed. Then, the proposed design is explained in detail.

Chapter 3: The proposed mapless navigation method artificial potential functions with reinforcement learning (APF-RL) is presented in this chapter. First, related literature on mobile robot navigation is summarized. Following, the details of the proposed method are explained. First, navigation based on artificial potential functions is discussed. Here, obstacle representation based on ellipses is formulated. Following, the usage of deep reinforcement learning is explained. Two measures of environmental complexity are defined and used to determine the range of learning scenarios. Finally, both simulation and experimental results along with a comparative study are presented.

Chapter 4: The proposed socially compliant navigation method - Social APF-RL, is presented in this chapter. First, related literature on social navigation is discussed. Next, the details of the proposed method are explained. Finally, experimental results with a comparative study are presented.

Chapter 5: The details of human following based on modifying Social APF-RL method are given in this chapter. First, related literature on human following is discussed. Next, the details on human detection and following methods are explained. Finally, experimental results are presented.

Chapter 6: The thesis concludes with a summary and future work as presented in this chapter.

2. Social Robot Design

This chapter focuses on the design and development of a social mobile robot. The robot is expected to have both likable appearance and social navigation capability as to realize its given tasks. The head and the face of the robot are known to be the main attention regions for humans, so their designs needed to be addressed separately. Social navigation requires the robot to be able to navigate around without any collisions while also taking human presence into account. As this requires the robot to sense its surroundings and in particular to detect humans, the robot should be endowed with the appropriate visual sensors.

The outline of the chapter is as follows: First, a brief summary of existing social robot designs is presented in Section 2.1. Following the design and development of SempRob is explained in Section 2.2 with the explanation of the design choices.

2.1. Related Literature

The design of a social robot can be viewed as consisting of two main parts: the head and the body. While, simple shapes that encase the hardware of the robot might be sufficient, head design requires more craftsmanship. This is primarily attributed to the fact that the face plays a key role in the robot's social acceptability. Mathur and Reichling's work aims to explain likability of robot designs based on real life examples [4]. By looking these examples, one can notice that initial designs are too edgy and mechanical which makes them unattractive. Some recent robots are designed to resemble human appearance. A human-like appearance is often preferred for robots that operate in human environments. However, a completely anthropomorphic design has the disadvantage that it implicitly raises expectations regarding certain cognitive capabilities of the platform, which cannot be accomplished with current technology. This can lead to disappointments or to refusal of the system, which is known as Uncanny Valley effect [5]. Using an LCD screen as a face has recently become popular -

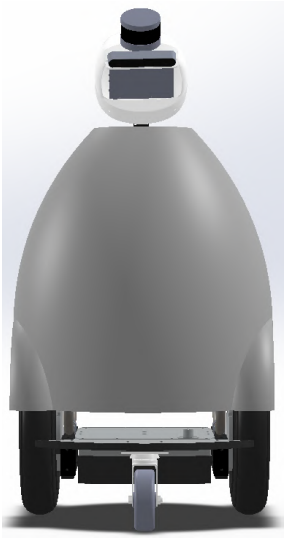
as it is modular and relatively simple. Face animations are shown in these screens and the animations might be updated based on the situation, for example it can be used to express emotions.

The body of the robot must also encase the hardware of the robot and be likable. There are two main approaches on body design: designs that resemble upper human torso and designs that resemble the whole human body. Two well-known examples are Jackrabbot [6] and Pepper [7]. Jackrabbot has a design similar to an upper body while it has a single arm and visible wheels. On the other hand, Pepper has much more human-like design with a human-like stance.

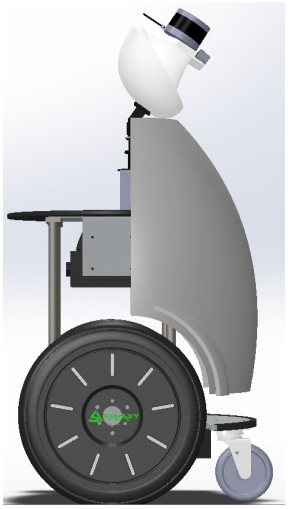
2.2. Our Design

The design and development of our social robot SempRob has been done as follows:

- Robot base: The robot base is realized using self-balancing system developed by Segway [8]. This is a a commercially available differential wheel mechanism.
- Body encasing: The robot's base is to be encased by a body shell printed from 3D printer.
- Head: The head is realized using a pan-tilt mechanism. As such, the head is rotatable.
- Sensing: The head will have a stereo camera and a LIDAR for environmental sensing. The stereo camera is a ZED 2 stereo camera and the LIDAR is a RoboSense RS-16 3D.
- Face: A screen will be placed on the head for the face animations, because it is prettier and easier than adding facial components. An open-source animations are used [9] and it lets our robot to express emotions.
- Processing: A powerful on-board computer will be used to process the data collected from the sensors and to send control commands to the motion controller. In particular, Nvidia Jetson Xavier board is used.



(a) Front view.

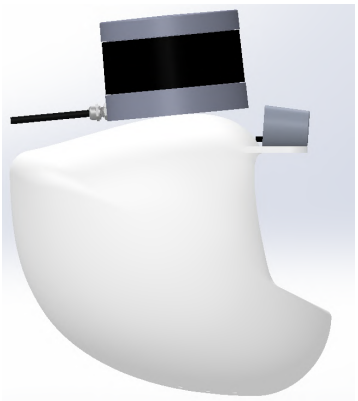


(b) Side view.

Figure 2.1. SempRob: Complete design.

In the design of body and head, the two primary considerations have been design cuteness and ease of production. The complete design with the body shell and the head is shown as front and side view in Figure 2.1. A simple torso-like part has been designed to be used as the body shell. As it can be seen, the body shell covers the front of the robot and resembles a torso. The design of the head and face is shown as front and side view in Figure 2.2. As it can be seen, it has a sympathetic expression with soft lines and face animation. The sensors are placed on top of it so that they can rotate with the head.

(a) Front view.



(b) Side view.

Figure 2.2. SempRob: Head and face design.

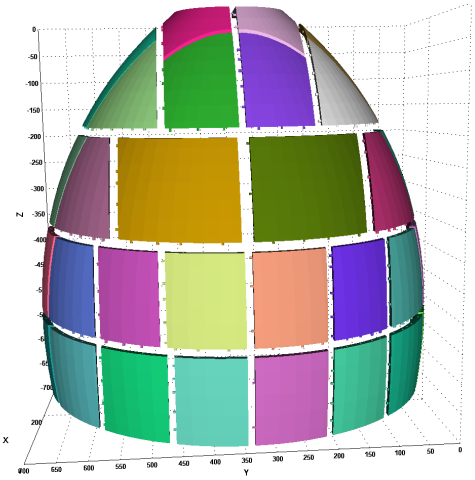
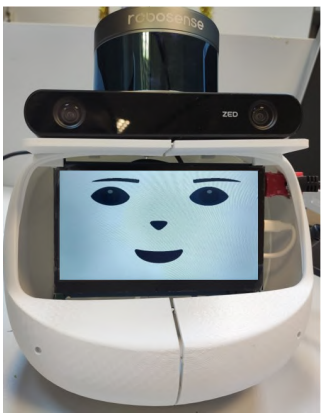


Figure 2.3. Body Shell Parts.

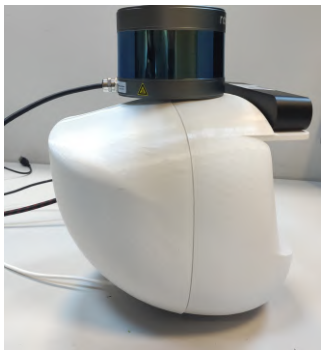
The body and head parts are manufactured using a 3d printer. However, it is not possible to print them as a single part because of size limitations of the printer. So, they are split to smaller parts and printed as parts as can be seen in Figure 2.3. The parts have plug connectors to merge them after they are all printed.

The complete realization of SempRob can be seen in Figure 2.4. And the realized head can be seen in Figure 2.5.

Figure 2.4. SempRob robot.



(a) Front view.



(b) Side view.

Figure 2.5. SempRob head and face.

2.3. SempRob Software Design

The base robotic platform has a host computer that handles inner-loop control and reads data from base sensors like wheel encoders and imu. The onboard Nvidia Xavier computer is connected to this host using ethernet and communication between these is handled using scripts provided by Segway. These scripts use Robot Operating System (ROS) to publish incoming data so that other modules can listen and use them. It also listens to velocity commands and sends them to the host computer. The velocity command is expected at 20 hz by default but it can be changed if needed. The velocity command is then converted to wheel speeds in host computer and applied to the robot. Odometry calculation is done using wheel speeds and it is used to localize the robot in the environment.

The camera and lidar sensors are directly connected to the onboard computer. Robosense lidar is connected using an ethernet cable. Its ROS packages are also provided and it publishes point cloud data that can be used for sensing the environment. The ZED 2 stereo camera is connected using USB. The company is provided an SDK and ROS wrappers. When the ROS scripts are run all topics including left, right, and merged RGB data and corresponding depth data. It also has an object detection model that can be activated if desired. The camera also has an inertial measurement unit (IMU) and its output can be used to improve odometry.

3. Mapless Robot Navigation

This chapter is focused on the first aspect of social navigation - namely mapless navigation. Here, the robot's navigation needs to be reactive - namely they should rely solely on the incoming sensory data and should be capable of reaching to their goal locations without any collisions along the way. Here, there may be static or dynamic entities in the environment. While some of these entities may be humans, the robot does not take this into account specifically. All are treated as obstacles with which there must be no collisions.

There has been extensive work done in this area [10, 11]. Classical reactive methods such as artificial potential functions (APF) are known to have safety guarantees in certain conditions [12]. However, while they have well-proven performance in simple spherical worlds, such performance does not extend to real-life environments that tend to be more complex. In particular, the robot is likely to get stuck in some local minima points so that arrival at the goal location is not ensured.

Recently, learning-based approaches are shown to scale well to complex environments [13]. However, they also have problems like sample inefficiency and generalization. It is hard to predict their performance in unseen environments and they do not have a safety guarantee.

In this chapter, a novel navigation method is proposed. This method is built upon prior work on navigation using artificial potential functions and reinforcement learning. The motivation has been to combine the classical and learning-based navigation methods to obtain the best of both worlds. As explained, APF is theoretically a collision-free reactive method and it is proven to work well in convex worlds. However, it doesn't show such performance in real-life scenarios where the spatial arrangement of the environment is more complex. On the other hand, learning-based navigation methods perform better in complex environments. However, they don't have a safety

guarantee especially in environments that are very different from the training environment. The proposed method combines these two methods to obtain safe mapless navigation in complex real-life scenarios.

The outline of the chapter is as follows: Related literature is summarized in Section 3.1. This is followed by a brief description of artificial potential function based navigation in Section 3.2. The APF method is modified so that obstacles are represented by ellipsoids as presented in Section 3.3. Deep reinforcement learning is described briefly in Section 3.4 respectively. The proposed approach - namely artificial potential functions with reinforcement learning (APF-RL) - is explained in detail in Section 3.5. The chapter concludes with experimental results for both simulation and real robot tests in Section 3.6.

3.1. Related Literature

Classical navigation methods can be divided into two groups based on using maps. In the methods of the first group, the robot uses a global and potentially dynamic map of the environment to follow a collision-free path to reach the target location. Dynamic Window Approach (DWA) is one of the well-known methods in this group [14]. It is usually combined with a global planner which produces a global plan which DWA follows. The required memory and computational resources increase as the complexity of the environment increases. The methods in the second group like artificial potential function (APF) do not use external inputs such as a map or a global plan. Rather, the control input is calculated based on the robot's instantaneous or short-term knowledge of obstacles around only [12, 15]. While these methods have behavioral guarantees, they only hold for restricted environmental settings i.e. convex worlds. These limitations have been addressed by using a sequence of intermediate goal locations that lead to the real goal in several works [16–20]. However, these methods also have some environmental restrictions or they require rigorous tuning of the method parameters including the locations of the intermediate goals. So both groups have drawbacks that need to be addressed.

Table 3.1. Related Works for Mapless Robot Navigation.

Work	**Learning**	**Mapless**	**Setting**	**Obstacles**	
				Internal	**Dynamic**
[21]	E2E	✓	Roads	✓	✗
[22]	E2E	✓	Maze	✗	✗
[23]	E2E	✓	Indoor	✓	✗
[24]	Mod	✓	Roads	✗	✗
[25]	Mod	✗	Indoor	✓	✓
[26]	E2E	✗	Maze	✗	✗
[27]	Mod	✓	Indoor	✓	✓
[28]	Mod	✗	Indoor	✓	✓
Proposed	Mod	✓	Indoor	✓	✓

Deep reinforcement learning (DRL) is observed to offer a lot of potential in this regard. Since it is proven to work well in several robotics tasks, navigation task can also be learned with trial and error. The related works on this topic can be classified depending on the structure of the method, usage of the maps and application areas as can be seen in Table 3.1. Most of these works are end-to-end which means the problem is directly solved with DRL and the output of the network is velocity command [21,23]. Although they work well in environments similar to their training environments, they have problems in more realistic settings. To handle sample inefficiency of E2E methods, works that use expert or imitation data for learning are proposed [24,29]. The drawback of using imitation data is that it makes the method biased with the data. Some E2E approaches such as [22, 26] consider maze-like game environments. However they may not be suitable for use in everyday scenarios since the agents have never encountered internal obstacles.

In order to obtain the best of both classical and learning-based methods, modular approaches that combines them are proposed. For example, learning agent selects way-points and reactive controller calculates velocity command for next way-point. [25] and [28] combines well know sampling-based global planners with DRL. However, they

require map of the environment for calculating the global plan. [27] teaches way-point selection to an agent using expert MPC data, then combines it with a feedback-based trajectory tracking controller.

3.2. Artificial Potential Functions

Navigation based on Artificial potential function (APFs) is a reactive navigation approach that proposes a safe and smooth navigation in disk-like worlds [12]. In this approach, all obstacles are assumed to known and are represented by enclosing disks. the gradient of the APF is used to construct a velocity vector field. The robot's navigation is achieved via simply moving on the resulting vector field. The APF is constructed so that the goal location creates an attractive field while the obstacles create a repulsive field. An example scenario with potential field visualization is shown in Figure 3.1.

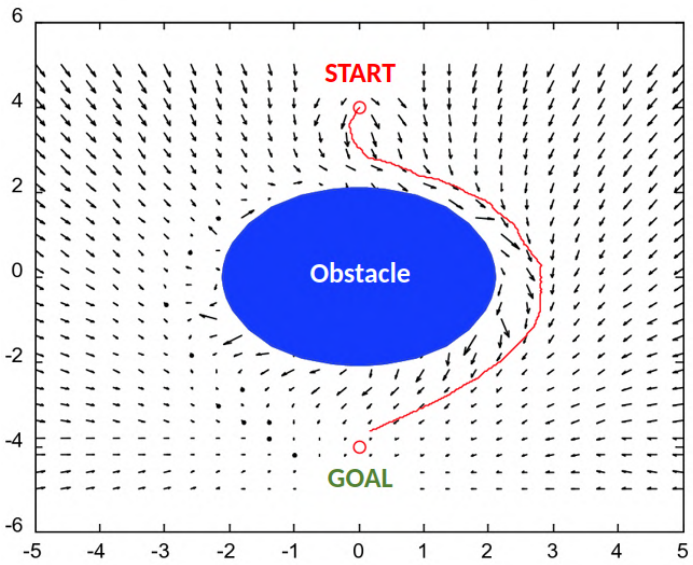


Figure 3.1. Potential Field Visualization.

Consider a robot with radius ρ_r that is located at $c \in R^2$ with coordinates $c = \begin{bmatrix} c_1 & c_2 \end{bmatrix}$. The potential function $\hat{\varphi}_{g,\mathcal{O}}$ is defined as

$$\hat{\varphi}_{g,\mathcal{O}}(c) = \frac{\gamma_g^k(c)}{\beta_{\mathcal{O}}(c)}. \tag{3.1}$$

Here, the numerator term defines attraction and depends on the goal position $g \in R^2$. The formulation of the $\gamma_g(c)$ is as follows:

$$\gamma_g(c) = (g - c)^T (g - c). \tag{3.2}$$

The denominator term $\beta_{\mathcal{O}}(c)$ encodes the obstacles as

$$\beta_{\mathcal{O}}(c) = \prod_{o \in \mathcal{O}} \beta(c, o) = \prod_{o \in \mathcal{O}} (\|c - c_o\|^2 - (\rho_r + \rho_o)^2). \tag{3.3}$$

$\mathcal{O}$ refers to the set of obstacles. Each obstacle $o \in \mathcal{O}$ is defined by its center c_o and radius $\rho_o > 0$. The obstacle function is defined so that as the robot approaches to an obstacle $o \in \mathcal{O}$, $\beta(c, o) \to 0$. If the robot touches the obstacle, then $\beta(c, o) = 0$ which will result infinite potential. This is what makes APF theoretically safety guaranteed.

The $k > 0$ parameter designates the weight of $\gamma_g(c)$ with respect to the obstacle function $\beta(c)$. Thus, it determines the balance between goal attraction and obstacle avoidance.

The velocity control u is derived from potential $\varphi_{g,\mathcal{O}}$ as

$$u(t) = \begin{bmatrix} v_{max} \cos\left(\angle(\nabla_c \varphi_{g,\mathcal{O}}(c))\right) \\ \omega_{max} \sin\left(\angle(\nabla_c \varphi_{g,\mathcal{O}}(c))\right) \end{bmatrix}. \tag{3.4}$$

Here, v_{max} and ω_{max} are linear and angular speed limits and $\angle(\nabla_c \varphi(c)) \in S^1$ is heading as defined by the gradient $\nabla_c \varphi_{g,\mathcal{O}}$ of the artificial potential function φ as

$$\angle(\nabla_c \varphi_{g,\mathcal{O}}(c)) = \ arctan(\frac{\partial \varphi_{g,\mathcal{O}}(c)}{dc_2}, \frac{\partial \varphi_{g,\mathcal{O}}(c)}{dc_1}). \tag{3.5}$$

The control input u is given to robot at each time step and the navigation is completed when goal location is reached, namely: $|c - g| < \tau_p$. The parameter τ_p denotes proximity threshold for goal position.

In this perspective, the APF method is observed to have the following shortcomings:

- First, the goal is set directly as the final goal, $g = g^*$. This is assuming that all obstacles are known and have convex shapes. However, both assumptions are not practical in real-life environments. As such, theoretical guarantees cease to hold. The robot might get trapped in some regions like u-shaped areas. This suggests that the robot should not aim at the goal location directly. Rather, it could try to reach a sequence of intermediate goal locations that eventually lead to the goal location. The intermediate goal locations would depend on both the environment and the robot's current location. Consequently, the goal function g needs to be defined as a piece-wise constant function $g : \boldsymbol{R} \to \mathcal{C}$ with g^* as its limit - namely $\lim_{t\to\infty} g(t) = g^*$.
- Second, $k-$ parameter is set as a constant - namely $k = k^*$ and this value needs to be set depending on the environment. However, it is not possible to find optimal value in unknown or dynamic environments. Setting it as small might cause problems in narrow passages while a larger value may result in robot speeding up unnecessarily. This suggests that it should be selected online and hence it needs to be defined as a function $k : \boldsymbol{R} \to \boldsymbol{R}^{\geq 0}$.

3.3. Elliptic Obstacle Modeling

As previously stated, the robot is endowed with two-dimensional (2D) laser with 360 °horizontal field. The incoming laser data is used to detect the obstacles in the environment. Obstacles have been modeled as disks in the original APF method. However, objects in real life usually do not have circular shape. This has been addressed using two alternative circular approximation methods: single disk and multiple disks. Approximation results for a single rectangular block is visualized in Figure 3.2. In the former, the approximation covers much more space than the real obstacle and it narrows down available navigation space. In the multiple disks model, while the representation is much more realistic, the construction of the potential field becomes

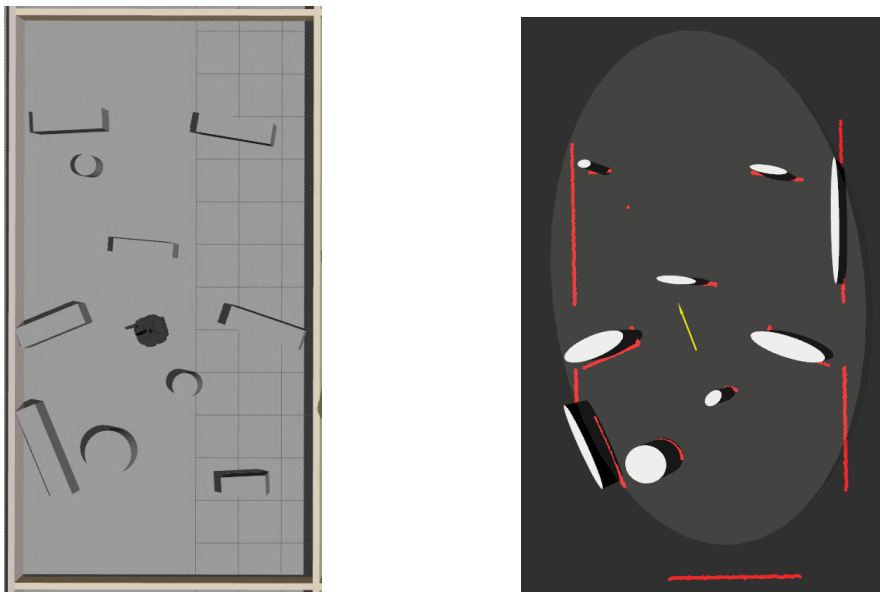
much more complex and navigation performance is adversely affected.

(a) Single Circle. (b) Subcircles.

Figure 3.2. Disk-Based Approximation.

Figure 3.3. Ellipse approximation.

In this thesis, the representation of obstacles is modified to ellipsoid approximation. A sample case is shown in Figure 3.3. As observed, the representation is much more realistic while also not increasing the complexity of APF formulation. The outer boundary is obtained from the ellipse fitted to the convex hull of the laser points. Distinct interior obstacles are detected via applying connected component analysis to the rest of the laser data. Each component is then approximated by an ellipse using a least square based method called numerically stable direct least squares fitting of ellipses [30]. A sample case is shown in Figure 3.4a. The corresponding laser data is shown in red in Figure 3.4b. The resulting obstacles are shown as ellipses and the outer ellipse is visualized as a faint white ellipse.



(a) Robot's environment.

(b) Laser data (red) and corresponding obstacles (white ellipses).

Figure 3.4. Representation of obstacles as ellipses for APF.

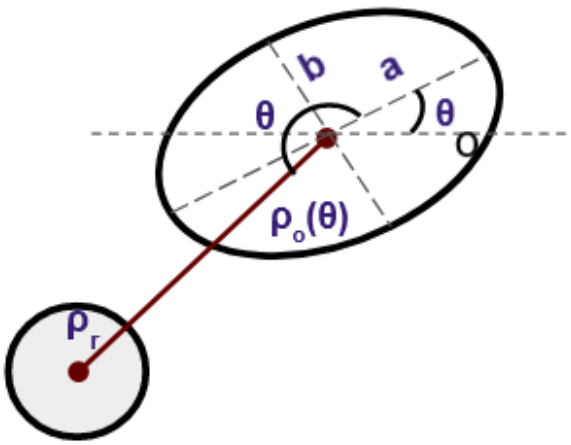


Figure 3.5. Distance computation to an ellipse obstacle model.

The ellipse representation requires the modification of the APF formulation - since the distance between the robot and the obstacle changes depending on their relative positioning and the orientation of the ellipse as seen in Figure 3.5. In particular, the term ρ_o is defined as

$$\rho_o = \frac{ab}{\sqrt{a^2 \sin^2(\theta) + b^2 \cos^2(\theta)}} \text{ where } \theta = \arctan(c - o) - \theta_o. \tag{3.6}$$

Here, a and b refers to the width and height of the corresponding ellipse along major and minor axis directions as shown in Figure 3.5. As this term is part of the obstacle avoidance term, the gradient of $\beta(c, o)$ is derived as

$$D_c\beta(c,o) = 2(c-o) - 2(\rho_r + \rho_o)D_c\rho_o, \tag{3.7}$$

$$D_c\rho_o = \frac{\mathrm{d}\rho_o}{\mathrm{d}\theta} D_c\theta, \tag{3.8}$$

$$\frac{\mathrm{d}\rho_o}{\mathrm{d}\theta} = \frac{ab\sin(\theta)cos(\theta)(a^2-b^2)}{(a^2\sin^2(\theta) + b^2\cos^2(\theta))^{\frac{3}{2}}}, \tag{3.9}$$

$$D_c\theta = \begin{bmatrix} \frac{\mathrm{d}\theta}{\mathrm{d}c_1} \\ \frac{\mathrm{d}\theta}{\mathrm{d}c_2} \end{bmatrix} = \begin{bmatrix} -\frac{c_2-c_{o2}}{(c_1-c_{o1})^2+(c_2-c_{o2})^2} \\ \frac{c_1-c_{o1}}{(c_1-c_{o1})^2+(c_2-c_{o2})^2} \end{bmatrix} \tag{3.10}$$

where $c_o = \begin{bmatrix} c_{o1} & c_{o2} \end{bmatrix}$ refers to the center of obstacle o.

3.4. Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning that is mainly used for decision-making problems. The basic scheme of reinforcement learning is shown in Figure 3.6. Unlike supervised and unsupervised learning, RL doesn't require pre-collected or labeled data. Rather, it uses trial and error for learning. RL problems are usually modeled as a Markov Decision Process (MDP). In this process, the agent takes an action based on the observation or state of the environment. Then the state of the environment changes and the agent chooses new action for this new state. It also gets a reward for each action and the reward amount measures the goodness of the selected action. The agent aims to find a policy that selects actions as to maximize the total reward.

In the simplest case the state space is finite and small. Hence, the agent can explore most of the state action pair quickly. Tabular reinforcement learning methods are used for such problems. As the space becomes bigger or is defined to be continuous, tabular methods cease to be practical. In these cases, function approximators that calculate the value of a state-action pair are used for such problems. These approximator

functions can be linear in the simplest case, but nowadays neural networks are mostly used.

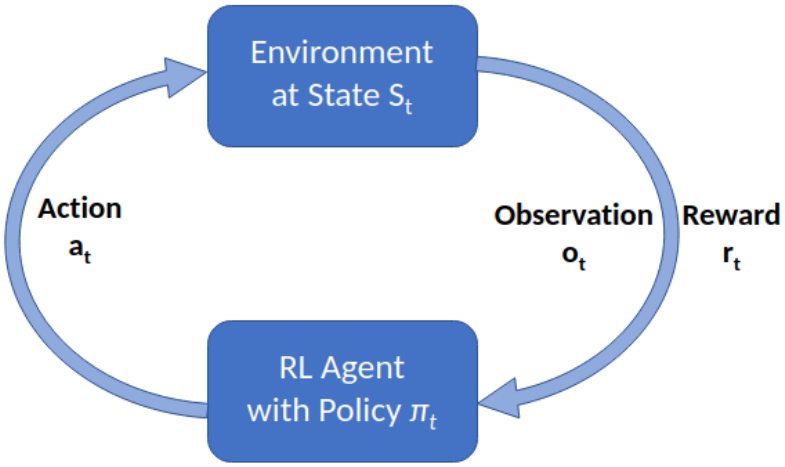


Figure 3.6. Basic Scheme of Reinforcement Learning.

The pioneer deep reinforcement (DRL) learning methods are deep deterministic policy gradient (DDPG) [31] and deep Q-learning(DQN) [32]. Many other DRL methods are developed based on them. Soft Actor Critic (SAC) is one of the most popular DRL methods, it is an actor critic method as it name suggests [33]. It separates action selection and value function learning using two networks - actor and critic. One of the key contributions of SAC is entropy regularization. It tries to maximize entropy, a measure of randomness in the policy, besides the expected return, which accelerates the exploration.

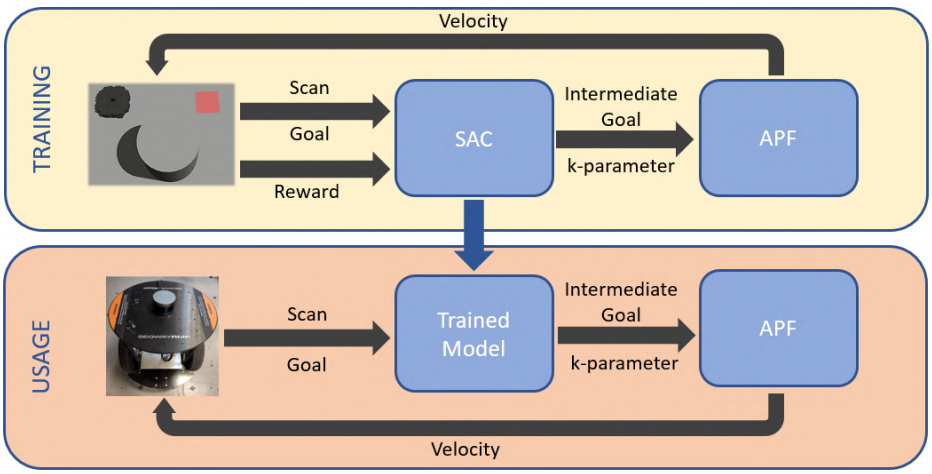


Figure 3.7. APF-RL method.

3.5. APF-RL

The proposed artificial potential function with reinforcement learning (APF-RL) method is shown in Fig. 3.7. In this approach, the two input parameters to the APF are considered:

- Intermediate goal locations g
- k-parameter

Our proposed approach APF-RL is focused on these two parameters. As such, our goal is to use reinforcement learning for learning the $k-$ parameter function and the intermediate goal function g so that the robot can use the learned functions for reliable APF-based navigation. Soft Actor-critic method is selected as the deep reinforcement learning method [33]. In addition, two novel measures of environmental complexity are introduced. The first is for static environments while the second is for dynamic environments. These measures are used to ensure the training scenarios encompass a range of environmental complexities.

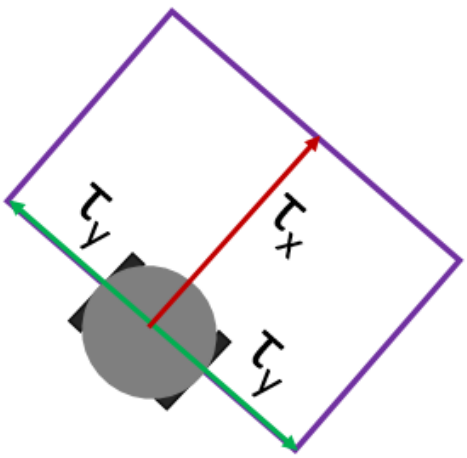


Figure 3.8. Intermediate goal region $\mathcal{W}$.

3.5.1. RL Method

Deep reinforcement learning is used to learn the inputs and parameters of APF - namely the goal function g and k-parameter function k. The problem is formu-

lated as a partially-observable Markov decision process (POMDP) as defined by the tuple $(\mathcal{S}, \Omega, \mathcal{A}, \mathcal{P}, r)$. The state space $\mathcal{S}$ defines the full state of the agent in the environment. At each time step, state $s(t) \in \mathcal{S}$ contains global information about the environment including the robot but the state is only partially observable by the robot. The observation space is denoted by Ω defines this partial information that the robot gets from the environment. The observation $\omega(t)$ consists of the current laser scan data $q(t)$ and the previous one $q(t - \delta t)$ and the respective goal positions - namely $\omega(t) = (q(t), q(t - \delta t), g(t), g(t - \delta t))$ where δt is the time-step of processing. The previous range data is added to the observation in order to encode the movement of the robot and obstacles.

The radius of the robot ρ_r is subtracted from the range data before adding to observation, so the trained model is independent from the robot size. Let $\mathcal{A} \subset \boldsymbol{R}^3$ define the action space. Each action $a \in \mathcal{A}$ is defined by the 3-tuple $a = \begin{bmatrix} k & g^T \end{bmatrix}$ where $k \in \mathcal{K}$ defines the $k-$parameter and $g \in \mathcal{W}$ corresponds to an intermediate goal location. The set $\mathcal{K} \subset \boldsymbol{R}$ defines the range of the k function. $\mathcal{P}$ is the state transition model and represents dynamics of the system. It is encoded in the physics simulator or implicit in the real world. The region $\mathcal{W} \subset \mathcal{C}$ denotes the region where intermediate goals are selected in. It is a rectangular region with sizes of $2\tau_x \times \tau_y$ as shown in Fig. 3.8. You can see that it is defined in front of the robot to encourage it to move forward and explore. The agent is continually rewarded during training considering goodness of the taken action. Selection of reward function has significant importance on the performance of the agent since the reward is the only feedback given to it. Let $r : R \to [R_{min}, R_{max}]$ denote the reward function with the parameters R_{min}, R_{max} corresponding to the minimum and maximum reward respectively. It is defined based on the proximity of the robot to the goal location as well as the range data in the observation $\omega(t)$ as

$$r\left(t\right) = r_g\left(a(t), \omega(t)\right) + r_o\left(a(t), \omega(t)\right). \tag{3.11}$$

The first term r_g considers the main objective which is reaching to the goal. A big reward is given when the goal is reached and small incremental rewards are given with each time step for getting closer to the goal as measured by $\delta\,|c(t) - g(t)|$. τ_p is the proximity threshold for reaching to goal so the episode ends successfully if the robot gets this close the target. Incremental rewards are positive when the robot gets closer to the goal and negative if it get away. Thus, the robot is encouraged to move towards the goal locations. In practice, we set $R_{g1} = 10$ and $R_{g2} = 2$ which resulted in the best performance after testing with different values. So goal reward function is defined as

$$r_g\left(a(t), \omega(t)\right) = \begin{cases} R_{g1} & \text{if } |c(t) - g(t)| < \tau_p, \\ R_{g2}\delta\,|c(t) - g(t)| & \text{otherwise .} \end{cases} \tag{3.12}$$

The second term r_o is related to the obstacle avoidance objective and penalizes the agent for collisions and getting close to obstacles. Distance to each obstacle $o \in \mathcal{O}(t)$ is calculated using the function $\beta(c, o) \geq 0$. Obstacles are considered to have a safety zone τ_o around them where localization errors or a wrong move may result in collisions. Collision penalty is selected as large since navigation safety is critically important. Thus, the robot is encouraged to follow a safe path. In practice, we set $R_{o1} = -0.1$ and $R_{o2} = -10$ which are observed to result in good balance with the obstacle avoidance and target reaching. So obstacle reward function is defined as

$$r_o\left(a(t), \omega(t)\right) = \begin{cases} R_{o1} & \exists o \in \mathcal{O}(t) \text{ s.t. } 0 < \beta(c, o) < \tau_o, \\ R_{o2} & \text{if } \beta_{\mathcal{O}}(c) = 0. \end{cases} \tag{3.13}$$

End-to-end methods tend to fail in maze-like environments with long walls, because reward function encourages the agent to get closer to the goal. However, in such scenarios the robot needs to move away from the goal to get behind of these walls. Getting away is undesirable for the agent since it means taking negative rewards for tens of steps. As such behaviors are associated with long-term planning, they are problematic

for reinforcement learning methods. In the proposed system, the takes less frequent actions than end-to-end agents as it is integrated with APF. So, it can handle such situations by taking only a few negative rewarded actions.

Soft Actor-critic (SAC) method is selected as deep reinforcement learning method [33]. It is an off-policy actor-critic method that is based on the maximum entropy reinforcement learning framework. Here, policy learning and value function evaluation are done separately. Consequently, there are two networks. The actor network decides which action to take while the critic network tells the actor how good the action was and how it should adjust. The former is achieved via maximizing expected reward while also maximizing entropy. It has three main advantages. Firstly, it is more sample efficient compared to on-policy methods such as TRPO [34], PPO [35] and A3C [36]. Secondly, the method has been shown to have outstanding performance in several continuous control tasks [33], so it could also be suitable for navigation tasks. Finally, it is robust to hyperparameters which means intensive tuning is not necessary unlike DDPG [37] and this characteristic decreases computational needs considerably.

The radius of the learning robot is $\rho_r = 0.4$ meters. The set $\mathcal{K} \subset \boldsymbol{R}$ is defined as $\mathcal{K} = [1, 5]$ from practical considerations - namely the maximum number of obstacles the robot is likely to encounter at a time. The $\mathcal{W}$ region is set with $\tau_x = 1$ m and $\tau_y = 1$ m by experimenting with different values. These values are set as those that give the smoothest path. The architectures of both actor and critic networks are the same. They consist of three hidden layers - each having 512,512,512 neurons respectively. ReLu is used as the activation function except in the action layer with tanh as suggested in [33].

3.5.2. Environmental Complexity

During training, the scenarios that the robot is exposed to need to cover a wide range of environmental complexities as possible. The effectiveness of robot's learning will depend on the range of this exposure. Hence, it is important to quantify this

measure. For the static environments, an environmental complexity EC_s is defined as

$$EC_s = \frac{N_o(N_o - 1)\kappa}{2\beta_s} \tag{3.14}$$

where $N_o = |\mathcal{O}|$ refers to the cardinality of the set of obstacles $\mathcal{O}$. β_s is calculated based on distances between obstacles in the environment as

$$\beta_s = \sum_{i=1}^{N_o-1} \sum_{j=i+1}^{N_o} \beta_{ij} \text{ where } \kappa = \frac{\max(w_e, l_e)^2}{\rho_r^2}.$$

Here w_e and l_e are the respective width and length of the corresponding environment. β_{ij} is calculated as the minimum distance between ith and jth obstacles similar to β in APF formulation, it is formulated as

$$\beta_{ij} = \|c_{oi} - c_{oj}\|^2 - (\rho_{oi} + \rho_{oj})^2. \tag{3.15}$$

Calculation of ρ of an ellipse is explained in Section 3.3. The complexity measure is observed to have the following properties:

- Environmental complexity increases as the number of obstacles N_o increases.
- It increases as the obstacles are located closer to each other. Relatively, it increases as the obstacle size gets larger.
- Finally, it increases as the room gets smaller while the inner obstacles don't change.

Using this definition, static environmental complexity levels are defined as easy ($EC_s < 20$), medium ($20 \leq EC_s < 30$) and hard ($EC_s > 30$). It is harder to define complexity for dynamic environments. In that case speed and number of the dynamic obstacles are used in addition to static complexity. The formula for the complexity of

dynamic environments is

$$EC_d = EC_s + \sum_{i=1}^{N_{obs}} \frac{v_i}{v_r}. \tag{3.16}$$

In this case, as the speed of a dynamic obstacle increases, dynamic environmental complexity increases accordingly. Speed of the robot, v_r is used as a normalizer here.

3.5.3. Training

It is not usually possible to train DRL agents using real robots due to safety constraints, so the training is done using Gazebo simulation software. A training environment with 13 various sized rooms are designed as can be seen in Fig. 3.9. These room are designed to cover complexity space as much as possible.The first three are relatively smaller and differ primarily in the number and shape of obstacles. The next three (4,5,6) have larger size and they are more complex. The next three rooms (7,8,9) have a smaller number of static obstacles, but they additional contain dynamic obstacles with varying velocities. Then there are two corridor-like rooms with long and narrow passages. Finally, the last two rooms (12,13) have a maze-like very complex design. Rooms have defined initial and target spawn regions and the robot and the target is placed inside these randomly at the start of each episode. Episodes continue until one of the three conditions are realized:

- i) robot reaches the goal position,
- ii) robot collides with an obstacle,
- iii) the predefined maximum time $T_M = 120$ seconds is exceeded.

The proximity threshold for reaching the goal $\tau_p = 0.2$ m and safety zone for collision $\tau_o = 0.2$ m. The training starts at the first room and continues with the next one when it is learned. When all rooms are learned, a new room for training is selected at every 10 episodes and selection is done inversely proportional the respective success rate of the rooms. Training is continued until convergence of the average reward $\bar{r}$ is

obtained. For each episode $m > N_T$, average reward $\bar{r}(m)$ is defined as

$$\bar{r}(m) = \frac{1}{N_T} \sum_{m=0}^{m-N_T} \frac{1}{T_m} \int_0^{T_m} r(t')dt'. \tag{3.17}$$

Here, T_m refers to the duration of m-th episode and N_T refers to the number of last episodes considered in the averaging. In particular, we consider $N_T = 300$ episodes. The evolution of $\bar{r}$ is shown in Figure 3.10. It is observed that convergence occurs between $m \in [4000, 5000]$ episodes.

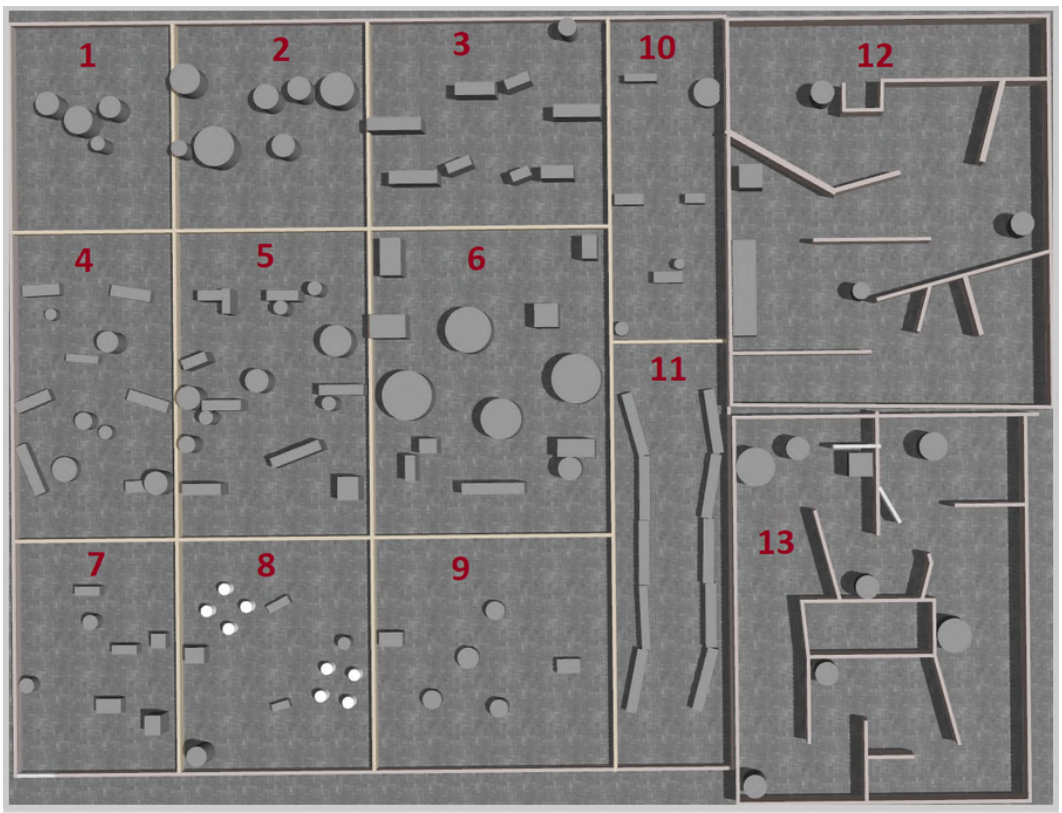


Figure 3.9. Training Environment.

3.6. Experimental Results

The proposed method, APF-RL, is tested both in simulation and with a physical robot. Testing environments are different than training environments and totally unseen by the agent. Performance metrics are selected as success rate (safe arrival at the goal) and path length until reaching the goal. Path lengths are computed considering only successful runs and are meaningful only with high success rates.

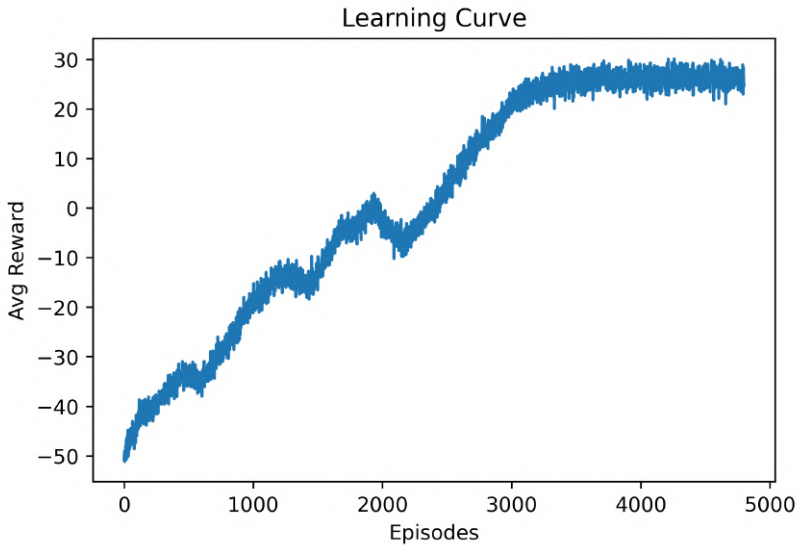


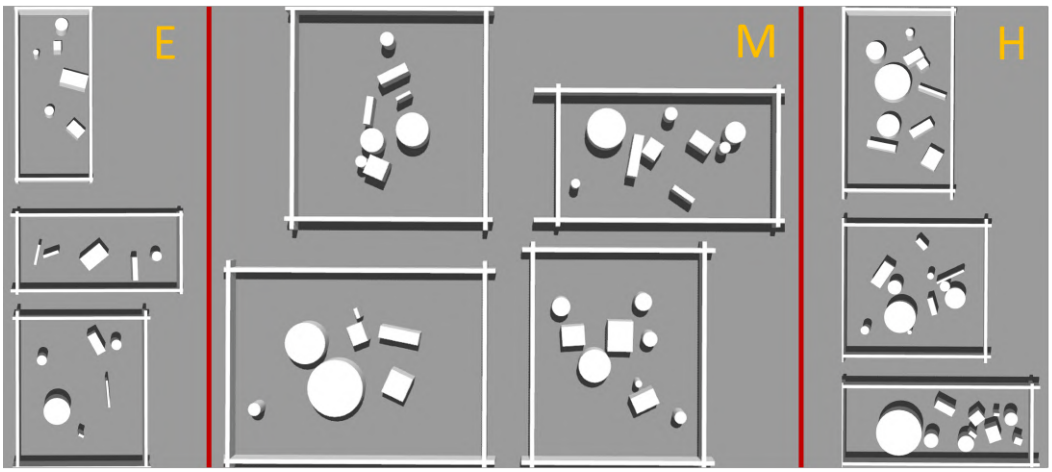
Figure 3.10. Learning Curve of APF-RL.

As the reactive baseline method, we use a modified version of classical APF algorithm in which $k-$parameter is not constant, but rather is set based on number of detected obstacles. As DRL baseline, we trained an end-to-end model. It has a similar network to APF-RL and it outputs linear and angular velocity directly. It is also trained in the same training environment shown in 3.9. Finally as a strong baseline, we use the mapping-based DWA method together with global planner based on Dijkstra (DWA+D). DWA+D has 14 planner parameters that should be tuned properly as they have significant effect on navigation performance. In addition, there are more than 10 additional parameters. While these parameters are less significant, nevertheless they need to be set according to robot and environment specifications.

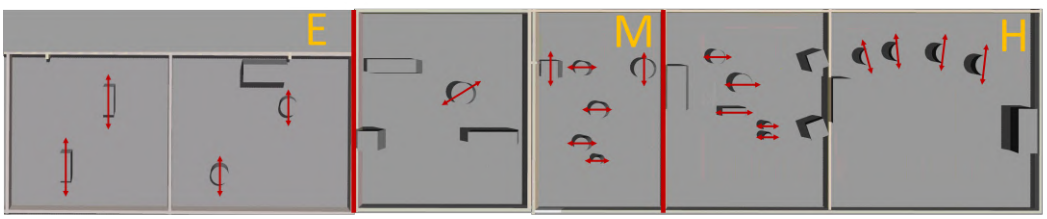
3.6.1. Simulation Results

The evaluation is first done in the simulation environments. 10 environments with only static obstacles are generated as shown in Fig. 3.11a. 3 of them are easy with $EC_s < 20$, 4 of them have medium environmental complexity and finally 3 of them are hard. For evaluating navigation performance with the presence of dynamic obstacles, 6 rooms with moving obstacles are designed as shown in Fig. 3.11b. 2 of these rooms are easy, other 2 of them have medium complexity and the last 2 of them

are hard. 5 initial and goal locations are selected in each room. All methods are tested in these rooms with same initial and target locations.Results of the static environment tests are given in Table 3.2. APF is observed to have worst performance even in the easy rooms. E2E performs average in easy environments but its performance degrades critically as the complexity increases. On the other hand, APF-RL and DWA+D have shown very close performance in all measures. This is promising considering DWA+D make use of a global planner and is a search-based online optimization method while APF-RL is a learning-based reactive method.



(a) Static test scenarios.



(b) Dynamic test scenarios.

Figure 3.11. Simulation test environments.

Table 3.3 shows the results for dynamic environments. APF has a better performance in dynamic environments than static environments. It is related to the fact that, it can reactively escape from obstacles and wait for a passage to be opened. DWA+D performs worse than other methods in the dynamic scenarios because dynamic entities require fast response to the environmental changes and DWA+D is a computationally heavy method. Furthermore, moving obstacles can possibly collide with robot while

it tries to update its plan because a collision avoidance model is not implemented for obstacles. E2E method performs better in dynamic environments but it it has higher collision rate thab APF-RL. APF-RL showed the best performance in dynamic environment tests. It is observed that the robot does not wait for the dynamic entities to move away like APF, rather it preserves its velocity while steering away from them.

Table 3.2. Static environment simulation results.

Method	**Success Rate**			**Path Length (m)**		
	Easy	Med	Hard	Easy	Med	Hard
APF	0,40	0,25	0,07	15,09	12,53	11,44
E2E	0,67	0,35	0,13	14,39	12,85	15,39
DWA+D	0,87	0,75	0,67	12,86	12,75	12,58
APF-RL	0,80	0,70	0,47	13,27	13,90	13,99

Table 3.3. Dynamic environment simulation results.

Method	**Success Rate**			**Path Length (m)**		
	Easy	Med	Hard	Easy	Med	Hard
APF	0,70	0,60	0,90	7,98	10,31	14,84
E2E	0,90	0,70	0,40	7,73	10,03	11,80
DWA+D	0,70	0,50	0,40	5,97	10,19	16,72
APF-RL	1,00	0,70	0,60	7,58	8,86	12,00

3.6.2. Real Robot Experiments

The real robots are done using SempRob as explained in Chapter 2. As its RoboSense RS-16 3D Lidar provides point cloud data, it is mapped to a 2D laser scan. The experiments are conducted in 5 static and 2 dynamic scenarios. The static test environments are set up in a way that each has a unique challenge. For example, the robot needs to get behind of the obstacles that form an L-shape in one of them while in another it needs to pass a sequence of narrow passages. In dynamic tests, there are people walking and they follow the social convention also try not to collide unlike simulation tests. In the first scenario, there are people crossing the path of the robot while in the second, pedestrians are approaching the robot from the opposite directions.

For all cases, methods are repeated 10 times with different initial and goal locations g^*. The results of the physical robot tests are presented in Table 3.4 including results for the baseline methods.

Table 3.4. Real robot results.

Method	**Static Environment**		**Dynamic Environment**	
	Success Rate (%)	Path Length (m)	Success Rate (%)	Path Length (m)
DWA+D	93	6.3	100	5.8
APF	30	10.1	100	6.3
E2E	60	9.2	100	5.7
APF-RL	93	8.5	100	5.5

Similar to the simulation results, APF has the worst performance in static tests. E2E performs better than APF but it is still not good as APF-RL or DWA+D. Interestingly, both DWA+D and APF-RL achieved similar success rates. DWA+D method is observed to generate slightly smoother paths. This is attributed to the fact that it makes use of a search-based global planner. Some sample paths followed using these two methods are shown in Figure 3.12 for comparison. For the case in Fig. 3.12a, DWA+D has smoother and shorter trajectory. In the case of Fig. 3.12b, while path lengths are almost equal, the path of DWA+D is still smoother. In dynamic environments, all methods perform satisfactorily which might be surprising by looking the simulation results. It is attributed to the fact that passing people tend to avoid collisions by stopping or moving away from the path of the robot slightly. This gives DWA+D a chance to stop and update its plan so its success rate is increased considerably. Nevertheless, APF-RL method perform better in dynamic environments in terms of path length and time measures.

(a) Case 1. (b) Case 2.

Figure 3.12. Comparative paths: APF-RL vs DWA+D.

4. Socially Compliant Robot Navigation

This chapter is focused on socially compliant navigation. As discussed previously, dynamic environments are challenging for mobile robots. Unfortunately, human presence further exacerbates the difficulty. Hence, mere collision avoidance does not suffice. Rather, the presence of humans requires novel approaches that consider both the constraints of human comfort and social rules. Proxemics define zones for interpersonal distances for human comfort [38]. These zones are classified depending on their proximity to the human - as seen in Figure 4.1.

- Intimate: 0 - 45 cm
- Personal: 45 - 120 cm
- Social: 1.2 - 3.6 m
- Public: 3.6 - 7.6 m

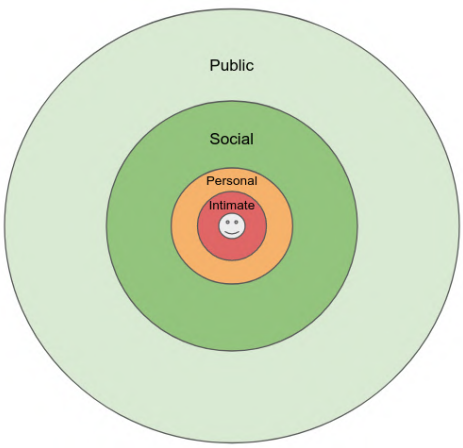


Figure 4.1. Proxemics zones.

In addition to the individual social zones, the individual's interaction with the environment creates other virtual spaces that people around recognize and respect. These are called group, activity and affordance spaces. The group space is based on the interaction with other humans, while the activity space is related to action carried out like taking a photo. The affordance space is related to the potential activity-based of affordance of objects around [39] . In socially compliant navigation, the goal is to

make robots navigate like a human and not invade the personal or intimate space of people around unless direct interaction with them is intended.

The outline of the chapter is as follows: Firstly, related literature is summarized in Section 4.1. Next, human detection is explained in Section 4.2. Then, the details of human tracking method is explained in Section 4.3. Following, the proposed Social Apf-RL method will be detailed in Section 4.4. The chapter concludes with a discussion of experimental study including both simulation and real robot results in Section 4.5.

4.1. Related Literature

Robots that work in human-populated areas and interact with people need to have socially compliant navigation. They should be able to reason about various criteria ranging from clearance, environment structure, social conventions, proximity constraints, presence of a person as well as human groups [40].

Earlier work in this area mostly rely on Social Force Model [41]. In the social force model, the movement of a pedestrian is defined through the sum of attractive and repulsive forces. Attractive forces are based on the the pedestrian's target position and speed, while repulsive forces are applied by obstacles and other humans around. This model has been applied in an application requiring a robot to accompany a person in a socially compliant manner [42]. In this work, an attractive force is associated with the accompanied person. This approach has been extended as to better represent person–person, object–person and robot–person interactions [43].

Recent work has focused on using learning methods for social navigation. These work differ with respect to the learning method and the inputs their method use. For learning, they commonly use either reinforcement learning (RL) and inverse reinforcement learning (IRL). The difference RL and IRL methods lies on the design of the reward function. While the former uses a hand-crafted reward function, the latter is designed using expert demonstrations. A method that combines long short term memory

(LSTM) with deep RL is proposed in [44]. An attention mechanism is used to increase performance of a deep RL approach [45]. The problem is divided into ego safety and social safety and solved suing an end-to-end deep RL method [46]. A hybrid method that addresses both freezing robot problem and socially compliant navigation based on RL is presented in [47]. IRL with different feature sets has been used to achieve socially compliant navigation [48]. Bayesian IRL is used to learn socially normative robot navigation behaviors [49]. A multiagent collision avoidance algorithm that exhibits socially compliant behaviors is proposed in [50]. An imitation learning based method uses a pedestrian trajectory data set to obtain human-like movement [51].

These works also differ in the input that their methods use. They are mainly two categories: i) Human position and velocities are externally provided; or ii) Laser scan readings are input. The former methods take advantage of the recent progress in human detection [50], [44] and [45]. This is considered as a sub-problem of object detection that has become very popular in recent years. Initial work are done using classical image descriptors like histogram of gradients (HOG). However, the use of deep learning methods has caused a leap in recognition performance. Nowadays, a convolutional neural networks (CNN) are extensively used for this problem. YOLO (You Only Look Once) [52] is of the well-known object detection methods. YOLO and most of the other methods outputs only bounding boxes in image space, but there are also methods that directly outputs spatial positions of the objects directly [53]. In the latter, the network also derives the human-related information [47], [51] and [46]. As such, these networks tend to more more complex and require more data.

4.2. Human Detection

Socially compliant navigation requires human detection. Convolutional neural network-based YOLO method is used for this purpose. It has a really complex model with 106 layers. It is trained to classify 80 object classes but only human predictions are used for this work. YOLO takes an RGB image input and returns bounding boxes of the detected objects in the image as output. A sample detection output can be seen

in Figure 4.2. Since YOLO uses RGB image and its outputs are in the image space, we need to locate the detected person in the physical world. The depth stream of the stereo camera is for this purpose. First, the distance of each human is calculated by taking average of depth values from a square region at the upper center of the bounding box. Then the distance is converted to relative planar position based on the location of the person in the image.

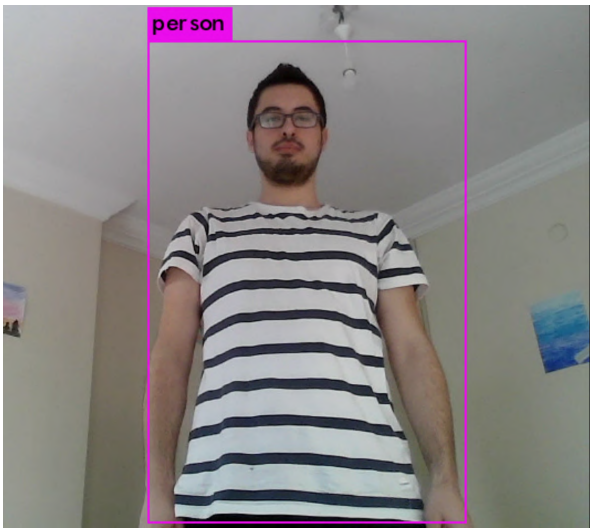


Figure 4.2. YOLO Human Detection Example.

4.3. Human Tracking

For social navigation tasks locating the human target accurately is critically important. Hence, human detection results are augmented with Kalman filtering - as to smooth noisy depth measurements and also predict the location when a human is not detected. It is also used to calculate velocity based on the position measurements. The flow of processing in human tracking is shown in Figure 5.1. The design of the filter is as follows: The state $X = \begin{bmatrix} x_1 & x_2 & \dot{x}_1 & \dot{x}_2 \end{bmatrix}$. Here, x_1 and x_2 are planar positions and $\dot{x}_1, \dot{x}_2$ are the respective speeds of the human target. The human is assumed to move with constant velocity and the system dynamics matrix $A$is constructed accordingly. Human localization module only gives position information so the columns of the observation matrix H corresponding to velocity states are set to zero. The matrices

of the filter are defined as

$$X = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

In order to eliminate the impact of the movement of the robot, Kalman filter calculations are done in a fixed frame which is created using localization of the robot.

4.4. Socially Compliant Robot Navigation

For socially compliant navigation, Social APF-RL method is proposed. This is based on APF-RL. Although APF-RL is developed and tested with dynamic obstacles, it doesn't take human comfort and social rules into account. In the Social APF-RL method, the following changes are introduced:

- Obstacle representation of humans
- Inputs of DRL
- Outputs of DRL
- Rewards of DRL

First, the obstacle representations of humans are modified as to encode movement direction. Recall that each obstacle is represented by an ellipse. This holds for both static and dynamic objects. However, humans might be uncomfortable if a robot gets very close to them - differing from non-living obstacles. We propose an approach in which the ellipse representations of the humans are expanded depending on their motion. Recall that each ellipse is defined by its major e_a and minor axes e_b with corresponding width a and height b as seen in Figure 3.5. Then, the amount of enlargement

is defined by the expansion of the ellipse along the major and minor axis directions as

$$a = a_o + \delta a, \tag{4.1}$$

$$b = b_o + \delta b \tag{4.2}$$

where a_o and b_o refer to the width and length along the respective axes e_a and e_b corresponding to the initial ellipse representation of the human. Suppose the human is detected to be moving with planar velocity v_h. It is calculated using Kalman filter as explained in Section 4.3 as

$$v_h = [\dot{x} \quad \dot{y}]. \tag{4.3}$$

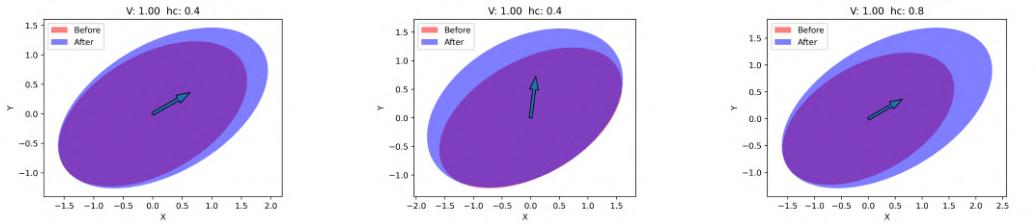
The expansion is done as to minimize the possibility of the robot intercepting with the pedestrian's path. Hence, rather than a symmetric increase in size, corresponding ellipse representations are enlarged in the direction of their movements. To make enlargement possible without increasing the size in the back side, the center is moved in the enlargement direction by half of the enlargement. The amount of expansion is defined by the product of an expansion parameter h_c and the speed of the human along this direction - namely

$$\delta a = \xi_a h_c \|v_h\|, \tag{4.4a}$$

$$\delta b = \xi_b h_c \|v_h\|, \tag{4.4b}$$

$$\xi_a = \begin{cases} 1 & \text{if } v_h^T e_a > v_h^T e_b, \\ 0.5 & \text{otherwise.} \end{cases} \qquad \xi_b = \begin{cases} 1 & \text{if } v_h^T e_a < v_h^T e_b, \\ 0.5 & \text{otherwise.} \end{cases} \tag{4.4c}$$

Three different cases are shown in Figure 4.3. In each figure, the arrows represent the velocity vector.



(a) $v_h^T e_a = 1.0$ $h_c = 0.4$. (b) $v_h = 1.0$; $v_h^T e_b = 0.8$; $h_c = 0.4$. (c) $v_h^T e_a = 1.0$; $h_c = 0.8$.

Figure 4.3. Examples of enlargement of human ellipses.

Secondly, the input to the DL network is changed as to incorporate this information. APF-RL only had current range data $q(t)$ and the previous one $q(t-\delta t)$ and the respective goal positions. Now, each data is labeled as human or not, $q_h(t)$. Thus, the observation is as follows $\omega(t) = (q(t), q(t-\delta t), g(t), g(t-\delta t), q_h(t))$ where δt is the time-step of processing. The motivation for using human labels is to make agent take extra precautions for humans.

In Social APF-RL, the set of learned parameters consisting of the k-parameter function and the waypoint function g is expanded to learning the h_c parameter function. Let $\mathcal{A} \subset \boldsymbol{R}^4$ be the action space. Each action $a \in \mathcal{A}$ is now defined by the 4-tuple $a = \begin{bmatrix} k & g^T & h_c \end{bmatrix}$ where $k \in \mathcal{K}$ defines the $k-$parameter, $g \in \mathcal{W}$ corresponds to an intermediate goal location and $h_c \in \mathcal{H}$ corresponds human ellipse enlargement coefficient . The set $\mathcal{K} \subset \boldsymbol{R}$ defines range for the k function. denotes the region where intermediate goals are selected in. It is a rectangular region with sizes of $2\tau_x \times \tau_y$ as shown in Fig. 3.8. It is defined in front of the robot to encourage it to move forward and explore. And the set $\mathcal{H} \subset \boldsymbol{R}$ defines range for the ch function. During training, the agent is continually rewarded considering goodness of the taken action.

The selection of reward function has significant effect on the performance of the agent since the reward is the only feedback given to it. Let $r : R \rightarrow [R_{min}, R_{max}]$ denote the reward function with the parameters R_{min}, R_{max} corresponding to the minimum

and maximum reward respectively. It is defined based on the proximity of the robot to the goal location as well as the obstacles sensed based on the observation $\omega(t)$. A penalty for intruding the personal zone of the humans is added in order to obtain a socially compliant navigation. Thus, it is defined as

$$r(t) = r_g\left(a(t), \omega(t)\right) + r_o\left(a(t), \omega(t)\right) + r_h\left(a(t), \omega(t)\right). \tag{4.5}$$

Here, the definitions of r_g and r_o are the same with those in the definition of APF-RL as formulated in Section 3.5.1. The human reward function r_h is defined as

$$r_h\left(a(t), \omega(t)\right) = \sum_{i=1}^{N_h} r_{hi}\left(a(t), \omega(t)\right) \quad where \tag{4.6a}$$

$$r_{hi}\left(a(t), \omega(t)\right) = \begin{cases} R_h(D_h - |c(t) - h_i(t)|) & \text{if } |c(t) - h_i(t)| < D_h, \\ 0 & \text{else.} \end{cases} \tag{4.6b}$$

Here, D_h denotes human distance threshold for social penalty and the penalty increases proportionally as the robot get closer to the pedestrian. R_h is the multiplier of the social penalty and it determines its importance against other rewards. In practice, human distance threshold D_h is set as 1.4 as the sum of personal zone distance and robot radius and R_h is set as 0.6 after testing with different values.

Soft Actor Critic is used as the deep learning method again for social navigation. Actor and critic networks have three hidden layers and each of them has 512 neurons. Hidden layer weights of the learned APF-RL model are used as initial values of this new model to jump-start the training.

Training environment for the social training is shown in Figure 4.5. First two rooms have only static obstacles because the agent needs to learn reaching the target and obstacle avoidance first. Then, the pedestrians are added with increasing numbers to each room. The training starts at the first room and continues with the next one

when it is learned. When all rooms are learned, a new room for training is selected at every 10 episodes and selection is done inversely proportional the respective success rate of the rooms. Training is continued until convergence of the average reward $\bar{r}(t)$ is obtained. Again, it is computed using last $N_T = 300$ episodes. The evolution of $\bar{r}(t)$ is shown in Figure 4.4. Its convergence occurs around 3500 episodes.

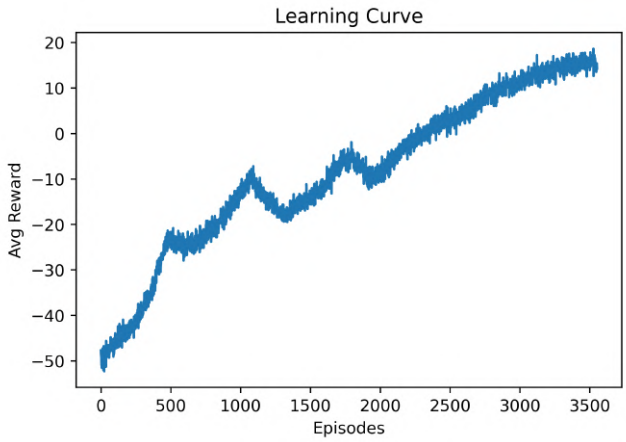


Figure 4.4. Learning Curve of Social APF-RL.

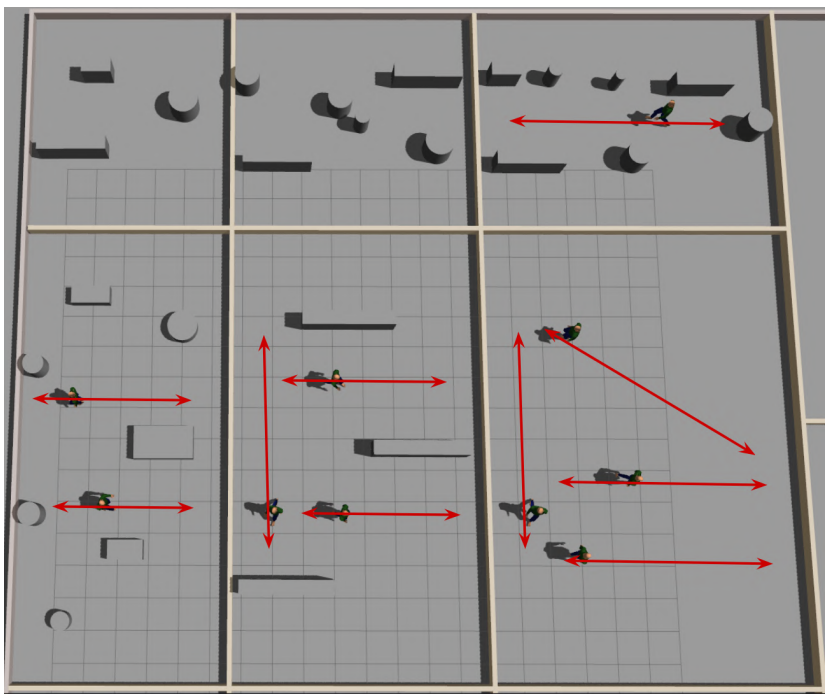


Figure 4.5. Training Environment of Social APF-RL.

4.5. Experimental Results

The proposed method, APF-RL, is tested both in simulation and with a physical robot. Testing environments are different than training environments and totally unseen by the agent. Performance metrics are selected as success rate (safe arrival at the goal), travel distance until reaching the goal and mean distance to humans in the environment. Path lengths are computed considering only successful runs and are meaningful only with high success rates.

To observe the novelty of the proposed method a comparative study is done. The classical social navigation method social force model is selected as a baseline. Also, APF and APF-RL are used as the other baselines.

4.5.1. Simulation Results

The evaluation is first done in the simulation environments. Nine test rooms are designed as can be seen in Figure 4.6. In each row, the static complexity of the room increases as the number of obstacles increases and the arrangement of obstacles is more complex. In each column, dynamic complexity which is represented by the number of obstacles increases. In the third row, humans walk in groups so it is additionally harder to avoid them. All methods are tested in these rooms with the same initial and target locations. The evaluation is done based on these 20 location couples and metrics are given as the average of them.

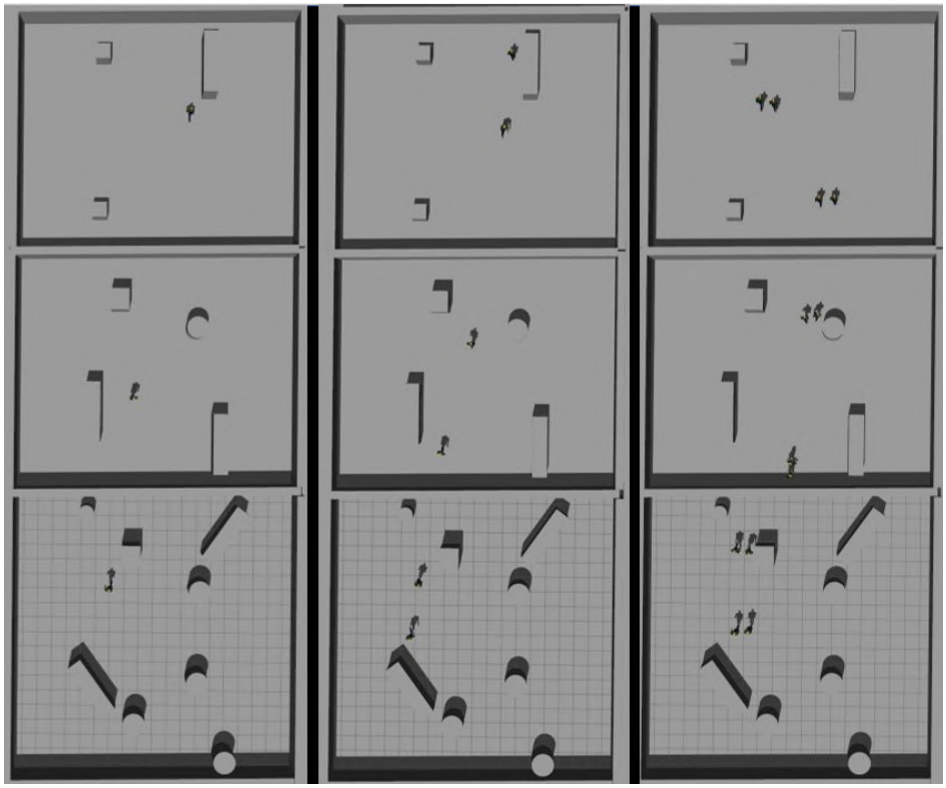


Figure 4.6. Test Environments for Social Navigation.

Results of the simulation tests are given in Table 4.1. The rooms are named based on the complexity levels and results are given for each of them. The number after "S" represents static complexity and the complexity level gets higher as the number increases. Same applies for dynamic complexity and it is represented by the number after "D". It can be seen that the proposed method has the best success rate in all scenarios. Its travel distance is usually higher than APF-RL which can be related to moving away from people around. The mean human distance of Social APF-RL is higher than both APF and APF-RL, and it is comparable to SFM. However, SFM has a much lower success rate, especially in complex environments. It is observed that SFM struggles at avoiding humans and obstacles at the same time. It has parameters for the weights of obstacle and human avoidance, they might need to be tuned based on the scenario for better performance.

Table 4.1. Socially Compliant Navigation: Simulation Results.

Methods	**Success Rate (%)**	**Travel Dist (m)**	**Human Dist (m)**	**Success Rate (%)**	**Travel Dist (m)**	**Human Dist (m)**	**Success Rate (%)**	**Travel Dist (m)**	**Human Dist (m)**
	S1D1			**S2D1**			**S3D1**		
APF	90	11.3	4.7	75	13.8	4.5	65	14.9	4.1
APF-RL	100	10.1	4.5	100	12.2	4.3	95	13.5	3.8
SFM	90	11.8	5.3	70	13.1	5.0	60	14.0	4.7
Social APF-RL	100	11.1	5.2	100	12.9	4.9	95	13.9	4.5
	S1D2			**S2D2**			**S3D2**		
APF	85	12.9	4.2	70	15.2	3.9	60	15.3	3.7
APF-RL	90	10.6	4.0	85	12.4	3.6	80	13.9	3.3
SFM	85	12.3	4.7	65	13.9	4.5	50	14.5	4.1
Social APF-RL	100	11.9	4.7	90	13.4	4.3	85	14.9	3.9
	S1D3			**S2D3**			**S3D3**		
APF	75	13.7	3.1	60	13.7	3.1	50	15.7	3.0
APF-RL	80	11.4	2.9	75	11.4	2.9	60	14.5	2.8
SFM	75	13.0	3.9	60	13.0	3.9	40	14.9	3.7
Social APF-RL	90	12.5	3.6	80	12.5	3.6	70	15.6	3.2

4.5.2. Real Robot Results

The real robot experiments are done using SempRob which is described in Chapter 2. ZED2 camera has its own human detection module and it outputs positions and velocities of the humans around. The social navigation performance is tested at 2 types of scenarios: i) only humans, ii) humans and static obstacles. The humans follow different paths like crossing the road and approaching from the opposite direction. An example test environment can be seen in Figure 4.7. Each scenario is repeated 5 times and evaluation is done based on the average of them.

Figure 4.7. Sample test environment for real robot experiments.

The results for the real robot experiments can be seen in Table 4.2. Similar to simulation results proposed method has the best performance in terms of success rate and mean distance to humans. Its path length performance is also on par with the best result by the APF-RL method. An example path followed by the robot using social APF-RL is shown in Figure 4.8. Here the green curve represents trajectory of the human, the yellow one represents the path of the robot. The robot successfully avoids obstacles and the pedestrian and reaches the goal location. Classical APF has a low success rate and human distance. It makes the robot oscillate back and forth around humans which might be scary for the people around. So it doesn't have socially compliant behavior. The social force model also has a low success rate. It fails to avoid obstacles when there is a human nearby and the robot gets stuck or collides with the

obstacles.

Table 4.2. Socially Compliant Navigation: Real Robot Results.

Method	**Success Rate (%)**	**Travel Dist. (m)**	**Human Dist. (m)**
APF	65	8,2	2,7
APF-RL	90	7,1	3,1
SFM	65	7,7	3,2
Social APF-RL	95	7,0	3,4

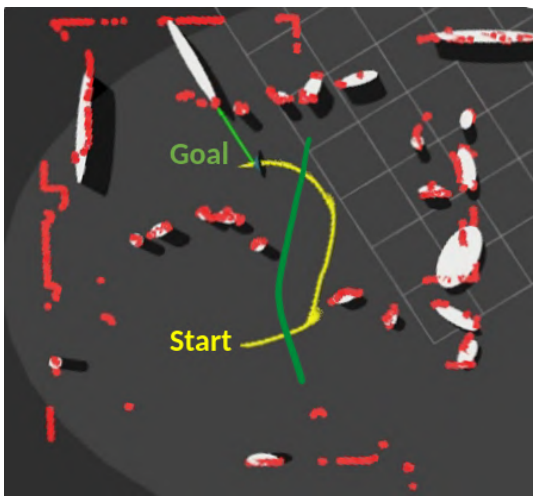


Figure 4.8. Social APF-RL Example: Human path (green) vs robot path (yellow). The robot successfully reaches its goal while also avoiding obstacles and respecting the comfort zone of the pedestrian.

5. Socially Compliant Human Following

The chapter focuses on socially compliant human following. Here, differing from previous navigation scenarios, the robot is expected to stay close to the accompanied person and not lose its sight of him/her. This requires the robot to be capable of detecting and tracking the target person. Furthermore, its movement must be sufficiently fast and smooth. Simultaneously, it shall avoid all collisions and should not disturb other people around. Thus, the navigation method needs to be both reactive and social.

The outline of the chapter is as follows: Firstly, related literature is summarized in Section 5.1. Next, human following method is explained in detail in Section5.2. The chapter concludes with an experimental evaluation - including both simulation and real robot results in Section 5.3.

5.1. Related Literature

Most work in human following has focused on how to control the robot to follow the detected human. One of the early work in this area has proposed a simple curvature-velocity model [54] - based on [55]. A vision based method that makes use of a simple neural controller has been proposed in [56]. Ellipse-shaped social zones are to be avoided using a social force based model in [57]. However, it doesn't have a human detection system on the robot and it uses scene information obtained from a ceiling camera. Long-short term memory based human motion prediction with model predictive controller is proposed for human following in complex environments [58].

Figure 5.1. Flow of processing for human tracking.

5.2. Human Following

In the human following task, the target location dynamically changes based on the location of the followed human. This differentiates human following from the previously discussed navigation scenarios. The target might be directly selected as the human location, however, this location is occupied by the human. In order to eliminate the impact of the movement of the robot, Kalman filter calculations are done in a fixed frame which is created using localization of the robot. So it might result in failure to find a path for search-based planners and oscillations or collision for reactive methods. Instead of directly using human location, a point around the human must be selected.

In the presence of the other humans in the scene, the human detected first is selected as the target. In case multiple humans are detected, the detection which is nearest to the target's last detection is fed to the target localization and target modules.

The target location relative to the human location can be selected based on the preferred companion location. The preference can be represented with two parameters as shown in Figure 5.2:

- Following angle θ_f,
- Following distance d_f

All possible locations are in the back of the human because the robot needs to keep visual contact. The limits of the following angle must be set based on the horizontal field of view of the used camera. In order to determine which side is the back of the human,the moving direction is used. The direction is calculated based on the velocity output of the tracking module.

Once the human is detected and the target location is selected, it is given to a navigation module to follow the target. For navigation, both APF-RL and Social APF-RL methods are used. If the visual contact with the target person is lost, the

robot goes to the latest target and makes a 360 degrees rotation to in order to gain its sight again.

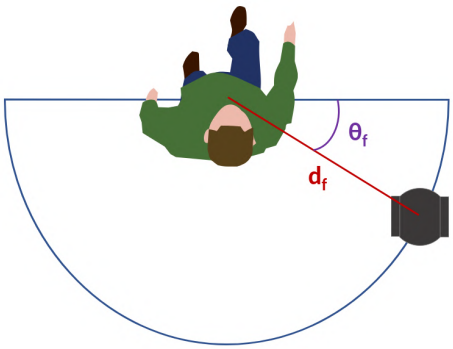


Figure 5.2. Human Following Location Selection.

5.3. Experimental Results

Human following is tested both in simulation and with a physical robot. The following distance r_f is set as 1.5 m and the following angle θ_f is set as 180° which means the robot will follow the human from the back. Two performance metrics are computed:

- Average distance error: It calculated as the robot's distance to the target location.
- Human path deviation: The deviation of the robot's path from the path followed by the human is computed.

APF-RL and Social APF-RL methods are tested and their performances are compared.

5.3.1. Simulation Results

First, simulation tests are done. A more realistic house-like test environment is used for simulation tests. A single person is added to this room and a set of way-points are given to it for each scenario. The simulated person follows the path with constant velocity as long as the path is not occupied. The robot is placed 1.5 meters back of the human initially and the test starts with the movement of the human. The simulation environment and sample paths followed by the human and the robot are shown in

Figure 5.3. Here, the blue path is that of the human while the yellow one corresponds to that of the robot's.

Figure 5.3. Human following simulation: Human path vs robot path.

Tests are repeated for 5 times for each trajectory and metrics are calculated as the average. The results are given in Table 5.1. As can be seen in the results, there is no significant difference between these methods. They both can successfully follow the human.

Table 5.1. Human Following: Simulation Results.

Method	**Follow Dist. Error (m)**	**Path Deviation (m)**
APF-RL	0.35	0.21
Social APF-RL	0.31	0.24

5.3.2. Real Robot Results

Real robot experiments are done using SempRob. Two scenarios are considered: i) A long square corridor, ii) Straight with obstacles. In the first scenario, there are no obstacles except the walls around. The human completes a full lap which is approximately 80 meters and the robot follows it. In the second, the human moves straight

while avoiding obstacles and the robot tries to keep following without collision. Environment for the second scenario is shown in Figure 5.4. Example paths followed by the human and the robot in this environment is shown in Figure 5.5. Here, the blue trajectory shows the path of the human and the yellow one shows the robot's path.

Figure 5.4. Example Following Environment.

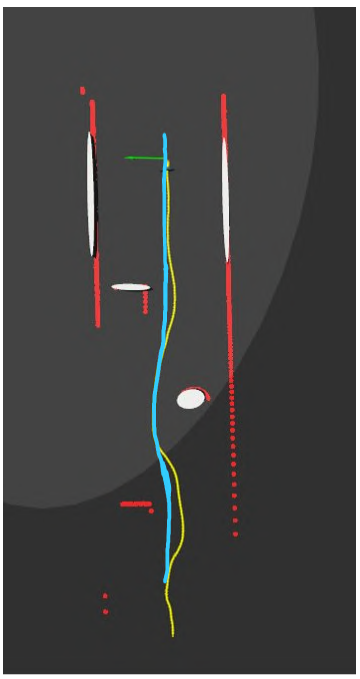


Figure 5.5. Following Paths Comparison.

Tests are repeated for 5 times for both scenarios and metrics are calculated as the average. The results are given in Table 5.2. Similar to simulation results, there is no significant difference between these methods as can be seen in the results. Both enable the robot to follow the human successfully. Interestingly, they both perform better in long scenario that doesn't have obstacles. In the presence of obstacles, they sometimes struggle and lose time around obstacles so their performance metrics get worse.

Table 5.2. Human Following: Real Robot Experiments.

	Long Scenario		**Obstacle Scenario**	
Method	**Follow Dist. Error (m)**	**Path Deviation (m)**	**Follow Dist. Error (m)**	**Path Deviation (m)**
APF-RL	0.23	0.12	0.38	0.25
Social APF-RL	0.21	0.13	0.39	0.27

6. CONCLUSION AND FUTURE WORK

This thesis is focused on the design and development of a social robot that can navigate around in a socially compliant manner. In this thesis, this problem is addressed in two parts. The first part has focused on the physical design and development of a social robot named as SempRob. In the second part, the social navigation capability of the social robot is developed.

Chapter 2 has presented the development of a social robot. It is designed by taking likability and usability into consideration. After manufacturing the body and head parts, it is equipped with the necessary visual sensing hardware as to perceive the environment. It is used for experiments of this thesis and it can be used for a variety of daily-life tasks. As future work, head rotation capability can be added by using a pan-tilt mechanism.

Chapter 3 has presented a novel mapless robot navigation method. This method combines artificial potential functions and deep reinforcement learning in order to obtain safe and efficient navigation. The method is fully trained via a simulation environment and the learned functions are then successfully transferred to the real robot. Differing from the previous work, this method does not require map of the environment and it can be used even in complex environmental settings as well. Additionally, ellipse representation of obstacles is introduced and novel environmental complexity measures are proposed in this chapter.

Chapter 4 has extended APF-RL to human-populated environments to obtain a socially compliant navigation method - namely Social APF-RL. The robot using this method takes social conventions into account while navigating and aims to stay away from the personal zones of pedestrians around as much as possible. Human detection is also added to the robot for this task. In the future, human-likeness of the trajectory of the robot can be improved by comparing and using human trajectory data sets.

Finally, Chapter 5 is on adaptation of proposed navigation methods for the human following task. These methods are updated to make use of dynamic goal locations. The goal is placed at a point behind the human based on the human preferences. Human detection is augmented with human tracking capability. Human following performance is experimentally evaluated both in simulations and on the developed SempRob robot. In the future, human following can be tested in crowded areas based on social compliancy and can be updated according the results.

REFERENCES

1. Kruse, T., A. K. Pandey, R. Alami and A. Kirsch, "Human-Aware Robot Navigation: A Survey", *Robotics and Autonomous Systems*, Vol. 61, No. 12, pp. 1726 – 1743, 2013.

2. Dautenhahn, K., "Socially Intelligent Robots: Dimensions of Human-Robot Interaction.", *Philosophical transactions of the Royal Society of London. Series B, Biological Sciences*, Vol. 362 1480, pp. 679–704, 2007.

3. Charalampous, K., I. Kostavelis and A. Gasteratos, "Recent Trends in Social Aware Robot Navigation: A Survey", *Robotics and Autonomous Systems*, Vol. 93, pp. 85–104, 2017.

4. Mathur, M. B. and D. B. Reichling, "Navigating a Social World with Robot Partners: A Quantitative Cartography of the Uncanny Valley", *Cognition*, Vol. 146, pp. 22 – 32, 2016.

5. MORI, M., "Bukimi No Tani The Uncanny Valley", *Energy*, Vol. 7, pp. 33–35, 1970.

6. Kubota, T., "JackRabbot 2: The Polite Pedestrian Robot | Stanford University School of Engineering", , Sep. 2018, `https://engineering.stanford.edu/magazine/article/jackrabbot-2-polite-pedestrian-Robot`, accessed in December 2021.

7. Pandey, A. K. and R. Gelin, "A Mass-Produced Sociable Humanoid Robot: Pepper: The First Machine of Its Kind", *IEEE Robotics & Automation Magazine*, Vol. 25, pp. 40–48, 2018.

8. Sun, H., H. Zhou, X. Li, Y. Wei and X. Li, "Design of Two-Wheel Self-Balanced Electric Vehicle Based on MEMS", *4th IEEE International Conference*

on Nano/Micro Engineered and Molecular Systems, pp. 143–146, 2009.

9. Murtagh, A., "Robot Faces", `https://github.com/AndrewMurtagh/Robot_faces`, 2021, accessed in October 2021.

10. Patle, B., G. Babu L, A. Pandey, D. Parhi and A. Jagadeesh, "A Review: On Path Planning Strategies for Navigation of Mobile Robot", *Defence Technology*, Vol. 15, No. 4, pp. 582–606, 2019.

11. Esan, O., S. Du and B. Lodewyk, "Review on Autonomous Indoor Wheel Mobile Robot Navigation Systems", *International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–6, 2020.

12. Rimon, E. D. and D. E. Koditschek, "Exact Robot Navigation Using Artificial Potential Functions", *IEEE Transactions Robotics and Automation*, Vol. 8, pp. 501–518, 1992.

13. Bishop, C. M., "Novelty Detection and Neural Network Validation", *IEE Proceedings - Vision, Image and Signal Processing*, Vol. 141, No. 4, pp. 217–222, 1994.

14. Fox, D., W. Burgard and S. Thrun, "The Dynamic Window Approach to Collision Avoidance", *IEEE Robotics and Automation Magazine*, Vol. 4, pp. 23–33, 1997.

15. Karagöz, C. S., H. I. Bozma and D. E. Koditschek, "Coordinated Navigation of Multiple Independent Disk-Shaped Robots", *IEEE Transactions on Robotics*, Vol. 30, No. 6, pp. 1289–1304, 2014.

16. Lionis, G., X. Papageorgiou and K. J. Kyriakopoulos, "Locally Computable Navigation Functions for Sphere Worlds", *IEEE International Conference on Robotics and Automation*, pp. 1998–2003, April 2007.

17. Filippidis, I. and K. J. Kyriakopoulos, "Adjustable Navigation Functions for Unknown Sphere Worlds", *50th IEEE Conference on Decision and Control and Eu-*

ropean Control Conference, pp. 4276–4281, Dec 2011.

18. Conner, D. C., H. Choset and A. A. Rizzi, "Integrating Planning and Control for Single-Bodied Wheeled Mobile Robots", *Autonomous Robots*, Vol. 30, No. 3, pp. 243–264, Apr 2011.

19. Arslan, O. and D. E. Koditschek, "Sensor-Based Reactive Navigation in unknown convex sphere worlds", *The International Journal of Robotics Research*, Vol. 38, pp. 196 – 223, 2019.

20. Vasilopoulos, V., W. Vega-Brown, Ö. Arslan, N. Roy and D. E. Koditschek, "Sensor-Based Reactive Symbolic Planning in Partially Known Environments", *IEEE International Conference on Robotics and Automation*, pp. 1–5, 2017.

21. Richter, C., W. Vega-Brown and N. Roy, "Bayesian Learning for Safe High-Speed Navigation in Unknown Environments", A. Bicchi and W. Burgard (Editors), *International Symposium on Robotics Research*, Vol. 3, pp. 325–341, Springer, 2015.

22. Duan, Y., J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever and P. Abbeel, "RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning", *Computing Research Repository (CoRR)*, Vol. abs/1611.02779, 2016.

23. Tai, L., G. Paolo and M. Liu, "Virtual-to-Real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 31–36, 2017.

24. Pan, Y., C. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou and B. Boots, "Agile Off-Road Autonomous Driving Using End-to-End Deep Imitation Learning", *Computing Research Repository (CoRR)*, Vol. abs/1709.07174, 2017.

25. Faust, A., O. Ramírez, M. Fiser, K. Oslund, A. Francis, J. O. Davidson and L. Tapia, "PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning", *IEEE International Confer-*

ence on Robotics and Automation, pp. 5113–5120, 2017.

26. Savinov, N., A. Dosovitskiy and V. Koltun, "Semi-Parametric Topological Memory for Navigation", *Computing Research Repository (CoRR)*, Vol. abs/1803.00653, 2018.

27. Bansal, S., V. Tolani, S. Gupta, J. Malik and C. J. Tomlin, "Combining Optimal Control and Learning for Visual Navigation in Novel Environments", *Computing Research Repository (CoRR)*, Vol. abs/1903.02531, 2019.

28. Chiang, H.-T. L., J. Hsu, M. Fiser, L. Tapia and A. Faust, "RL-RRT: Kinodynamic Motion Planning via Learning Reachability Estimators From RL Policies", *IEEE Robotics and Automation Letters*, Vol. 4, pp. 4298–4305, 2019.

29. Pfeiffer, M., M. Schaeuble, J. Nieto, R. Siegwart and C. Cadena, "From Perception to Decision: A Data-Driven Approach to End-to-End Motion Planning for Autonomous Ground Robots", *IEEE International Conference on Robotics and Automation*, pp. 1527–1533, 2016.

30. Halır, R. and J. Flusser, "Numerically Stable Direct Least Squares Fitting of Ellipses", *Proc. 6th International Conference in Central Europe on Computer Graphics and Visualization. WSCG*, Vol. 98, pp. 125–132, Citeseer, 1998.

31. Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous Control with Deep Reinforcement Learning", *Computing Research Repository (CoRR)*, Vol. abs/1509.02971, 2016.

32. Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. A. Riedmiller, "Playing Atari with Deep Reinforcement Learning", *Computing Research Repository (CoRR)*, Vol. abs/1312.5602, 2013.

33. Haarnoja, T., A. Zhou, P. Abbeel and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", *Inter-*

national Conference on Machine Learning (ICML), 2018.

34. Schulman, J., S. Levine, P. Abbeel, M. I. Jordan and P. Moritz, "Trust Region Policy Optimization", *International Conference on Machine Learning (ICML)*, 2015.

35. Schulman, J., F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms", *Computing Research Repository (CoRR)*, Vol. abs/1707.06347, 2017.

36. Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning", *International Conference on Machine Learning (ICML)*, 2016.

37. Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous Control with Deep Reinforcement Learning", *Computing Research Repository (CoRR)*, Vol. abs/1509.02971, 2015.

38. Hall, E. T. and E. T. Hall, *The Hidden Dimension*, Vol. 609, Anchor, 1966.

39. Daza, M., D. Barrios-Aranibar, J. Diaz-Amado, Y. Cardinale and J. P. Vilasboas, "An Approach of Social Navigation Based on Proxemics for Crowded Environments of Humans and Robots", *Micromachines*, Vol. 12, 2021.

40. Pandey, A. K. and R. Alami, "A Framework Towards a Socially Aware Mobile Robot Motion In Human-Centered Dynamic Environment", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5855–5860, 2010.

41. Helbing and Molnar, "Social Force Model for Pedestrian Dynamics.", *Physical Review, Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, Vol. 51 5, pp. 4282–4286, 1995.

42. Ferrer, G., A. Garrell and A. Sanfeliu, "Robot Companion: A Social-Force Based Approach With human Awareness-Navigation in Crowded Environments",

IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1688–1694, 2013.

43. Ferrer, G., A. Garrell and A. Sanfeliu, "Social-Aware Robot Navigation in Urban Environments", *2013 European Conference on Mobile Robots*, pp. 331–336, 2013.

44. Everett, M., Y. F. Chen and J. P. How, "Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3052–3059, 2018.

45. Chen, C., Y. Liu, S. Kreiss and A. Alahi, "Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning", *International Conference on Robotics and Automation (ICRA)*, pp. 6015–6022, 2018.

46. Jin, J., N. M. Nguyen, N. Sakib, D. E. Graves, H. Yao and M. Jägersand, "Mapless Navigation Among Dynamics with Social-Safety-Awareness: A Reinforcement Learning Approach From 2D Laser Scans", *Computing Research Repository (CoRR)*, Vol. abs/1911.03074, 2019.

47. Sathyamoorthy, A. J., U. Patel, T. Guan and D. Manocha, "Frozone: Freezing-Free, Pedestrian-Friendly Navigation in Human Crowds", *IEEE Robotics and Automation Letters*, Vol. 5, pp. 4352–4359, 2020.

48. Vasquez, D., B. Okal and K. O. Arras, "Inverse Reinforcement Learning Algorithms and Features for Robot Navigation in Crowds: An Experimental Comparison", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1341–1346, 2014.

49. Okal, B. and K. O. Arras, "Learning Socially Normative Robot Navigation Behaviors with Bayesian Inverse Reinforcement Learning", *IEEE International Confer-*

ence on Robotics and Automation (ICRA), pp. 2889–2895, 2016.

50. Chen, Y. F., M. Everett, M. Liu and J. P. How, "Socially Aware Motion Planning with Deep Reinforcement Learning", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1343–1350, 2017.

51. Hamandi, M., M. D'Arcy and P. Fazli, "DeepMoTIon: Learning to Navigate Like Humans", *28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–7, 2019.

52. Redmon, J., S. K. Divvala, R. B. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

53. Bertoni, L., S. Kreiss and A. Alahi, "MonoLoco: Monocular 3D Pedestrian Localization and Uncertainty Estimation", *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6860–6870, 2019.

54. Gockley, R., J. Forlizzi and R. G. Simmons, "Natural Person-Following Behavior for Social Robots", *2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 17–24, 2007.

55. Simmons, R. G., "The Curvature-Velocity Method for Local Obstacle Avoidance", *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 4, pp. 3375–3382 vol.4, 1996.

56. Capi, G., H. Toda and T. Nagasaki, "A Vision Based Robot Navigation and Human Tracking for Social Robotics", *IEEE International Workshop on Robotic and Sensors Environments*, pp. 1–6, 2010.

57. Herrera, D., M. Monllor, D. Santiago, F. Roberti and R. O. Carelli, "Null-Space Based Control for Human Following and Social Field Avoidance", *XVII Workshop on Information Processing and Control (RPIC)*, pp. 1–6, 2017.

58. Ashe, A. K. and K. M. Krishna, "Followman: Control of Social Person Following Robot", *IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 3590–3595, 2021.

7. APPENDIX A: ROBOT MANUAL

The detailed specifications of the base platform are given in Table 7.1. Camera and lidar sensors are placed on this robot and a Nvidia Xavier AGX computer is used to run software modules. All sensors and computer get power from the robotic platform. The platform can be charged using an external power supply. The charging steps are as follows:

- Make sure the the robot is powered off.
- Connect the External Power Supply to the charge port (Connector IV).
- Plug the power cord into the IEC connector on the External Power Supply and into a grounded AC outlet (100 – 240 V, 50 – 60 Hz).
- Toggle the power switch on the External Power Supply to the ON (l) position.
- When charging is complete, toggle the power switch to the OFF position, unplug the External Power Supply from the grounded AC outlet, and disconnect the External Power Supply from the robot.

Table 7.1. Characteristics of the Base Platform.

Characteristic	**Value**	**Unit**
Max. Speed	8	m/s
Turn Radius	0	m
Max. Slope	10	°
Peak Torque (Each Wheel)	100	Nm
Maximum Range	50	km
Max. Run Time	24	h
Max. Payload	181	kg

To start the robot first toggle the power switch to the ON position. This will power the base platform. Then to connect to it power on the onboard Nvidia Xavier computer. After logging into Xavier, run start_all.sh that will start connection to the

robot and the sensors. If there is a connection error, check ip addresses of ethernet connections. It should be 192.168.0.100 for the robot and 192.168.1.102 for the lidar.

8. APPENDIX B: SOFTWARE USAGE

After the robot and sensors are powered and start_all script is run, the robot is ready to perform navigation tasks.

- To use APF-RL, run:
  roslaunch apfrl apfrl.launch
  This will start obstacle generator, APF-RL and APF nodes. It will also open the visualization module, here you can give it a target using "2D Nav Goal" tool.
- To use Social APF-RL, run:
  roslaunch apfrl social_apfrl.launch
  This will start obstacle generator, Social APF-RL and APF nodes. Again it will be waiting for a target.
- To use human following, start APF-RL or Social APF-RL first. Then run:
  rosrun apfrl human_goal_pub_node
  This will publish target location based on the human detection.

9. APPENDIX C: IMAGE USAGE

The images are generated within the scope of this thesis and whose copyright have been transferred to the publishers, were used in accordance with the publication policy of the publisher.

The human figure in Figure 5.2 is licensed with Creative Commons so it can be used freely without asking for permission.

Figure 5.3 depicts a simulation environment which is distributed freely so it can be used without restriction.